

University of Hong Kong
Department of Computer Science
COMP4801 Final Year Project



CritiQ

*Game Testing and Evaluation Platform with Machine Learning for
Game Developers*

Final Report

Supervisor: Dr. Chim T. W.

Second Examiner: Professor Zhao, Hengshuang

Group Members:

Lee Chi Ho 3035785520

Cheng Pak Yim 3035784942

Siu Yuk Shing 3035779430

Submitter:

Siu Yuk Shing 3035779430

Date of Submission: April 26, 2024

Abstract

This project addresses the limited application of Natural Language Processing (NLP), particularly Language Model Libraries (LLMs), in the gaming industry and game reviews. Recognizing a gap in existing game review platforms such as Steam, Openritic, and Metacritic, which predominantly serve end-users and offer limited functionalities in terms of review analysis and filtering, a web-based game review platform called CritiQ is developed. CritiQ is a complimentary game review platform that leverages NLP to provide a variety of services including automated review analysis, generation of aggregated review summaries, and analytical features pertaining to game reviews such as review length and sentiment distribution. The project outcomes are categorized into three main sections: frontend, backend, and machine learning. The frontend concludes in a responsive, user-friendly, well-designed, and highly accessible web application compatible with various devices. The backend solution, hosted on cloud platforms, utilizes a comprehensive CI/CD pipeline for swift development and deployment processes. Machine learning models, including sentiment analysis, topic modeling and large language models, are effectively implemented with high precision and performance. CritiQ serves as a significant contributor to the gaming industry by streamlining the game development process and facilitating an efficient feedback loop between game developers and players. It also provides substantial benefits to the general gaming community as it enables them to utilize the platform for game discovery, expressing their viewpoints about various games, and collectively enhancing the gaming experience through constructive feedback. The project demonstrates the feasibility and benefits of integrating NLP into the analysis of game review. It concludes that with a thoughtfully constructed system architecture, NLP tools can be smoothly incorporated into pre-existing systems, thereby enhancing the understanding of game developers and users about the strengths of the games and areas for improvement.

Acknowledgment

We would like to express our deepest gratitude and appreciation to the project supervisor, Dr. Chim T.W., for his invaluable guidance and unwavering support since the start of the project. We are grateful for his expertise, and patience in mentoring from the beginning of the project.

Table of Contents

<i>Abstract</i>	<i>ii</i>
<i>Acknowledgment</i>	<i>iii</i>
<i>List of Figures</i>	<i>vi</i>
<i>List of Tables</i>	<i>ix</i>
1. Introduction	1
1.1. Background.....	1
1.2. Objectives.....	2
1.3. Deliverables.....	2
1.4. Contributions.....	2
1.5. Outline.....	2
2. Related Work	3
3. Methodology	6
3.1 Machine Learning and Natural Language Processing	6
3.1.1 Sentiment Analysis.....	6
3.1.2 Topic Modelling.....	20
3.1.3 Keyword Extraction.....	23
3.2 Frontend Web Application	25
3.2.1 Technologies Involved.....	25
3.2.2 Design Approach.....	25
3.2.3 Frontend Authentication.....	26
3.3 Backend Technologies	28
3.3.1 Spring Boot Server Application.....	28
3.3.2 Authentication with JWT.....	29
3.3.3 Email Service.....	30
3.3.4 Database.....	31
3.3.5 Object Storage.....	32
3.3.6 Continuous Integration/Continuous Delivery (CI/CD).....	33
3.3.7 Message Queue.....	34
3.3.8 Hosting.....	36
3.3.9 Monitoring.....	37
4 Result	38
4.1 Natural Language Processing Tasks	38
4.1.1 Sentiment Analysis.....	38
4.1.2 Topic Modeling.....	46
4.1.3 Keyword Extraction and LLM Prompting.....	50
4.2 Web Application	53
4.2.1 Toolbar.....	54
4.2.2 Login and Registration.....	56
4.2.3 Forgot Password Page and Reset Password Page.....	59
4.2.4 Landing Page.....	62
4.2.5 Search Result Page.....	65
4.2.6 Game Page.....	69
4.2.7 Game Reviews Page.....	75
4.2.8 Review Page.....	76
4.2.9 Game Analytics Page.....	79
4.2.10 Profile Page.....	83

4.3	Web Scraping.....	87
4.4	Backend System.....	88
4.4.1	CI/CD	89
4.4.2	API Endpoints and Database	90
4.4.3	API Security	92
4.4.4	S3 Bucket Storage	94
4.4.5	Stability and Testing	95
5	<i>Difficulties and Limitations</i>	96
5.1	HTTPS and SSL Certificate	96
5.2	Message Queue Disconnection	98
5.3	Next.js Compatibility and Debugging Issues	100
5.4	Dataset Noise.....	101
5	<i>Project Schedule</i>	103
6	<i>Work Distribution</i>	103
7	<i>Conclusion and Future Works.....</i>	104

List of Figures

Figure 1: Usual stages in Sentiment Classification. (a): Six usual stages in Sentiment Classification. (b): Overall framework of section Sentiment Analysis of the project. Two additional stages were added and labeled in pink.	7
Figure 2: Steam automated comment filtering. The sensitive word was replaced by consecutive heart symbols.	8
Figure 3: Number of reviews in each sentiment class in the cleaned dataset with a 5.14: 1 positive to negative ratio.	10
Figure 4: Procedure of further data cleaning on the cleaned dataset.	10
Figure 5: Number of appearances of top 20 frequent words in the cleaned dataset after further data cleaning. (a): Words in the cleaned dataset with both positive and negative reviews. (b): Words in the cleaned dataset with only positive reviews. (c): Words in the cleaned dataset with only negative reviews. (d): Common words in (b) and (c).	10
Figure 6: Model structure of CNN in model GloVe-CNN.	15
Figure 7: Customized further data cleaning to each model. Left: TFIDF-RF. Middle: GloVe-CNN. Right: BERT.	17
Figure 8: Example output of BERTopic using KeyBERT and Llama2 to name the topics. ...	23
Figure 9: Two game reviews and the response from ChatGPT hosted by Azure when prompting to classify their sentiment.	24
Figure 10: JWT Claims extracted from the Access Token user received on login.	30
Figure 11: Database Entity Relations Diagram.	31
Figure 12: Sample Code to upload a file to the S3 Bucket.	32
Figure 13: Digital Ocean Spaces Dashboard.	33
Figure 14: Jenkinsfile pipeline written for deploying the Backend Server and NLP Server with the use of docker and dockerfiles.	33
Figure 15: Message Queue Structure for Supporting Inter-process Machine Learning Application.	35
Figure 16: RabbitMQ Management Panel showing all the queue connections.	35
Figure 17: Grafana Dashboard displaying uptime, CPU, and Memory Utilization by the Spring Boot Application.	37
Figure 18: Grafana Dashboard shows the database connection pool size, maintaining a stable connection to the database.	37
Figure 19: Results of all models trained with 120K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.	38
Figure 20: Precision of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative.	39
Figure 21: Recall of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative.	40
Figure 22: Results of all models trained with 240K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.	40
Figure 23: Results of all models trained with 480K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.	40
Figure 24: Results of all models trained with balanced datasets of all three sizes. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.	41

Figure 25: Results of all models trained with imbalanced datasets of all three sizes. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models	42
Figure 26: ROC-AUC of all models trained with the balanced dataset. (a): on the imbalanced validation set. (b): on the balanced validation set.....	44
Figure 27: ROC curve of BERT fine-tuned on 240K balanced training set. (a): on the imbalanced validation set. (b): on the balanced validation set.	44
Figure 28: Median inference time of original and ONNX model on different machines. (a): Machine 1 (Windows i5-8250U). (b): Machine 2 (Apple M1 Max)	45
Figure 29 List of game genres and the review frequencies	46
Figure 30 NPMI score of models trained with different architectures and reviews on games in the action genre	47
Figure 31 Topic name results after applying the pre-trained topic model on action game reviews to the game Starfield.....	49
Figure 32 Procedure of aggregated review summary generation with LLM and prompting ..	51
Figure 33 Procedure of keyword extraction with LLM and prompting.....	51
Figure 34 Prompting template to extract keywords from the reviews for the LLM.....	52
Figure 35 An example prompt and the returned response from the LLM	53
Figure 36 Web application's toolbar design. (a): Toolbar design for desktop viewport. (b): Toolbar design for mobile viewport. (c): Avatar icon button drop down menu.....	54
Figure 37 Web application's register modal popup	56
Figure 38 Register modal input validations	56
Figure 39 Web application's login modal popup	57
Figure 40 Web application's forget password page design.....	59
Figure 41 Reset password email	59
Figure 42 Web application's reset password page design.....	60
Figure 43 Web application's reset password page with invalid token	61
Figure 44 Web application's landing page design. (a): Landing page design for desktop viewport. (b): Landing page design for mobile viewport. (c) Game card component.....	62
Figure 45 Search result page searching example	65
Figure 46 Web application's search result page design. (a): Search result page design for desktop viewport. (b): Search result page design for mobile viewport	65
Figure 47 Advanced search modal for search result page	66
Figure 48 Web application's game page design. (a): Game page design for desktop viewport. (b): Game page design for mobile viewport.	69
Figure 49 Game detailed information popup modal	70
Figure 50 Add review section design. (a): Implementation for authenticated user that have not made a review to the game. (b): Implementation for authenticated user that made a review to the game	71
Figure 51 Game review card design. (a): Game review card under desktop viewport. (b): Game review card under mobile viewport.....	73
Figure 52 Web application's game reviews page design. (a): Game reviews page design for desktop viewport. (b): Game reviews page design for mobile viewport	75
Figure 53 Web application's review page design, review context has been shortened to reduce the length of the screenshots. (a): Review page design for desktop viewport. (b): Review page design for mobile viewport.	76
Figure 54 Game analytics page design for desktop viewport	79
Figure 55 Game analytics page design for mobile viewport. The Image is cropped as the page is too long to display	79
Figure 56 Game Analytics Page sections tracking feature	82

Figure 57 Web application's profile page design under the owner view. (a): Profile page design for desktop viewport. (b): Profile page design for mobile viewport.	83
Figure 58 Update profile banner modal	84
Figure 59 Update username modal	84
Figure 60 Profile page control panel for the profile owner	84
Figure 61 Update user avatar modal. (a) Modal before image upload. (b) Modal after image upload, with avatar editor backdrop to set the icon. (c) Modal after the image is uploaded and the icon is set.....	85
Figure 62 Web application's profile page design under visitor view and private profile. (a): Private profile page design for desktop viewport. (b): Private profile page design for mobile viewport.	86
Figure 63 Sample Database Records of Scraped Games from Steam	87
Figure 64 Backend Solution Architecture Graph.....	88
Figure 65 Jenkins deployment User Interface with different stages of deployments.	89
Figure 66 API call to /findGameById to fetch specific game information finishes in 65ms ..	90
Figure 67 API call to /findGamesWithSearch to perform exhaustive game search finishes in 251ms.....	91
Figure 68 Current Database Plan with 2GB RAM offer a maximum of 150 concurrent connections	91
Figure 69 Spring Boot utilizes 80 concurrent connections to the database	91
Figure 70 Access Control by verifying the user based on the JWT sent in HTTP requests using the @AuthenticationPrincipal annotation.	92
Figure 71 Access Control by verifying the user's role based on JWT sent in HTTP requests prior to method invocation using the @PreAuthorize annotation.	92
Figure 72 403 Forbidden Error on Unauthorized Access to Protected API endpoints.....	93
Figure 73 Fetching of Cached Image(s) can be performed within 50ms.....	94
Figure 74 Uploader Header is set to the user's name during file upload	94
Figure 75 Backend Load-Testing using Gatling, showing the ability to sustain 60 active users performing complex queries.	95
Figure 76 Google Chrome Browser Error Page on Visiting Website with a self-signed digital certificate.....	96
Figure 77 Certificate Viewer on Backend Domain Address using Google Chrome.....	98
Figure 78: Code snippet to initiate the RabbitMQ connect with heartbeat check disabled. ..	100

List of Tables

Table 1: Percentage change of all models trained with balanced datasets with different sizes. Up: Weighted Average F1-score. Bottom left: Weighted Average Precision. Bottom right: Weighted Average Recall.	41
Table 2: Percentage change of all models trained with imbalanced datasets with different sizes. Up: Weighted Average F1-score. Bottom left: Weighted Average Precision. Bottom right: Weighted Average Recall.....	42
Table 3: Weighted Average F1-Score of all models on the imbalanced validation sets.	43
Table 4: Weighted Average F1-score of all models on the balanced validation sets.	43
Table 5: Median Speedup of inference time of both machines.	45
Table 6 Representative texts and topic keywords generated by the topic models on the topic of "crashes and bugs"	48
Table 7 Representative texts and topic keywords generated by the topic models on the topic of "horror puzzle game".....	48
Table 8 Proposed Schedule for the project	103
Table 9 Work Distribution Table of the project.....	103

Abbreviations

Abbreviations and Acronyms	Full Term / Definition
ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag-of-Words
CA	Certificate Authority
CDN	Content Delivery Network
CI/CD	Continuous Integration/Continuous Delivery
CNN	Convolutional Neural Network
CRUD	Create, Read, Update, Delete
CSS	Cascading Stylesheets
CTM	Contextualized Topic Model
EDA	Exploratory Data Analysis
ESM	ECMAScript Modules
GloVe	Global Vectors for Word Representation
HTTP	Hypertext Transfer Protocol
LDA	Latent Dirichlet Allocation
LLM	Large Language Model
ML	Machine Learning
MUI	Material UI
NLP	Natural Language Processing
NPMI	Normalized pointwise mutual information
RAG	Retrieval-Augmented Generation
RBO	Rank-Biased Overlap
RTT	Round-Trip-Time
RWD	Responsive Web Design
SDK	Software Development Kit
SSL	Secure Socket Layer
SSR	Server-Side Rendering
TF-IDF	Term Frequency-Inverse Document Frequency
UI	User Interface
UI/UX	User Interface/User Experience
URL	Uniform Resource Locator
VPS	Virtual Private Server

1. Introduction

This section discusses the Project Background (Section 1.1), Objectives (Section 1.2), Deliverables (Section 1.3), Contributions (Section 1.4) and Outline (Section 1.5).

1.1. Background

The emergence of game review platforms such as Steam, Openritic, and Metacritic are predominantly designed to serve the needs of end-users. Feedback from game reviews can provide valuable feedback on bug fixes and game design suggestions (Lin et al., 2019). However, these platforms offer limited functionalities particularly in terms of review analysis and filtering. Consequently, this poses a challenge for developers seeking to derive meaningful insights from user opinions through the platforms.

This project aims to create a comprehensive full-stack review website with a primary objective of leveraging machine learning (ML) and natural language processing (NLP) techniques to empower game developers by eliminating the need for labor-intensive manual analysis and allowing game developers to make data-driven decisions. The project aims to revolutionize the review process by implementing advanced data processing and visualization capabilities, automating the analysis of reviews, and generating insightful feedback for developers. By leveraging the power of NLP, the website seeks to extract valuable information from user reviews, identify patterns and trends, and provide developers with detailed feedback that can improve their software development practices through easy-to-understand visualization. Developers can gain deeper insights and actionable intelligence from the reviews, enabling them to enhance their game and optimize the overall experience. This project represents a significant step towards empowering game developers through cutting-edge technologies and facilitating continuous improvement in their game development processes.

1.2. Objectives

The five main objectives of this project are listed below.

- Research into NLP, including tokenization, stemming, and stopwords removal.
- Provide feedback using Sentiment Analysis, Topic Modelling, and Keyword Extraction.
- Perform web-scraping for data extraction, processing, and model training.
- Develop a full-stack scalable modern web application with responsive web design approach.
- Provide intuitive data visualization to users in the web application.

1.3. Deliverables

This project aims to deliver a full-stack cloud-native web application, including a frontend webpage, backend system, and database and distributed ML models that support the NLP functionalities of the application.

1.4. Contributions

This project plan to develop a complimentary game review platform call CritiQ, which offers services such as automated review analysis, generation of aggregated review summaries, and analytical features pertaining to game reviews such as review length and sentiment distribution. Our platform serves as a significant contributor to the gaming industry by streamlining their game development process and facilitating efficient feedback loop between game developers and players. Furthermore, it provides substantial benefits to the general gaming community as it enables them to utilize our platform for game discovery, expressing their viewpoints about various games, and collectively enhancing the gaming experience through constructive feedback.

1.5. Outline

This report will present a comprehensive analysis of the project, focusing on key sections, including Related Work (Section 2), Detailed Methodologies (Section 3), Results (Section 4), Difficulties and Limitation (Section 5), Full Project schedule (Section 6), and a Conclusive Summary with Future Works (Section 7).

2. Related Work

Game reviews are user-generated posts that provide feedback, opinions, and discussions about specific games. They focus on evaluating the gameplay, graphics, and overall experience of the game. To better understand the characteristics of game reviews, related literature was reviewed. Lin et al (2019) conducted a thorough empirical study of game reviews to analyze their length, topics and relationship with players' playtime when writing the reviews. Comparisons were made between positive and negative reviews, early-access reviews and non-early-access reviews, indie games reviews and triple-A games reviews. Guzsvinecz & Szűcs (2023) analyzed the length and distribution of sentiments in over 35 million game reviews from 11 popular genres, such as Action, Racing and Sports.

However, special characteristics of the text in game reviews pose significant difficulty in the task of sentiment classification, in which the performance of the classifiers may be diminished. Vigiato et al (2022) identified six major characteristics. First, game reviews often come with frequent uses of contrast conjunctions as both advantages and disadvantages of the game are pointed out. Second, game reviews include words that is usually viewed negatively in other context, yet in a neutral or positive manner in game reviews, such as, "kill", "zombie", and "fire" in First Person Shooters and Action games. Third, sarcasm was frequently observed in game reviews. It occurs when a positive text was used to convey a negative attitude, or vice versa. The remaining three characteristics are unclearness, game comparison with another game or with a previous version of itself, and mismatched recommendation.

Sentiment analysis is one of the numerous aspects of NLP that aims to extract sentiments and opinions from texts (Birjali et al., 2021). It has been well applied in various domains, such as analyzing customers' product reviews, establishing a reliable recommendation system based on reviews, and public healthcare monitoring (Birjali et al., 2021). As for the gaming industry, various researchers attempted to classify the sentiment of comments using various machine learning models and deep neural networks.

Tam et al (2021) conducted a comparative study about the ability of machine learning models to classify around 15K game comments and reviews scrapped from Steam and Metacritic. Three-class sentiment labels, i.e., Positive, Negative and Neutral, were first created with pre-trained sentiment analysis models, such as VADER, on reviews. The Synthetic Minority Oversampling Technique (SMOTE) was then applied to create a balanced dataset, tackling data

imbalance in neutral and negative reviews over positive reviews. Five machine learning algorithms, which are Logistic Regression (LR), Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM), Multi-layer Perceptron Classifier (MLP), and Extreme Gradient Boosting Classifier (XGBoost) were then applied to build sentiment classifiers with both imbalanced dataset and balanced dataset. Results suggested that training with a balanced dataset with oversampling significantly improved the performance of most models.

Ruseti et al (2020) conducted a study about three-class sentiment classification on a 117K dataset with games reviews from Metacritic. Reviews were first classified into positive, neutral, or negative based on the score on Metacritic. Then a range of ML models, which were SVM, MNB, and deep neural networks (DNN), combined with bag-of-words (BoW), word2vec embedding, or Universal Sentence Encoder, were trained on a balanced dataset to classify the reviews. All models achieved accuracy between 61% and 67%.

Al Mursyidy Fadhlurrahman et al (2023) applied Bidirectional Encoder Representations from Transformers (BERT), Bi-directional Long-short Term Memory (BiLSTM), and Bi-directional Gated Recurrent Unit (BiGRU) on two class sentiment classification with a balanced dataset containing 7K comments from 10 most reviewed games on Steam. They then presented another model, BERT-BiLSTM-CRF, to enhance sentiment classification over fine-tuned BERT on the dataset. The original fine-tuned BERT model achieved 0.88 in F1 score, accuracy, and recall.

The above-mentioned related works provides expected sentiment classification rate in the context of game comments and reviews. However, works that compare the classification performance of game comments with different architectures on a large training and testing dataset are rare. As a result, our research in sentiment classification contributes to the aforementioned works as an experimental expansion by comparing the real-life performance of various approaches of feature extraction and model architectures.

Topic modeling is one of the numerous aspects of NLP that compresses a set of documents and return a set of highly representative topics that describe the content in an accurate and coherent manner (Churchill & Singh, 2022). Due to the nature of the length of game reviews, and rareness of applying topic modeling on game reviews, literature reviews regarding short text topic modeling on game reviews and social media posts were conducted.

Yu et al (2022) applied Latent Dirichlet Allocation (LDA) to explore the prominent topics in reviews from Dark Soul 3 and Dark Soul 1 separately, which the former is a sequel of latter, and compare the common topics within the two games. 14 and 15 topics were uncovered respectively from a total of 130K English reviews from Steam. Topics uncovered reflected players enjoyed the combat, character, overall experience, and difficulty of both games, and other commonly found aspects, such as graphic and gameplay.

Similar approach was also taken in (Stepien, 2021) to analyze the topic in 3 popular games: DOTA2, PUBG and GTA5. LDA and LDA Sequential were applied to uncover the distribution of changes of shares of topics by training both models on reviews scrapped from Steam of each game. The result suggested that changes in shares of topics were observed when major game updates were released, or major tournaments were organized.

More advanced topic modelling models have been applied to analyze different short text in social media posts. Eagger & Yu (2022) compared the ability of four common topic modeling techniques, LDA, Non-negative Matrix Factorization (NMF), Top2Vec, and BERTopic, in analyzing the topics in Twitter posts regarding travel and COVID-19 pandemic. 31800 unique tweets were collected from Twitter, and comparisons were drawn between the created topics and their keywords of LDA and NMF, and that of Top2Vec and BERTopic. Result suggested BERTopic and NMF were effective in analyzing Twitter data.

A similar comparison of topic models was conducted by Gan et al (2024) to compare three topic modeling methods: LDA, Top2Vec and BERTopic, in analyzing the latent topic in Twitter and Weibo posts regarding the topic ChatGPT. Result suggested generated topics from BERTopic were better segmented, more independent, with clear semantics understanding in both English and Chinese.

The above-mentioned works provide a baseline regarding the ability of various topic modelling techniques. Yet, works related to topic modeling on game reviews only focused on training separate models on each game. Works that compare the performance of applying a single topic modeling model across multiple games were hardly found. As a result, our research in topic modelling contributes to the works as an experimental approach of applying a single trained topic model to analyze various common aspects of games in different games.

3. Methodology

The project can be divided into three main sections which are Machine Learning and Natural Language Processing (Section 3.1), Frontend Web Applications (Section 3.2), and Backend Technologies (Section 3.3). The methodology of development will be discussed in detail.

3.1 Machine Learning and Natural Language Processing

This section discusses three NLP tasks which are Sentiment Analysis (Section 3.1.1), Topic Modeling (Section 3.1.2) and Keyword Extraction (Section 3.1.3).

3.1.1 Sentiment Analysis

This section presents the Problem Definition (Section 3.1.1.1), Data Preparation (Section 3.1.1.2), Text Preprocessing (Section 3.1.1.3), Exploratory Data Analysis (EDA) (Section 3.1.1.4), Dataset Preparation (Section 3.1.1.5), Feature Extraction and Model Selection (Section 3.1.1.6), Model Implementation (Section 3.1.1.7), Model Training (Section 3.1.1.8), Model Evaluation (Section 3.1.1.9), and Model Deployment (Section 3.1.1.10).

3.1.1.1 Problem Definition

Sentiment analysis is a crucial task for both game developers and potential players. It enables the former to understand the feedback and preferences of the gaming community, and the latter to form a clear and unbiased impression of the game's expected experience. This can facilitate better decision-making for both parties, such as improving game quality, features, and bug fixes, and making informed purchase choices. However, manually reading and analysing all the comments and reviews about a game is impractical and inefficient, especially for popular titles that generate a large volume of text data. Therefore, automated sentiment analysis can provide a useful and convenient way to obtain a summary of the community's opinions and sentiments toward a game.

The process of sentiment analysis typically consists of six stages (See Figure 1 (a)). First, text data is collected from relevant sources or platforms, and structured and stored in a suitable format, forming a dataset. Second, data preprocessing is applied to the text dataset to remove noise, irrelevant information, and reduce data dimensionality. Third, feature extraction is performed on the preprocessed text data to create a feature space that can be used by machine learning or deep learning models. Fourth, a model is implemented and trained on the extracted features to

learn how to classify the sentiment polarity of the text data, such as positive, neutral, or negative. Fifth, evaluation is conducted on separate testing or validation datasets to measure the performance of the trained model and select the best one. Sixth, the selected model is deployed to real-world scenarios on a system.

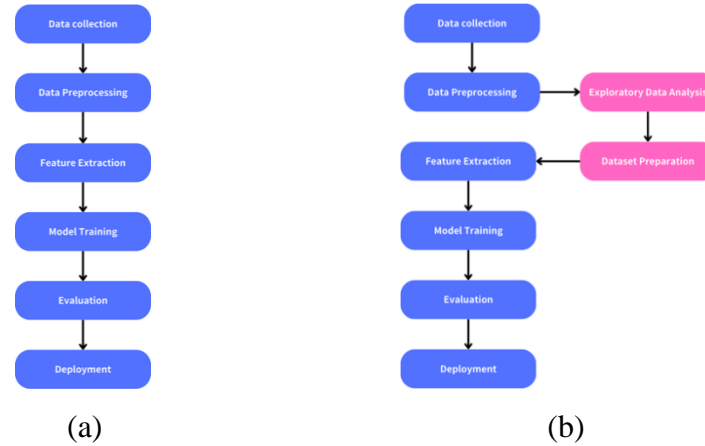


Figure 1: Usual stages in Sentiment Classification. (a): Six usual stages in Sentiment Classification. (b): Overall framework of section Sentiment Analysis of the project. Two additional stages were added and labeled in pink.

As mentioned in Section 2, Related Works, it is not uncommon to find a huge imbalance in several positive and negative reviews in a game. Also, research from 2015 showed that the performance of ML models is positively correlated with the size of the training dataset in sentiment classification (Prusa et al., 2015). Therefore, three research questions were created to guide the process of selecting the best performant model in sentiment classification of game reviews. These questions are in the following.

RQ1: Does an imbalance training dataset hamper model performance?

RQ2: What is the relationship between dataset size and performance?

RQ3: What is the best model with little hyperparameter selection?

Due to the research questions, compared to the common stages of sentiment classification, two stages, EDA, and Dataset Preparations, were performed after Data Pre-processing and before Feature Extraction to understand the distribution of data and prepare corresponding datasets for RQ1 and RQ2. Hence, the framework of this section involves eight stages (See Figure 1 (b)).

3.1.1.2 Data Preparation

An existing dataset created by Sobkowicz (2017) with reviews scrapped from Steam was selected. The dataset contains over 6.4 million publicly available English reviews from different games and genres on Steam. The dataset contains five columns, which are: ['app_id', 'app_name', 'review_text', 'review_score',

‘review_votes’], with each representing the id of the game on Steam, the name of the game, the review text, an indicator whether the review recommends the game, and an indicator whether the review was recommended by another user respectively. The sentiment of each comment was labelled by the column ‘review_score’, whether a ‘1’ indicates a positive review, and a ‘-1’ indicates a negative review, as there are two distinct values in the column ‘review_score’, which was either ‘1’ or ‘-1’. All ‘-1’ labellings in column ‘review_score’ were converted to ‘0’ for the convenience of model training, as typically, labels for classes began from ‘0’.

Selecting an existing dataset with a large number of comments saves a significant amount of time in data collection from creating a scraping program and running the data scraping program, speeding up the development process.

3.1.1.3 Text Preprocessing

Data cleaning is first performed on the cleaned dataset. First, rows with empty values in columns ‘app_name’ or ‘review_text’ were removed. Then, unhelpful comments containing merely the phrase ‘Early Access Review’ were removed. Next, rows with reviews containing filtered content were removed, as Steam replaced sensitive words with the symbol ‘♥’, leading to incomplete comments. Figure 2 displays an example of Steam's automated filtering. Rows with comments containing merely whitespaces were moved also. Finally, rows with less than 20 characters were removed to remove comments that contained too short, difficult-to-interpret content. After performing the mentioned data cleaning procedure, the number of rows in the dataset was reduced to 3.95M, a 38.4% reduction in terms of size.

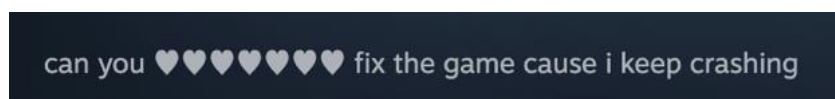


Figure 2: Steam automated comment filtering. The sensitive word was replaced by consecutive heart symbols.

3.1.1.4 Exploratory Data Analysis

Exploratory data analysis (EDA) was performed on the preprocessed dataset before model training. EDA is concerned with finding information from raw data and generating informal conclusions about the data. It aims to think about the data from various points of view by utilizing an array of tools, such as analyzing statistical values and graph plotting Fields(Morgenthaler, 2009). First, the ratio between the number of positive reviews and negative reviews was calculated to be 5.14: 1 (See

Figure 3), which was not surprising as similar ratios were recorded in a previous study across games in multiple genres (Guzsvinecz & Szűcs, 2023).

Next, the focus was shifted to analyzing the distribution of the number of words in reviews. After calculation with Python and Pandas, the medium number of words was 29, with a median number of 154 characters. It was suggested that our dataset contained comments with relatively shorter game reviews than a previous study, as the median number of characters was 25% smaller than a previous study of game reviews (Lin et al., 2019). Also, 99% of reviews were 549 words or less, suggesting handling the majority of reviews will not be a computationally intensive task in terms of the length of a review. Further investigation regarding the distribution of the number of words in positive and negative sentiment reviews was conducted. It was discovered that negative reviews were longer than positive reviews, as the former had a median number of words equal to 40, while the latter was 27. The result also echoed the findings in the same previous study (Lin et al., 2019).

Then, the focus was shifted to analyzing the common words of the reviews. Before analyzing, a further cleaning was performed, in which the flowchart of the process was shown below (See Figure 4). We first removed any hyperlinks and special markups (like “>”, “"”, “<p>”), in the comments. Then we removed any emojis in the sentences. Next, we convert all letters to lowercase and unify consecutive whitespaces to a single whitespace character. Then we remove any punctuation except “,”, “.” and “!”. Finally, we performed stopword removal and stemming using NLTK for each review. Stopwords are a set of words that are ubiquitous yet carry little meaning to the text, such as ‘a’, ‘I’, ‘do’, ‘be’, ‘then’, ‘that’, and ‘so’. Removing them can reduce the noise in the text. Stemming is a technique to reduce words in multiple-word forms to their base form. Applying the technique can reduce the feature space of words, avoid redundancy, and lead to a more consistent representation of the text. A. After that, the number of appearances of each word was calculated and the top 20 common words in the whole dataset were discovered. It was shown that the words were about the game, such as the word ‘game’, ‘play’, ‘one’, and ‘story’, and feeling towards the game, such as ‘like’, ‘good’, ‘fun’, ‘love’ (See Figure 5 (a)). If analyzing the sets with only positive or negative comments, 15 out of 20 frequent words in either positive or negative comments were in common (See Figure 5 (d)), showing huge overlapping in the set

of most common words in the set of positive-only reviews and negative-only reviews.

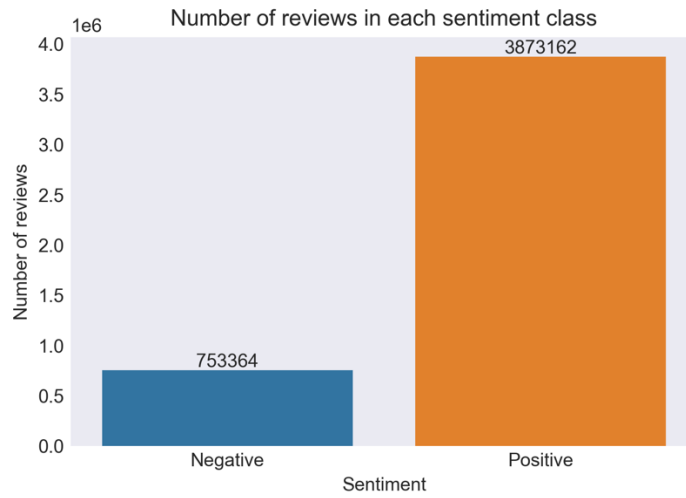


Figure 3: Number of reviews in each sentiment class in the cleaned dataset with a 5.14: 1 positive to negative ratio.

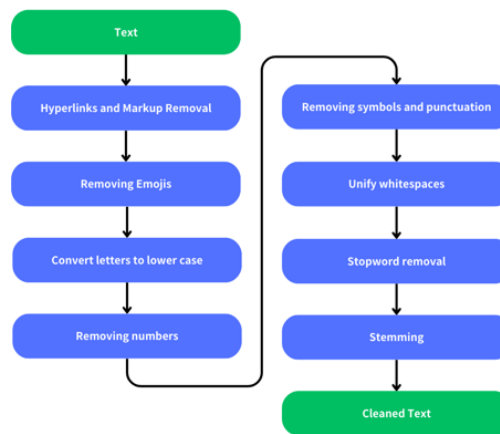


Figure 4: Procedure of further data cleaning on the cleaned dataset.

Common_words	count
0	game 8323674
1	play 2436824
2	like 1684616
3	get 1409738
4	one 1190300
5	good 1153401
6	time 1119516
7	fun 1115395
8	great 986062
9	realli 969311
10	stori 840158
11	make 835193
12	would 760233
13	even 695935
14	much 639383
15	well 591188
16	go 585183
17	love 578859
18	best 578752
19	feel 561540

(a)

Common_words	count
0	game 6717449
1	play 2003455
2	like 1348067
3	get 1095421
4	one 989361
5	fun 984819
6	good 978059
7	great 903207
8	time 882286
9	realli 779545
10	stori 728484
11	make 662531
12	would 600507
13	best 544695
14	love 525180
15	much 508792
16	well 507692
17	even 489625
18	go 450627
19	feel 441335

(b)

Common_words	count
0	game 1606225
1	play 433369
2	like 336549
3	get 314317
4	time 237230
5	even 206310
6	one 200939
7	realli 189766
8	good 175342
9	make 172662
10	would 159726
11	go 134556
12	buy 132174
13	much 130591
14	fun 130576
15	want 129230
16	tri 124038
17	feel 120205
18	bad 119376
19	thing 115886

(c)

Common_words	count_x	count_y
0	game 6717449	1606225
1	play 2003455	433369
2	like 1348067	336549
3	get 1095421	314317
4	one 989361	200939
5	fun 984819	130576
6	good 978059	175342
7	time 882286	237230
8	realli 779545	189766
9	make 662531	172662
10	would 600507	159726
11	much 508792	130591
12	even 489625	206310
13	go 450627	134556
14	feel 441335	120205

(d)

Figure 5: Number of appearances of top 20 frequent words in the cleaned dataset after further data cleaning. (a): Words in the cleaned dataset with both positive and negative reviews. (b): Words in the cleaned dataset with only positive reviews. (c): Words in the cleaned dataset with only negative reviews. (d): Common words in (b) and (c).

3.1.1.5 Datasets Preparation

Eight datasets were constructed from the cleaned dataset after Text Preprocessing. Two of them were for validation, and the remaining were for model training.

Regarding validation datasets, a balanced dataset and an imbalanced dataset were created to effectively perform cross-model performance comparison and evaluate the performance of models in real-life situations. The balanced dataset contained 268588 reviews with the same number of positive and negative reviews, while the imbalanced dataset contained 790655 reviews with the ratio of number of positive and negative reviews approximated to 4.90: 1, similar to the cleaned dataset. The creation process was as below. First, the imbalanced dataset was created by selecting 20% of the reviews randomly. These selected reviews were then dropped from the cleaned dataset. Next, the balanced dataset was created by first selecting 20 percent of the total number of negative reviews in the cleaned dataset, and then randomly selecting the same number of positive reviews from the cleaned dataset. Same as before, these selected reviews were dropped before the creation of training datasets.

Regarding training datasets, six training datasets were created to compare the performance of different models trained with different training dataset sizes and class distribution. Three different sizes were selected, which were 120K, 240K, and 480K. Then, for each size, a balanced and an imbalanced dataset was created. The ratio between the number of positive and negative was exactly 5: 1 in the imbalanced datasets. First, a balanced 120K dataset and an imbalanced 120K were created by random sampling without replacement. Then, each subsequent datasets in balanced and imbalanced categories were treated by doubling the number of training instances. The approach was analogous to the procedure in (Prusa et al., 2015). It was mentioned that with each smaller dataset being a subset of a larger dataset, a more meaningful comparison between performance and dataset size can be achieved, as any change in performance is the result of additional reviews instead of randomly selecting a completely new set of reviews (Prusa et al., 2015).

3.1.1.6 Feature Extraction and Model Selection

Feature extraction is a fundamental and indispensable task in sentiment classification as it directly influences performance (Birjali et al., 2021). It extracts valuable information that describes the characteristics of the text from the words

(Birjali et al., 2021). Birjali et al (2021) identified two major representations of features, which were Bag-of-Words (BoW), and Distributed Representation (also called Word Embedding). BoW first creates a vocabulary of all unique words occurring in the document, then encodes a sentence as a vector with the length of the vocabulary of known words. The value of each position in the vector represents a count or frequency of the word in the vocabulary. However, BoW is incapable of representing the syntactic information of the text as it does not consider word order, sentence structure, or grammatical construction. (Birjali et al., 2021). Distributed Representation distributes the information of a word in a vector space with a fixed dimension where each word can be represented by a vector. Relation between the semantic meaning of words can be represented with vector operation. A famous example is that the result of $vector("King") - vector("Man") + vector("Woman")$ is closest to the vector representation of the word "Queen" (Mikolov et al., 2013). Considering the difference in feature extraction, three methods of feature extraction and the corresponding model were selected to compare the performance of different feature extraction and representation methods in sentiment classification on game reviews.

The first model, TFIDF-RF, applied Term Frequency-Inverse Document Frequency (TF-IDF) and Random Forest Classifier. TF-IDF is an example of BoW feature extraction. It measures the importance of a word to a corpus by considering its frequency in the document, which is a review, and its rarity in the whole corpus, which is all reviews in the training dataset. TF-IDF value can be calculated by first calculating the Term Frequency (TF) of term t within a document d , where $f_{t,d}$ represents the frequency of the term t in document d (Eq. 1). Then the Inverse Document Frequency (IDF) of term t within the whole corpus D was calculated, in which the numerator represents total number of documents, and the denominator represents the document frequency of term t (Eq. 2). An extra '1' in both numerator and denominator equation (2) was to prevent zero divisions. Lastly, multiplying TF and IDF results in TF-IDF (Eq. 3). TF-IDF assigns higher weights to words that occur frequently in a document but are rare in the corpus, allowing models to prioritize terms that are more indicative of sentiment in each review. Random forest was selected because of its ensemble learning approach, as multiple decision trees are combined to make predictions.

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (1)$$

$$idf(t, D) = \log \frac{1 + |D|}{1 + |\{d \in D: t \in d\}|} \quad (2)$$

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (3)$$

The second model, GloVe-CNN, applied Global Vectors for Word Representation (GloVe) and Convolutional Neural Network (CNN). GloVe is an example of distributed representation. It captured the global corpus statistics directly by computing the word vectors based on the probability of the appearance of a target word given a context word, which is the co-occurrence probabilities of words. It first computed a large co-occurrence matrix based on the number of vocabs in the corpus, then optimized the word vector representation with least square regression with a custom loss function. Unlike TF-IDF, it encodes semantic relationships between words, which allows models to understand contextual information and capture sentiment nuances. CNN was selected because of its frequent use in sentence classification with ongoing advancements (Fachrul et al., 2022; Li et al., 2022; Maisa et al., 2023; Wenxuan & Yuxuan, 2022). It treats a list of word vectors from the sentence like an image with size (d, w) , where d = dimension of each word vector, w = length of a sentence with padding, if necessary. Then the filters learn the word features by adjusting the weights of each element in the kernel matrix.

The third model, BERT, was a fine-tuned model from pre-trained Bi-directional Encoder Representation from Transformer (BERT). Although embedding each word in the form of distributed representation, the model has some fundamental differences compared to the first two models. First, regarding the nature of the embedding vector, the embedding from BERT contains more information than previously mentioned distributed representations, such as word2vec and GloVe. In the second model where GloVe embedding was applied, a static vector was used to represent the target word regardless of the context words around itself. However, large language models, such as Generative Pre-trained Transformer (GPT), BERT, and ELMo, used contextualized embeddings, in which different embedding vectors were used for the same word in different contexts. Indeed, a recent review

confirmed that BERT token representations contain both syntactic and semantic information fields (Rogers et al., 2021). Second, regarding the training method, this model trained with a larger corpus with a larger number of words. Unlike the first two models which a model is trained directly and only from the corpus of the dataset of the task, this model adopted the approach of fine-tuning for a downstream task, such as text classification, translation, and question-answering, from a pre-trained language representation in a language model, such as GPT, and BERT. It allowed a larger model to learn a universal representation of words that can transfer to a wide range of tasks with little adaptation (Radford et al., 2018). Third, in the second model, given a target word represented by its word vector, it only considered a fixed window size of the context word due to the fixed kernel size in convolution operation in CNN. However, thanks to the self-attention mechanism in the Transformer, a target word can consider any word within the document and assign a different value. Last, BERT introduced bi-directional pretraining instead of using uni-directional pretraining in GPT. It enabled the ability of a given word to consider context words in both directions, which was believed to be more powerful than a uni-directional model or a shallow concatenation of left-to-right and right-to-left models (Devlin et al., 2019). For the conciseness of the report, the technical structure and computational details of BERT were omitted, to which the corresponding details can be referred (Devlin et al., 2019).

3.1.1.7 Model Implementation

Regarding the first model, a training pipeline was implemented containing three major components. First, the top N words with the highest TF score were selected to form the vocabulary. Next, the TF-IDF representation of these selected words was calculated. Each sentence was then transformed into a vector with each element representing the TF-IDF value of each word. Finally, the vectors were fed to train a Random Forest classifier as a group of bootstrapped classification trees. The first model was implemented using scikit-learn.

Regarding the second model, a training pipeline consisting of an embedding matrix and a CNN model similar to the structure in (Kim, 2014) was implemented. Before entering the pipeline, the top N words with the highest frequency across the whole training dataset were selected. Also, the length of a sentence was limited to a fixed

length L to prevent extended calculation time, and padding would be applied if necessary. In the pipeline, the embedding matrix was used to construct vector embedding of words in the sentence. The embedding matrix was set to be non-static, and the embedding vectors were updated during training to grasp the semantic meaning of words in the context of game reviews. It was reported that CNN models trained with non-static word vectors uniformly outperformed those trained with static word vectors (Zhang & Wallace, 2016). The CNN model consisted of three convolutional layers, placed in a flat structure. However, unlike convolutional layers in image-related tasks where a square kernel matrix will go through the whole image in both directions, the kernel will look into a window of a fixed number of words to produce a new feature. Then, each layer was followed by a 1D max pooling layer to select the most important feature in each channel of each convolutional layer. The features were then concatenated into a single-dimension array, and dropout was applied to the array as regularization. Finally, a fully connected layer is applied after the dropout layer to produce classification results. In implementation, instead of assigning a different filter size to each convolutional layer, we followed the recommendation of assigning all three convolutional layers with the same filter size (Zhang & Wallace, 2016), as it resulted in better performance than combining different sizes with the default setting. Original L2 weight regularization was omitted for simplified implementation. The resulting model is shown in the figure below (See Figure 6). The model was implemented using Keras, a high-level abstraction library for building deep learning models created by Google.

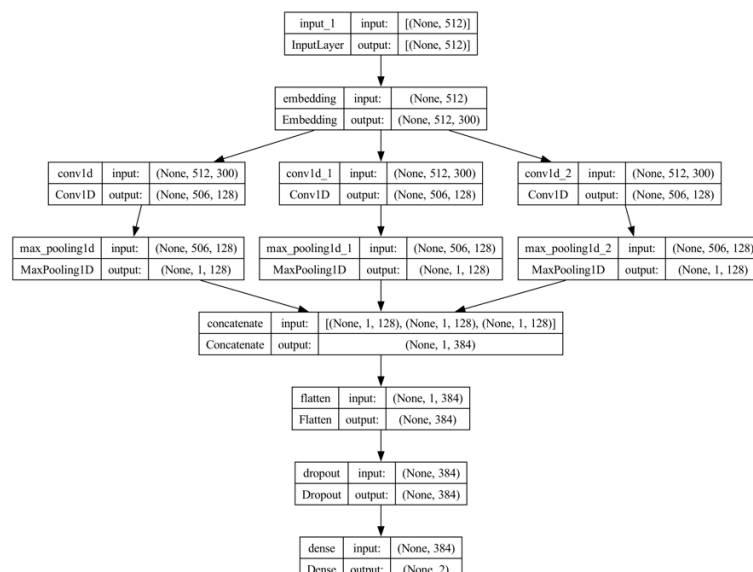


Figure 6: Model structure of CNN in model GloVe-CNN.

Regarding the third model, a training pipeline consisting of a tokenizer and a pre-trained BERT model was implemented. BERT_{BASE} was selected instead of BERT_{LARGE} as the former contained only about 1/3 of the parameters while achieving state-of-the-art results compared to other models (Devlin et al., 2019). The reduced parameters also significantly reduced the fine-tuning time required with a large training dataset on personal-scale hardware. Regarding the tokenizer, a pre-trained case sensitive tokenizer was selected, as capitalized words often contain more extreme emotions. Truncating and padding were applied to each review, as there is a maximum length of tokens the model can receive as input, which is 512. Regarding the model, a pre-trained BERT_{BASE} paired with the case sensitive tokenizer was initialized. The training pipeline was implemented with HuggingFace, a high-level abstraction library for building transformer models.

3.1.1.8 Model Training

Before training, further data cleaning customized with each model was performed on each training dataset. Applying customized further data cleaning for each model is necessary as it allows features to be consistent with the different feature extraction methods of each model, enhancing their performance. Flowcharts of three further data-cleaning processes are presented below (See Figure 7). Regarding the first model, the further data cleaning process following the same as that before analyzing the common words of the reviews was applied. The reasons behind applying the further data cleaning process were to reduce the feature space of words, avoid redundancy due to various verb forms, and lead to a more consistency representation of words in the corpus. Regarding the second model, the same further data cleaning process was applied to the training dataset except for stopword removal and stemming, as no stopword removal and stemming were performed during the pretraining GloVe. Regarding the third model, only the removal of emojis, hyperlinks, and markups was performed. Punctuations and numbers were retained as pre-defined tokens were available in the tokenizer of BERT. Also, punctuation was necessary for BERT to separate different sentences using special preserved tokens defined in the tokenizer. Both stemming and stopword removal were not applied as the tokenizer can break down words in various forms to one or more than one token which represented the stem and the verb form.

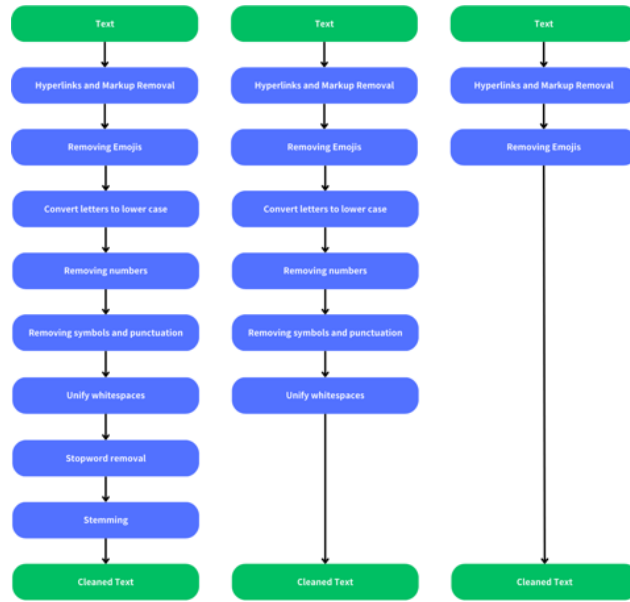


Figure 7: Customized further data cleaning to each model. Left: TFIDF-RF. Middle: GloVe-CNN. Right: BERT

For model training with all different training datasets, 10% of the training dataset will be used as a testing dataset during training to monitor the training process.

Regarding model parameters in the first model, N was set to 20K, which was chosen to represent non-arcane vocabs without posing significant computational difficulty in the reviews. The number of classification trees was 100. Other parameters of components used in the pipeline followed the default parameters in scikit-learn.

Regarding the model parameters in the second model, N was set to 20K, and L was set to 512, which was identical to the maximum length of tokens in BERT. A GloVe representation pre-trained on 6 billion tokens, with 300-dimensional word vectors, was used to initialize the word embedding layer. Each convolutional layer contains 128 filters, with filter size = 7. The dropout probability was set to 0.3. The batch size was set to 128 to fully utilize the memory of the GPU while presenting a more stable gradient. The loss was computed using sparse categorical cross-entropy. Adam optimizer was applied with learning rate = $1e-3$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. Other parameters of components used followed the default parameters in Keras. To prevent overfitting, the learning rate was reduced by 0.8 when there was no reduction in testing loss during training. Early stopping was performed when there was no reduction in testing loss for 5 consecutive epochs. After training, the best model with the lowest testing loss will be stored and used for evaluation.

Regarding the model parameters of the third model, a pre-trained BERT_{BASE} (Devlin et al., 2019) model was used for further fine-tuning. Following the recommendations of (Devlin et al., 2019), the batch size was set to 32. Adam

optimizer was used with learning rate = $2e-5$, as a higher learning rate will result in catastrophic forgetting shown by (Sun et al., 2019), $\beta_1 = 0.9$, $\beta_2 = 0.999$. L2 weight decay was set to 0.001. No learning rate warmup was applied as the number of steps in training with 120K datasets was fewer than 10000. The number of epochs was set to 3. Other parameters of components used followed the default parameters in HuggingFace. The best model with the lowest testing loss will be restored and saved for evaluation.

3.1.1.9 Model Evaluation

The evaluation was performed with the view to answering the research questions, and therefore, selecting the best performance model for deployment. Weighted average F1-score was chosen for the chief performance metric as the equation considered both recall and precision by calculating a harmonic mean of them. To define a weighted average F1-score from F1-score, we first let there be n sentiment classes and define y_i to be the number of samples within the sentiment class i . Then we calculate the precision and recall for each sentiment class i . Next, we calculate a weighted average of the precision (See Eq. 4) and recall (See Eq. 5) of all sentiment classes n . Finally, we calculate the weighted average F1-score by considering the weighted average of precision and recall (See Eq. 6).

$$WeightedAveragePrecision = \frac{\sum_{y=1}^n y_i \frac{TP_i}{TP_i + FP_i}}{\sum_{y=1}^n y_i} \quad (4)$$

$$WeightedAverageRecall = \frac{\sum_{y=1}^n y_i \frac{TP_i}{TP_i + FN_i}}{\sum_{y=1}^n y_i} \quad (5)$$

$$WeightedAverageF1 = \frac{2}{\frac{1}{WeightedAveragePrecision} + \frac{1}{WeightedAverageRecall}} \quad (6)$$

3.1.1.10 Model Deployment

Before deploying on the virtual machine, the trained model was converted to ONNX format for fast CPU inference with ONNXRuntime, as a previous study showed that a 36.5% reduction in inference execution time was recorded on ONNX + ONNXRuntime than Torch Script + Libtorch, a library for running PyTorch models

on C++ (Öğüt, 2021). Comparisons of end-to-end inference times of a prediction on CPU between the original and converted ONNX model were drawn on two different machines, one on a moderate window machine running WSL 2 and another on an Apple laptop. An end-to-end inference refers to the inference from a pre-processed review, then producing possibilities on both sentiment classes. WSL2 was used instead of running natively on Windows as TensorFlow, the backend of Keras, stopped complete support on native-Windows. The specifications of the two machines are listed below.

	Machine 1	Machine 2
CPU	Intel Core i5-8250U, 4 Cores, 8 Threads	Apple M1 Max, 8 Performance Cores, 2 Efficiency Cores
RAM	8GB (in WSL 2)	32GB
OS	Ubuntu 22.04 (in WSL 2) Windows 10 22H2	macOS Monterey 12.6.2

Python was used to run the measurement program as it was the programming language for both ML development and deployment. For the sake of completeness, the version information of used software is provided below.

- Python 3.9.18
- scikit-learn 1.3.0
- TensorFlow 2.15.0
- Keras 2.15.0
- torch (PyTorch) 2.1.0
- transformers (HuggingFace) 4.35.0
- accelerate (HuggingFace) 0.24.1
- onnx (ONNX) 1.14.1
- onnxruntime (ONNX Runtime) 1.16.3
- skl2onnx 1.16.0
- tf2onnx 1.15.1
- optimum (HuggingFace) 1.16.1

Regarding the conversion progress, an end-to-end ONNX model was generated from scikit-learn. However, the text vectorizer component in Keras and the tokenizer component in HuggingFace were unable to convert to ONNX as ONNX

does not support string manipulation. Therefore, original components from Keras and HuggingFace were used in the end-to-end ONNX inferencing.

For the measurement program, after loading the models and reviews, a warmup of inferencing 1000 reviews with batch size = 1 was performed. It ensures the model is fully loaded to the memory, reducing variance in inference time. Then, the time of inferencing 2000 reviews with batch size = 1 was measured. Batch size = 1 was selected as the sentiment classification result should be provided as soon as the review was posted to the platform, in which the message was consumed immediately by the Python NLP backend. The time required for performing data cleaning to the reviews was unrecorded as it is insignificant to the overall inference time. The times of inferencing each review were then stored for further analysis.

3.1.2 Topic Modelling

Topic Modelling is an unsupervised machine learning technique that aims to discover hidden themes in textual data and perform categorization (Churchill & Singh, 2022). Applying topic modeling for game reviews benefits both potential players and developers. To potential players, grouping reviews allows them to quickly glance at a specific aspect of a game, such as graphics, compatibility, and gameplay, without the need to read hundreds of fewer related comments. This shortens the purchasing decision-making progress. To developers, topic modeling helps them to better prioritize tasks to be done to cater to their players' needs. It distinguishes reviews based on various aspects of the game, such as graphics, gameplay, and bug reports, selecting contributing comments from unhelpful, emotional comments. This empowers developers to quickly locate issues that players complain about the most, for instance, game-blocking bugs and crashes (Lin et al., 2019), and reallocate manpower and time to provide a more satisfactory gaming experience to both current and future players.

Three topic modelling techniques were selected to apply to the task of topic modeling on game reviews. In particular, LDA, Contextualized Topic Model (CTM) (Bianchi et al., 2021), and BERTopic (Grootendorst, 2022) were selected. LDA was selected because of its generality and frequent usage in various topic modeling problems (Churchill & Singh, 2022). It serves as a baseline to compare existing results in examined previous studies. BERTopic and CTM were selected as contextualized embeddings was used for building topic models in both techniques, in which

contextualized embeddings produces the best performance among the three examined feature extraction methods in the task sentiment analysis.

LDA is a probabilistic, bag of words model that represents topic based on the probability of appearance of words in that topic. It assumed each word in the document is created from sampling a topic from the distribution of topics for the document, and then sampling a word from the topic (Abdelrazek et al., 2023). The algorithm aims at finding the topic-word distribution that maximizes the likelihood of documents in the dataset over K number of topics (Churchill & Singh, 2022). Two more parameters, alpha and beta, were used to define Dirichlet priors for drawing topic distribution and word distribution within a topic.

CTM is a bag of words model extended from neural topic model. A neural topic model is an encoder-decoder which first maps the bags-of-word (BoW) document representation to a continuous latent representation through encoder, then reconstructs the BoW by generating the words from the latent representation through decoder (Bianchi et al., 2021). An example is ProdLDA (Srivastava & Sutton, 2017), which improves from LDA by approximating the Dirichlet prior in LDA using Gaussian prior and replacing distribution of words with product of experts (Srivastava & Sutton, 2017). CTM extends ProdLDA by adding contextualized embedding generated from SBERT (Reimers & Gurevych, 2019).

Unlike the first two models which represent a topic by word distribution, BERTopic adopted a clustering embedding approach with four key steps. First, similar to the second model, contextual document embeddings are generated from SBERT. Then, dimensionality of embeddings is reduced using UMAP and clustering is performed using HDBSCAN. Next, a class-based TF-IDF (c-TFIDF) was used to model the importance of words in a cluster to generate topic-word distribution. Finally, topic merging was performed based on the c-TFIDF representation to reduce the number of topics to a specific value.

Data scrapping will be performed on the Steam platform to acquire a comprehensive dataset of all potential game genres. Subsequently, the review data will be categorized based on the genre of the game that the review pertains to. Then, it will undergo a

process of data cleansing and preprocessing to reduce noise and improve the efficiency of the training process. Utilizing this refined review data, we will train the topic models using the review datasets specific to each genre.

In relation to the datasets, EDA will be conducted subsequent to the data cleaning process. The reviews will be linked to the 'gameid' column, correlating them to the games they are critiquing. We will specifically select reviews that provide commentary on a specific genre of games. For instance, reviews that reference games are not present in the Steam game database will be excluded from consideration.

Moreover, Grid Search methodology is employed to discover the optimal quantity of topics for the topic models. All topic models will be trained with a range of topics spanning from 10 to 100, incrementing in steps of 10.

While conventional naming algorithms for topics generated by the model are predicated using keyword-extraction methodologies, they frequently fall short in providing human-comprehensible names for each topic. Given the challenges and laborious nature of topic naming via traditional methods, usage of LLM will be investigated as a potential alternative for this task.

In order to assess the performance of various topic models, both quantitative and qualitative approaches will be employed. From a quantitative perspective, measurements of topic coherence and topic diversity will be conducted using predefined metrics. More specifically, topic coherence will be evaluated using Normalized Pointwise Mutual Information (NPMI), which involves the identification of top words, calculation of pairwise NPMI, averaging of NPMI scores, and averaging across topics. In terms of topic diversity, it will be measured using Inverted Rank-Biased Overlap (RBO), which includes the processes of identifying top words, calculating pairwise RBO, inverting RBO, and averaging over topic pairs. Regarding qualitative approach, pyLDAvis will be utilized to qualitatively analyze the results of the LDA. A manual inspection will be performed on the top 10 keywords of each topic, and the representative texts for each topic will be evaluated to determine whether they are capable of extracting more meaningful representative documents. Similar

qualitative analytics tools are available for other topic models, which will not be discussed in the section.

3.1.3 Keyword Extraction

Keyword extraction is a technique to extract a set of keywords from a document without manual work (Khan et al., 2022). It can be applied to game reviews to extract critical information, especially for longer reviews, and to provide a high-level overview of the review content and sentiment. It can also act as a topic modeler to support the topic modeling tools by assigning interpretations to the topics categorized by the topic modeling models.

Despite the existence of various developed keyword extraction models, such as KeyBERT, YAKE, Text Rank, Page Rank, none of them produce interpretable and easy to read description of a topic after topic-modeling is applied. For instance, in Figure 8, KeyBERT generates short n-gram keywords that were created while Llama2 can summarize the topic-keywords into a single term. Therefore, our focus was shifted to using pre-trained Large Language Models (LLMs) to produce comprehensible descriptions of identified topics in section Topic Modeling.

Topic	Count	Name	CustomName	Representation	KeyBERT	Llama2	HR	Representative_Docs	
0	-1	34138	-1_the_of_and_to	Computer Vision and Machine Learning	[the, of, and, to, in, we, for, is, that, on]	[models, model, accuracy, datasets, networks, ...]	[Computer Vision and Machine Learning, ...]	[the, of, and, to, in, we, for, is, that, on]	[We propose a random convolutional neural ne...
1	0	10326	0_policy_reinforcement_rl_agent	Deep Reinforcement Learning Challenges	[policy, reinforcement, rl, agent, learning, c...]	[reinforcement, learning, robots, dynamics, ro...]	[Deep Reinforcement Learning Challenges, ...]	[policy, reinforcement, rl, agent, learning, c...]	[Efficient exploration is a crucial challenge ...]
2	1	3574	1_privacy_federated_fl_private	Privacy-preserving federated learning	[privacy, federated, fl, private, clients, dat...]	[federated, heterogeneity, decentralized, dist...]	[Privacy-preserving federated learning, ...]	[privacy, federated, fl, private, clients, dat...]	[Providing privacy protection has been one o...]
3	2	3523	2_speech_audio_speaker_music	Audio-Visual Speech Separation and Recognition	[speech, audio, speaker, music, asr, acoustic...]	[convolutional, encoder, voice, speech, traine...]	[Audio-Visual Speech Separation and Recognition...]	[speech, audio, speaker, music, asr, acoustic...]	[In recent years, neural network based metho...]
4	3	2275	3_equations_differential_physics_neural	Solving Partial Differential Equations using N...	[equations, differential, physics, neural, pde...]	[pdes, pde, modeling, dynamics, computational...]	[Solving Partial Differential Equations using ...]	[equations, differential, physics, neural, pde...]	[Physics-informed neural networks (PINNs) have...]
...	
117	116	160	116_augmentation_mixup_data_training	Data Augmentation Techniques in Deep Learning	[augmentation, mixup, data, training, augmenta...]	[adversarial, augmentation, imagenet, regulari...]	[Data Augmentation Techniques in Deep Learning...]	[augmentation, mixup, data, training, augmenta...]	[Data augmentation techniques have become st...]
118	117	160	117_crowdsourcing_workers_crowd_worker	Crowdsourced Data Labeling	[crowdsourcing, workers, crowd, worker, crowds...]	[crowdsourcing, crowdsourced, annotators, crow...]	[Crowdsourced Data Labeling, ...]	[crowdsourcing, workers, crowd, worker, crowds...]	[There is a rapidly increasing interest in c...]
119	118	155	118_summarization_summaries_summary_abstractive	Automated Document Summarization	[summarization, summaries, summary, abstractiv...]	[summarization, summarisation, summarizing, su...]	[Automated Document Summarization, ...]	[summarization, summaries, summary, abstractiv...]	[Abstractive Text Summarization is the proce...]
120	119	152	119_design_circuit_circuits_chip	Design Automation for Analog Circuits using Re...	[design, circuit, circuits, chip, synthesis, d...]	[circuits, circuit, analog, visi, optimization...]	[Design Automation for Analog Circuits using R...]	[design, circuit, circuits, chip, synthesis, d...]	[The design automation of analog circuits is...]
121	120	150	120_sentiment_aspect_analysis_polarity	Sentiment Analysis of Online User-Generated Co...	[sentiment, aspect, analysis, polarity, review...]	[embeddings, sentiment, sentiments, annotated...]	[Sentiment Analysis of Online User-Generated C...]	[sentiment, aspect, analysis, polarity, review...]	[Targeted Sentiment Analysis aims to extract s...]

Figure 8: Example output of BERTopic using KeyBERT and Llama2 to name the topics.

Instead of using existing online services like OpenAI, or Azure ChatGPT, the utilization of locally deployed LLMs offers two main advantages. Firstly, by opting for local deployment, the privacy of user data is ensured, mitigating the risk of any confidential or sensitive information being leaked. Secondly, local deployment allows for the

utilization of a wide range of models, including Llama2 from Meta, phi-2 from Microsoft, Mistral developed by Mistral AI, and even custom fine-trained models. This not only grants developers fine-grained control but also enables the use of "uncensored models" that are fine-tuned on datasets without filtered responses. This is particularly crucial in analyzing game reviews where the content may contain sexual, aggressive language that can trigger models content filtering mechanism. Examples of game reviews that triggered Azure ChatGPT content filtering mechanism are shown in Figure 9.

The image contains two side-by-side screenshots of game reviews and their corresponding ChatGPT responses. Both reviews have been filtered by Azure OpenAI's content management policy.

(a) SC2M - The eSports Visual Novel
 Review: My dream of the Jaedong and Flash threesome route is coming true 18/18
 Sentiment: Positive (Recommended)
 Response from ChatGPT: The response was filtered due to the prompt triggering Azure OpenAI's content management policy. Please modify your prompt and retry. To learn more about our content filtering policies please read our documentation: <https://go.microsoft.com/fwlink/?linkid=2198760>.
 Content filter result: safe

(b) Prison Architect
 Review: Treat all of my prisoners like gods, riot breaks out. Cry because I feel betrayed. MURDER ALL OF THEM. Wonder why my prison gets shut down. 18/18
 Sentiment: Positive (Recommended)
 Response from ChatGPT: The response was filtered due to the prompt triggering Azure OpenAI's content management policy. Please modify your prompt and retry. To learn more about our content filtering policies please read our documentation: <https://go.microsoft.com/fwlink/?linkid=2198760>.
 Content filter result: high

(a)

(b)

Figure 9: Two game reviews and the response from ChatGPT hosted by Azure when prompting to classify their sentiment.

Specially, three tools were selected to integrate LLMs into our system to perform the keyword extraction task, which are LangChain, a prominent framework for application development that integrate with LLMs, Mistral AI, a powerful and open source LLM developed by Mistral that offer excellent performance, and Chroma, a lightweight vector database that facilities Retrieval-Augmented Generation (RAG). With these tools, possible prompt techniques such as RAG and generated knowledge will be explored and instruct the LLM to perform the keyword extraction task.

The result of the keyword extraction will be evaluated quantitatively by manual inspection. In particular, the coherence between the generated name for the topic and some most representative documents of the topic will be examined.

3.2 Frontend Web Application

This section discusses the technologies that will be involve in developing the frontend system (Section 3.2.1), the user interface design approach and method for the web application (Section 3.2.2), and the proposed implementation of authentication process and user information access and management for the web application (Section 3.2.3).

3.2.1 Technologies Involved

First, the selected framework for developing the application is React, which is a widely adopted and popular web framework known for its extensive range of community-made packages, which adopt a declarative and component-based approach to build user interfaces.

Second, to enhance the appearance and functionality of the user interface, Material UI (MUI), a component library of React that provides a set of prebuilt and customizable UI components, will be used because it provides efficient, production-ready, and complete set of components that can facilitate the website development tremendously. For instance, it provides inputs components such as the text field, select and button components, which can be used to create different forms, such as the login form and the add review form.

Third, Next.js, a meta-framework built on top of React, has been chosen to provide support for modern features like Server-Side Rendering, File-based Routing, Secure Fetching, and Performance Optimization. These features enable the application to be maintainable, responsive, performant, and scalable.

Finally, TypeScript, a syntactic superset of JavaScript that offers high-level type safety, has been chosen as the programming language to enhance development efficiency and minimize unintended bugs caused by typing mismatch, as TypeScript ensures all variables only access authorized memory locations that are well-defined and permissible.

3.2.2 Design Approach

To cater to a broader audience and enhance accessibility for users across different devices and platforms, such as mobile, tablet, and desktop, the web application must be responsive and performant. Therefore, the web application will be designed under the Responsive Web Design (RWD) approach, which adjusts the size, position, and visibility of webpage elements based on the device viewport to ensure that the website will have a natural and intuitive appearance on different screen sizes, resolutions, and

orientations. This design approach ensures that the web application is compatible with various devices with different screen sizes. This will enable users to seamlessly interact with the platform regardless of their preferred device and platform, thereby expanding our reach and maximizing user engagement. To implement the RWD, our web application will utilize media queries and modify the style properties of Cascading Stylesheets (CSS) based on various breakpoints. We adhere to the breakpoints defined by MUI to represent different viewports. Specifically, a width of 0-600px is indicative of mobile devices, 900-1200px corresponds to tablet devices, and anything above 1200px is representative of desktop devices.

In addition, the design process of the web application is facilitated by Figma, a popular and collaborative design tool for application development. Figma offers valuable community assets for creating the web application prototype, such as the Material UI asset that contains all the prototype assets of the pre-built UI components provided by the Material UI library. In addition, Figma enables collaborative design features, which allow multiple people to join and participate in the design process as a team, making it appropriate for a group project.

Furthermore, to maintain the design consistency of our web application, we utilize the theming feature of MUI. By configuring the theme variables within the MUI theme provider, we ensure a consistent colour palette, typography, and breakpoints for (RWD across all pages.

3.2.3 Frontend Authentication

The frontend application's user authentication will be implemented using JSON Web Token (JWT) (See Section 3.3.2), HTTP cookie (browser cookies), and *useContext* hook from React. The user login authentication will invoke one of the two backend APIs, depending on the presence of the refresh token in the HTTP cookies.

The first API, login, takes the user credentials as the request body and returns the access token and refresh token in the response body. This API is invoked only when the HTTP cookie does not contain a valid refresh token. The users can access this API through the login form in the web application, where they must enter their username or email and password. The login form has a "Remember Me" checkbox. If the user selects this

option, the refresh token is stored as a persistent cookie with a 7-day expiration time. Otherwise, the refresh token is stored as a session cookie without an expiration time, and the cookie will expire if the user closes the browser session.

The second API, `refreshToken`, refreshes the session by generating a new access token. It takes the refresh token as the request body and returns a new valid access token in the response body. When the user accesses the web application, the frontend application will verify the existence of the refresh token in the HTTP cookies. If it exists, the application will invoke the API to refresh the session and obtain a new access token. The backend system will handle the invalid or expired refresh tokens by sending the appropriate error message in the response body.

Once the access token is obtained successfully by either method, it will invoke the `userAuth` API to fetch the updated user information. This API takes the access token as the request body and returns the user information in the response body if the access token is valid.

To manage and access the user information globally for the react application, the `useContext` hook will be employed. The user information will be stored in a mutable state using the `useState` hook. A context provider will be created with the user information state as a value. The page components will be wrapped within the provider to allow the provider to pass the user information state value to the pages. The page components can access the user information through the `useContext` hook via the context provider.

3.3 Backend Technologies

This section discusses the implementation of 9 services supporting the Backend solutions, including Spring Boot Server Application (Section 3.3.1), Authentication with JWT (Section 3.3.2), Email Service (Section 3.3.3), Database (Section 3.3.4), Object Storage (Section 3.3.5), Continuous Integration/Continuous Delivery (Section 3.3.6), Message Queue (Section 3.3.7), Hosting (Section 3.3.8) and Monitoring (Section 3.3.9).

3.3.1 Spring Boot Server Application

The backend system will be built using the Java Spring Framework, a common framework used to build high-performance and scalable Enterprise Application Programming Interface (API) solutions.

Spring Boot provides most of the functionality needed for a scalable backend API system, including security, MVC (Model View Controller), Batch Processing, High Performance, and non-blocking event loops. All data access requests will be made to the backend system through RESTful Hypertext Transfer Protocol (HTTP) requests. Most business logic and validation will only be performed in the backend system to provide security and high performance through parallelization on clusters as it has linear scalability.

By centralizing these processes in the backend system, parallelization can be leveraged on clusters for improved performance. Furthermore, this approach allows for linear scalability, ensuring that the system can efficiently handle an increase in demand.

External libraries that are used in the application will be installed with Apache Maven, an open-source tool for building and managing any Java-based project. The deployed server will use Maven to generate an executable JAR file using the production environment.

3.3.2 Authentication with JWT

The backend system implements the authentication mechanism based on the JSON Web Token (JWT) RFC standard, using Spring Security as the framework for authentication and access control. The system stores the tokens in both the client-side browser and the server-side database for security and convenience. When a user authenticates successfully, the system returns a JWT response to the client, which contains the user information and two tokens: the Access Token and the Refresh Token. The client utilizes these tokens to obtain authorization for API calls and to refresh the session when needed. The system ensures the security and validity of the authentication by applying a digital signature to the token data.

The system adopts HS256 as the encryption algorithm for the JWT signature, which is a symmetrical algorithm that relies on a shared secret key between the identity provider (Spring Security) and the application user. HS256 offers an adequate level of security and high performance for the system, as the system is the sole consumer of the JWT. An alternative algorithm, RS256, is an asymmetrical algorithm that uses a public and private key pair to generate and verify the JWT signature. RS256 is more suitable for scenarios where the client is not controlled by a single platform or application, as the client only needs to know the public key.

The system only includes non-sensitive user information in the Access Token, which can be decoded and viewed by using a public JWT reader (See Figure 10). The system sends the user's ID, email, and name as the user claims, along with the "exp" (Expired At) and "iat" (Issued At) claims, which indicate the expiration and issuance time of the token. The client can use these claims to determine when to renew the token or to prompt the user to log in again. Information can be added to the claims easily to support future

development by modifying the HashMap used to generate the Token without affecting Authentication.

```
{
  "email" : "u3578552@connect.hku.hk",
  "id" : 39,
  "exp" : 1703480516,
  "name" : "JackyLee997",
  "sub" : "u3578552@connect.hku.hk",
  "iat" : 1703394116
}
```

Figure 10 JWT Claims extracted from the Access Token user received on login

3.3.3 Email Service

Email service will be set up to support User authentication services, including email verification and forgot password. We will utilize the Gmail SMTP Server to send the email from the Google account created. Utilizing the Google SMTP Server instead of a separate Mail server reduces the workload of our virtual machine and improves the efficiency of the User registration and authentication workflow.

All emails will be sent from Spring Boot using the Spring Email library to help establish the connection to the SMTP server and provide detailed results and tracking of the email sent without additional setup and configuration.

Gmail offers free email service with the additional advantage of reducing the email being flagged as a Spam email or filtered compared to using an email attached to a custom domain.

3.3.4 Database

MySQL will be adopted as the preferred database due to its ability to handle relational data effectively and deliver high performance. Digital Ocean, a reputable cloud service provider, has been chosen to host the database.

To streamline database access and optimize Create, Read, Update, and Delete (CRUD) operations, Java offers the Java Persistence API (JPA). By leveraging the Spring JPA library, the application can benefit from improved performance and reduced boilerplate code for database interactions. JPA also encompasses features that support the Atomicity, Consistency, Isolation, and Durability (ACID) model, thereby ensuring data integrity and consistency when deployed as a distributed system. To ensure a structured and organized approach to database management, the creation and management of database tables and entities will be handled exclusively by Spring JPA. This approach guarantees the maintenance of data integrity and consistency throughout the lifecycle of the application. The entity relation diagram encapsulates the data necessary for the platform (See Figure 11), such as game, reviews, and user data and the attributes present in each entity. It also shows the relation and cardinality between entities and their representation in a database schema.

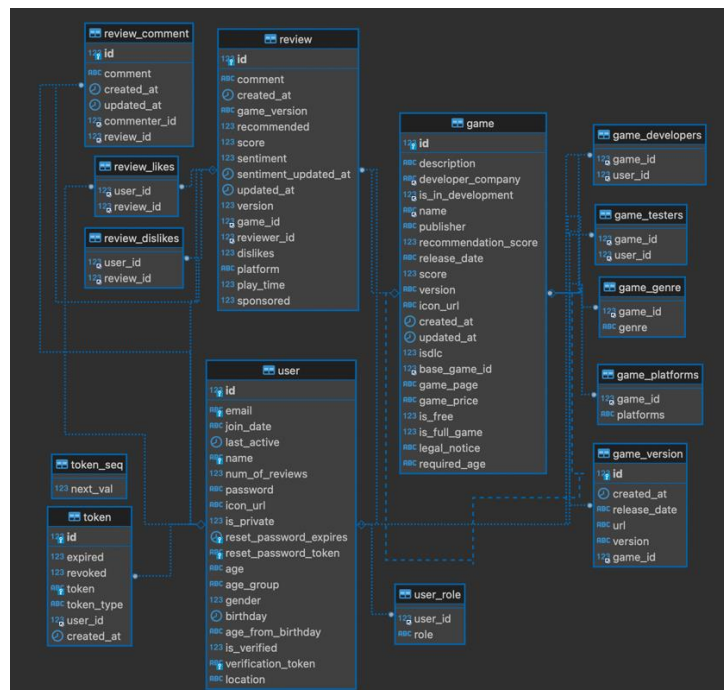


Figure 11 Database Entity Relations Diagram

Sensitive User data including Passwords and tokens will be encrypted before being stored in the database to ensure data security and will not be serialized and sent to users.

3.3.5 Object Storage

The project will incorporate a Simple Storage Service (S3) compatible storage solution to house all files provided by Digital Ocean. This includes but is not limited to, text documents, images, videos, and audio files. By opting for an S3-compatible storage bucket, the application can leverage the capabilities of the Amazon Web Service (AWS) Software Development Kit (SDK) to securely access, upload, and remove files from the bucket (See Figure 12).

```
public String uploadFile(final String fileName, final MultipartFile file) {
    try {
        ObjectMetadata metadata = new ObjectMetadata();
        metadata.setContentLength(file.getSize());
        metadata.setContentType(contentType(file));
        metadata.setHeader("x-amz-acl", "public-read"); // publicly accessible, comment this to not publicly accessible
        PutObjectResult result = amazonS3Client.putObject(bucketName, fileName, file.getInputStream(), metadata);

        System.out.println("Content - Length in KB : " + result.getMetadata().getContentLength());

        return result.getETag();
    } catch (IOException ioe) {
        logger.error("IOException: " + ioe.getMessage());
    } catch (AmazonServiceException serviceException) {
        logger.info("AmazonServiceException: " + serviceException.getMessage());
        throw serviceException;
    } catch (AmazonClientException clientException) {
        logger.info("AmazonClientException Message: " + clientException.getMessage());
        throw clientException;
    }
    return null;
}
```

Figure 12 Sample Code to upload a file to the S3 Bucket

The integration of an S3-compatible storage service offers a multitude of advantages. Primarily, it ensures high availability, thereby facilitating continuous access to the stored files. This is critical to guarantee uninterrupted service to the users. Furthermore, it provides high scalability, effectively accommodating the potential growth of the application and its associated file storage needs. This scalability ensures that the application remains future-proof and can handle increased demand efficiently. Lastly, the use of such a service guarantees high-performance file access and upload capabilities. This enhances the efficiency of file-related operations, thereby improving the overall user experience. **The dashboard** provided by Digital Ocean allows for clear analysis, modifications, and an overview of the files stored in the bucket (See Figure 13).

fyp / review / 33 4 items

Search review/33/ Create Folder Upload

<input type="checkbox"/>	Name	Size	Last modified	
<input type="checkbox"/>	0.jpg	26.02 KB	6 days ago	...
<input type="checkbox"/>	1.jpg	74.08 KB	6 days ago	...
<input type="checkbox"/>	2.jpg	36.93 KB	6 days ago	...
<input type="checkbox"/>	3.jpg	47.41 KB	6 days ago	...
<input type="checkbox"/>	4.jpg	51.33 KB	6 days ago	...

Figure 13 Digital Ocean Spaces Dashboard

3.3.6 Continuous Integration/Continuous Delivery (CI/CD)

To facilitate the development and deployment of our backend systems and minimize downtime during cutover, we have set up a Custom CI/CD pipeline using Jenkins, an open-source software for automating deployment using pipelines hosted on Digital Ocean's Ubuntu Virtual Machine.

Jenkins will poll GitHub, the Source Change Management platform used for this project every minute to query for changes and commits made to the repository. The pipeline written will deploy the backend and NLP solution when changes have been made to their respective directories. When such changes are detected, Jenkins will trigger their respective build stage to build the Docker Container using a Dockerfile, a file command to specify the building steps of the system and deploy the new changes (See Figure 14).

```

1 pipeline {
2   agent any
3   tools{
4     maven 'maven_3.9.5'
5   }
6   stages{
7     stage('Build Maven'){
8       steps{
9         checkout([$class: 'GitSCM', branches: [[name: '**/main']], extensions: [[$class: 'GitLFSPull']], userRemoteConfigs: [[credentialsId: '68abdbaa-2131-4b53-abcc-1
10         sh 'cd Backend && mvn clean install'
11       }
12     }
13     stage('Build docker image'){
14       when { changeset "Backend/**" }
15       steps{
16         script{
17           sh 'cd Backend && docker build -t critiq/backend .'
18           sh 'docker stop backend || true'
19           sh 'docker rm backend || true'
20           sh 'docker rmi $(docker images -f "dangling=true" -q)'
21           sh 'docker run -d --network=host --name backend critiq/backend'
22         }
23       }
24     }
25     stage('Build NLP'){
26       when { changeset "NLP/**" }
27       steps {
28         script{
29           sh 'cd NLP && docker build -t nlp/backend . --no-cache'
30           sh 'docker stop nlp || true'
31           sh 'docker rm nlp || true'
32           sh 'docker rmi $(docker images -f "dangling=true" -q) || true'
33           sh 'docker run -d --network=host --restart unless-stopped --name nlp nlp/backend'
34         }
35       }
36     }
37   }
38 }

```

Figure 14 Jenkinsfile pipeline written for deploying the Backend Server and NLP Server with the use of docker and dockerfiles

3.3.7 Message Queue

RabbitMQ will be used to support inter-process communication and maintain a durable message queue for high fault tolerance and asynchronous communication. Message Queue (See Figure 15) will be used to support real-time and batch-processing ML features, enhancing system performance and reliability in case of system failure.

RabbitMQ offers various benefits including Durability, Reliability, Scalability.

- **Durability:** RabbitMQ can persist messages to disk, ensuring that they are not lost in case of a failure or a restart. This also allows for message recovery and replay.
- **Reliability:** RabbitMQ provides various features to ensure the delivery and processing of messages, such as acknowledgments, confirmations, dead letter queues, and transactions. These features help to avoid message loss, duplication, or corruption.
- **Scalability:** RabbitMQ can handle high volumes of messages and concurrent connections, as well as distribute the load across multiple nodes in a cluster. Horizontal scaling can be done easily by deploying more Python Programs without any code modification to handle a larger amount of machine learning tasks.

Compared to other popular Message Queue solutions, including Apache Kafka and Amazon SQS, RabbitMQ offers two distinct advantages: Flexible Routing, and intuitive Management User Interface.

- **Flexible Routing:** RabbitMQ supports different types of exchanges and bindings, which allow for flexible and dynamic routing of messages based on various criteria, such as topic, header, or direct, and allow for dynamic and flexible routing of messages.
- **Intuitive Management User Interface:** RabbitMQ provides a web-based user interface that allows for easy monitoring and management of the broker, such as viewing queues, exchanges, bindings, messages, connections, channels, and statistics. The user interface also allows for performing common operations, such as creating, deleting, purging, or publishing messages.

The official Spring and Python RabbitMQ adaptors will be used to connect the Spring Boot Backend and the Python NLP program to the message queue server.

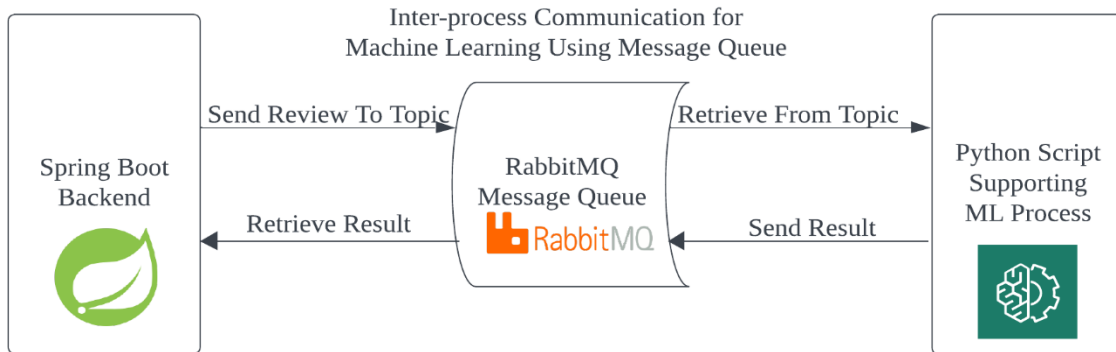


Figure 15 Message Queue Structure for Supporting Inter-process Machine Learning Application

The connections to the message queue server and the data stored in the queues can be accessed using a web-based management panel provided for debugging and analytic purposes. The first two connections are the Spring Boot Server and NLP Server Connection to the queues with the last being a local connection (See Figure 16).

Overview			Details			Network	
Name	User name	State	SSL / TLS	Protocol	Channels	From client	To client
109.██████████:49170 rabbitConnectionFactory#3555d804:0	FYP	running	○	AMQP 0-9-1	1	0 B/s	0 B/s
109.██████████:60836 ?	FYP	running	○	AMQP 0-9-1	1	2 B/s	2 B/s
42.██████████:57810 rabbitConnectionFactory#6c8e40fc:0	FYP	running	○	AMQP 0-9-1	2	0 B/s	0 B/s

Figure 16 RabbitMQ Management Panel showing all the queue connections.

3.3.8 Hosting

The backend services will be hosted on two different cloud providers, Digital Ocean, and Contabo.

Digital Ocean will host the MySQL Database, RabbitMQ Message Queue Server, and S3-compatible storage Bucket. Digital Ocean provides one-click setup and monitoring for these services on their control panel without needing to create separate virtual machines.

Contabo will only host the Virtual Machine running Ubuntu LTS 21. Virtual Machine hosted by Contabo provides high performance at a low cost of entry with high-bandwidth networking included.

All backend services will be deployed in Singapore, a region that is supported by both Digital Ocean and Contabo. By consolidating all our services in Singapore, we can reduce the latency and network traffic time between API calls and network requests and enhance the overall performance of the entire backend system. Complex API calls such as Advanced Searching or Analytics involve multiple database queries, which would be adversely affected by increasing the physical distance between the database and the Spring Boot server from sub-1 second to over 2.5 seconds.

3.3.9 Monitoring

To ensure stable performance and reliability, we use Prometheus and Grafana, two open-source software for monitoring and analytics, to monitor our Spring Boot backend application on our Virtual Machine. Prometheus collects and stores time-series data from the application actuator endpoints, and Grafana visualizes and analyzes the data in dashboards and panels.

A Grafana dashboard will be created to monitor essential information about the Spring Boot Application, including Application Uptime, CPU utilization, Application Load, and Database Connection Size (See Figures 17,18). If the application goes down for an extended period, a custom notification will be sent through a webhook.

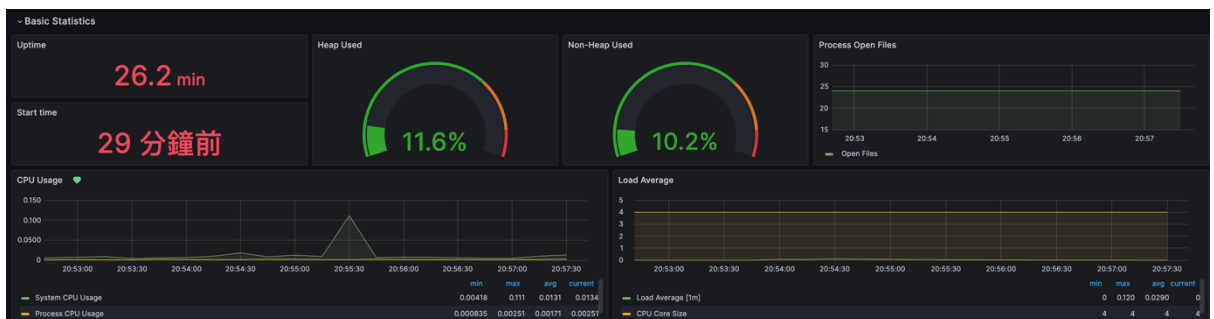


Figure 17 Grafana Dashboard displaying uptime, CPU, and Memory Utilization by the Spring Boot Application.

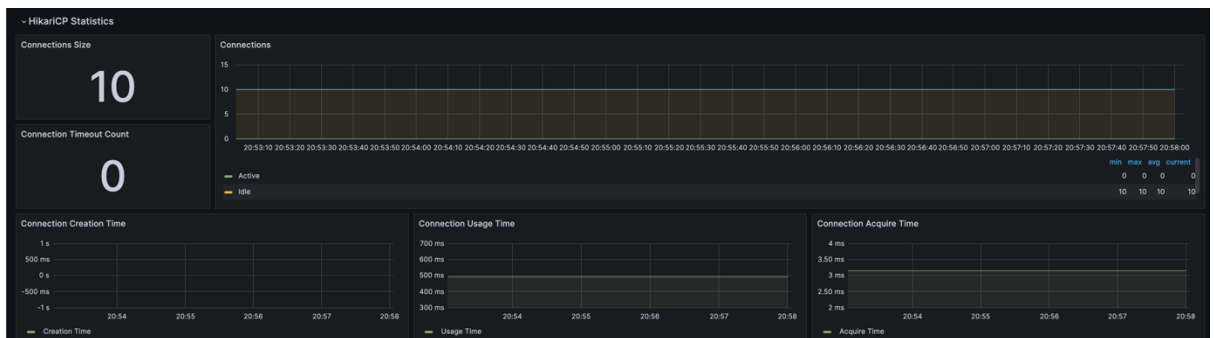


Figure 18 Grafana Dashboard shows the database connection pool size, maintaining a stable connection to the database.

4 Result

This section discusses the results discovered subsequent to the culmination of the project. These include tasks related to Machine Learning and Natural Language Processing (Section 4.1), the creation and functionality of Frontend Web Applications (Section 4.2), and the use and evaluation of Backend Technologies (Section 4.3).

4.1 Natural Language Processing Tasks

First, the results derived from the Natural Language Processing tasks are presented. These tasks include Sentiment Analysis (Section 4.1.1), Topic Modeling (Section 4.1.2), and Keyword Extraction (Section 4.1.3). For an in-depth explanation of the implementation of various NLP tasks, please refer to the report by my group mate, Cheng Pak Yim, Michael, who oversees the NLP tasks.

4.1.1 Sentiment Analysis

The evaluation was conducted on both balanced and imbalanced validation sets to address the three research questions, thus selecting the best model for deploying on the VM.

RQ1: Does an imbalanced training dataset hamper model performance?

Results of all models trained with 120K balanced and imbalanced datasets were presented in Figure 19.

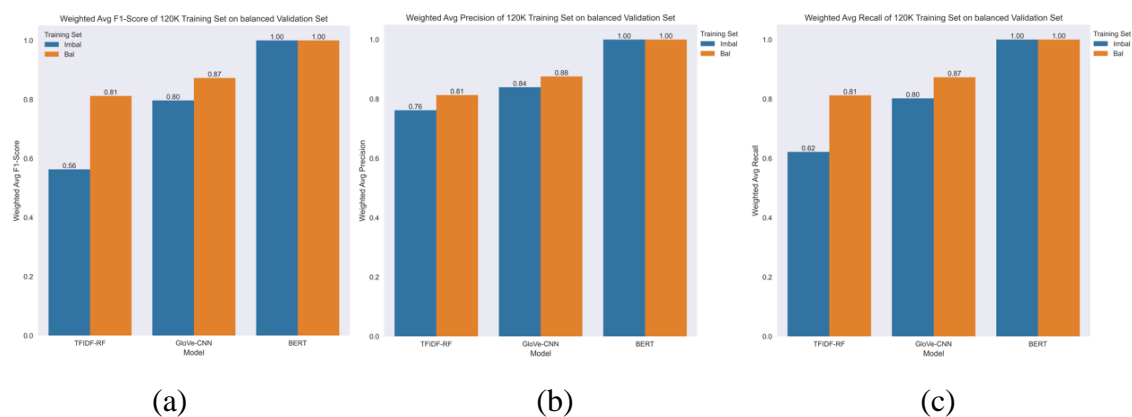


Figure 19: Results of all models trained with 120K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.

It was observed that TFIDF-RF and GloVe-CNN received a lower weighted average F1-score on the balanced validation set, with TFIDF-RF models receiving the biggest difference in the score between training with a balanced dataset and an imbalanced dataset. Since the weighted average F1-score considered both weighted average recall and precision, the difference was surmised to be a drop in recall and/or precision.

Considering the weighted average recall of models trained with the 120K dataset, both TFIDF-RF and GloVe-CNN models trained with an imbalanced dataset received lower recall than those with the balanced dataset. The same result was also noted in precision. To further dig into the cause of the difference in weighted average precision and recall, a comparison was drawn on these metrics in both positive and negative sentiment classes. Regarding weighted average precision, although achieved higher precision in classifying negative sentiment, models trained with an imbalanced dataset fell short in correctly classifying positive sentiment samples (See Figure 20). Models trained with a balanced dataset achieved more all-rounded performance in precision, resulting in higher weighted average precision (See Figure 19(b)). It is surmised that models trained with imbalanced datasets implicitly learned the distribution of the training dataset, and then made more positive guesses to reviews, leading to lower precision in positive sentiment, and higher precision in negative sentiment. While models trained with balanced datasets attained balanced performance in both sentiment classes, resulting in higher weighted average precision in balanced datasets. Regarding weighted average recall, although achieved higher recall in positive sentiment, models trained with an imbalanced dataset fell short in recalling negative sentiment samples (See Figure 21). While models trained with a balanced dataset achieved more all-rounded performance in recall (Figure 19(c)), resulting in higher weighted average recall. The observation further supported our earlier conjecture, leading to a much lower recall of negative reviews, and achieving a nearly perfect recall of positive reviews.

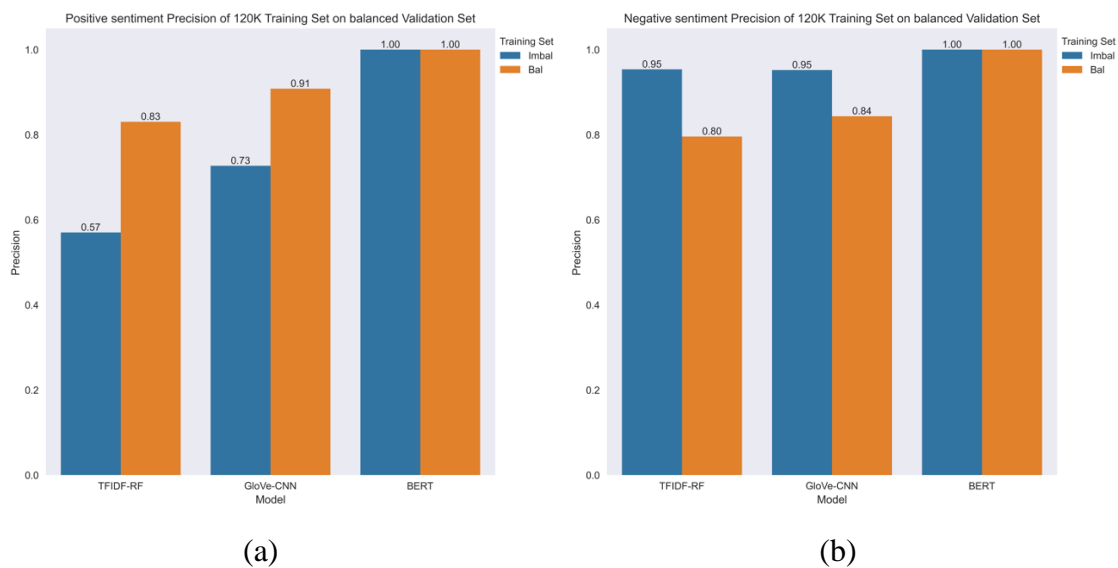


Figure 20: Precision of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative.

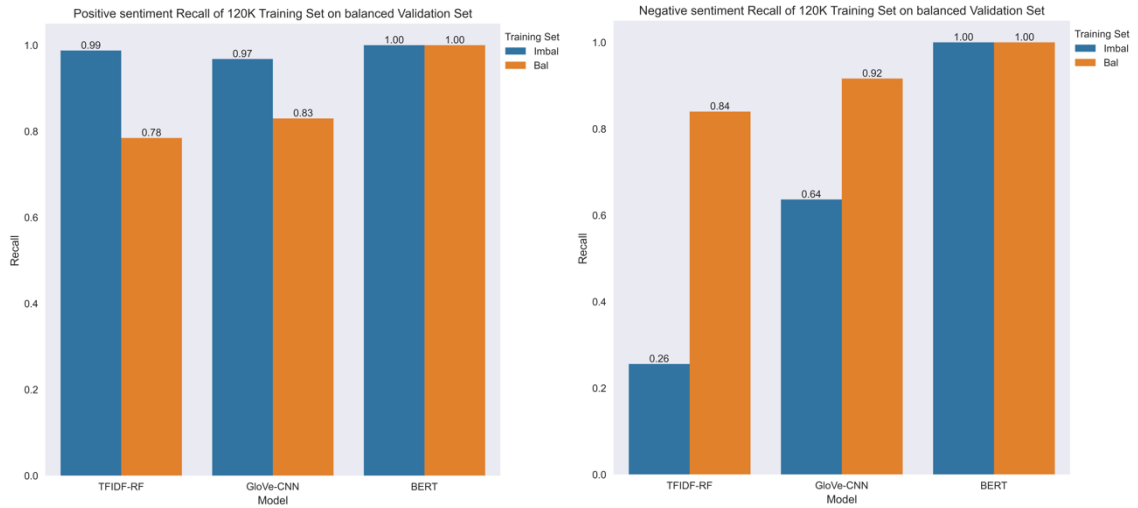


Figure 21: Recall of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative.

Referring to Figures 22 and 23, the same conclusion that models trained with an imbalanced dataset underperformed those trained with a balanced dataset was observed in training with both 240K and 480K training datasets. The consistent observation suggested that training with a balanced dataset was preferred, as it yielded higher and more balanced performance.

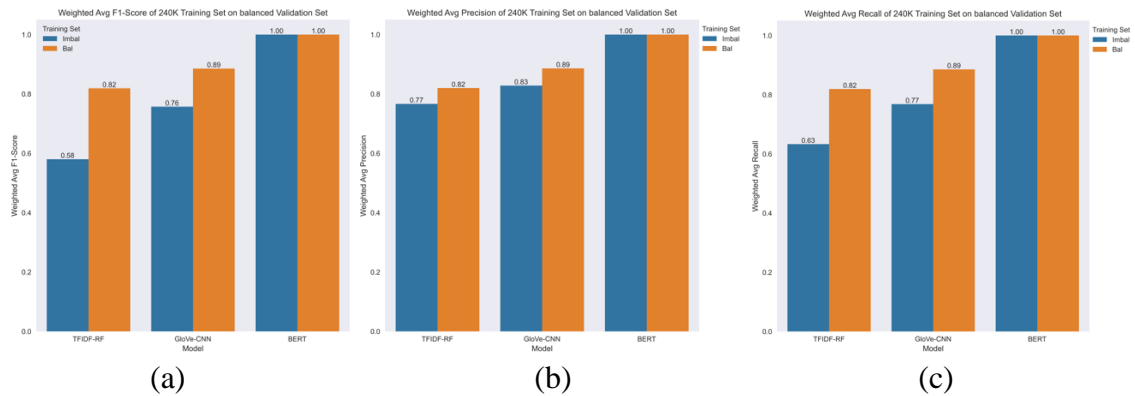


Figure 22: Results of all models trained with 240K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.

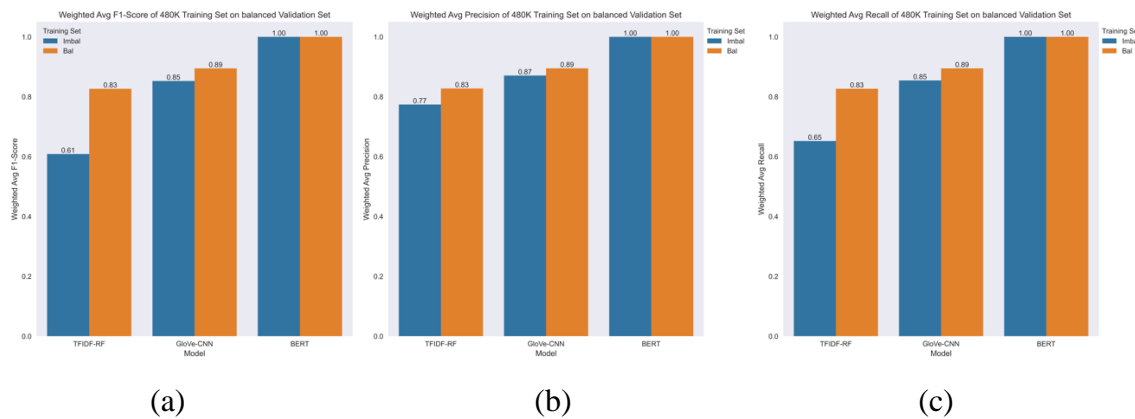


Figure 23: Results of all models trained with 480K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.

RQ2: What is the relationship between dataset size and performance?

Referring to Figure 24 (a), among the models trained with a balanced training set with different sizes, the Weighted Average F1-score of TFIDF-RF and GloVe-CNN increased as the size of the training set increased. The percentage increase was between 0.84% and 1.46% (See Table 1 (Up)). Further breakdown of the Weighted Average F1-score showed that both Weighted Average Precision and Recall increased as the size of the training set increased. The percentage increase in Weighted Average Precision was between 0.82% and 1.17% (See Table 1 (Bottom left)), while that in Weighted Average Recall was between 0.84% and 1.44% (See Table 1 (Bottom right)).

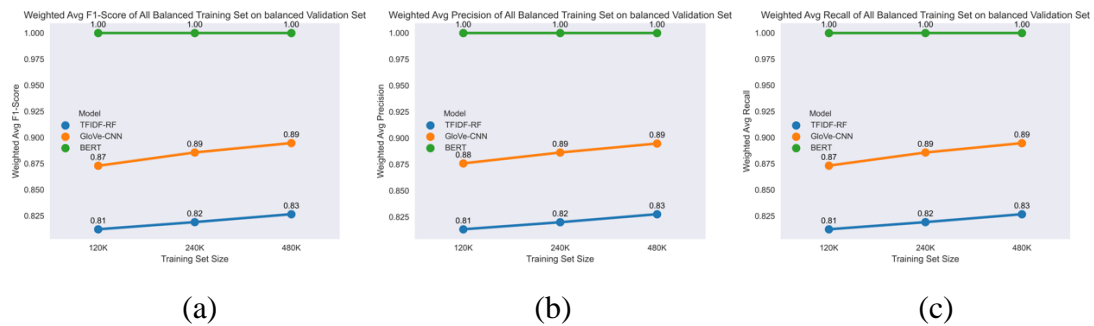


Figure 24: Results of all models trained with balanced datasets of all three sizes. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models

Model/Size	TFIDF-RF	GloVe-CNN	BERT
120K	0%	0%	0%
240K	+0.84%	+1.46%	0%
480K	+0.93%	+1.02%	0%

Model/Size	TFIDF-RF	GloVe-CNN	BERT	Model/Size	TFIDF-RF	GloVe-CNN	BERT
120K	0%	0%	0%	120K	0%	0%	0%
240K	+0.84%	+1.46%	0%	240K	+0.84%	+1.46%	0%
480K	+0.93%	+1.02%	0%	480K	+0.93%	+1.02%	0%

Table 1: Percentage change of all models trained with balanced datasets with different sizes. Up: Weighted Average F1-score. Bottom left: Weighted Average Precision. Bottom right: Weighted Average Recall.

A similar result can be observed in models trained with imbalanced datasets of all three sizes. The Weighted Average F1-score also increased as the size of the imbalanced dataset increased, except for model GloVe-CNN, which suffered from a drop in

performance in the 240K imbalanced dataset (See Figure 25). Considering only positive percentage changes, the percentage increase of the Weighted Average F1-score ranged between 3.05% and 12.61% (See Table 2 (Up)). Further breakdown of the Weighted Average F1-score showed that both Weighted Average Precision and Recall increased as the size of the training set increased. Considering only positive percentage changes, the percentage increase in Weighted Average Precision was between 0.61% and 5.15% (See Table 2 (Bottom left)), while that in Weighted Average Recall was between 1.81% and 11.20% (See Table 2 (Bottom right)).

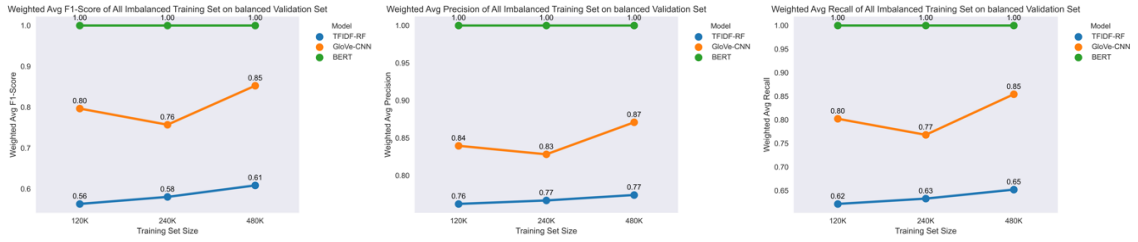


Figure 25: Results of all models trained with imbalanced datasets of all three sizes. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models

Model/Size	TFIDF-RF	GloVe-CNN	BERT
120K	0%	0%	0%
240K	+3.05%	-4.96%	0%
480K	+4.85%	+12.61%	0%

Model/Size	TFIDF-RF	GloVe-CNN	BERT	Model/Size	TFIDF-RF	GloVe-CNN	BERT
120K	0%	0%	0%	120K	0%	0%	0%
240K	+0.61%	-1.35%	0%	240K	+1.82%	-4.24%	0%
480K	+0.95%	+5.15%	0%	480K	+3.02%	+11.20%	0%

Table 2: Percentage change of all models trained with imbalanced datasets with different sizes. Up: Weighted Average F1-score. Bottom left: Weighted Average Precision. Bottom right: Weighted Average Recall.

RQ3: What is the best model with little hyperparameter selection?

Model	Balanced training set			Imbalanced training set		
	120K	240K	480K	120K	240K	480K
TFIDF-RF	0.82	0.82	0.83	0.84	0.84	0.85
GloVe-CNN	0.86	0.90	0.90	0.91	0.90	0.92
BERT	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>

Table 3: Weighted Average F1-Score of all models on the imbalanced validation sets.

Model	Balanced training set			Imbalanced training set		
	120K	240K	480K	120K	240K	480K
TFIDF-RF	0.81	0.82	0.83	0.56	0.58	0.61
GloVe-CNN	0.87	0.89	0.89	0.80	0.76	0.85
BERT	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>

Table 4: Weighted Average F1-score of all models on the balanced validation sets.

Referring to Table 3, both models trained with an imbalanced dataset and balanced dataset received similar weighted average F1-score on the imbalanced validation set, even the models trained with an imbalanced dataset slightly out-performed those trained with the balanced dataset. However, in Table 4, as mentioned in RQ1, models trained with an imbalanced dataset fell short in performance in a balanced validation set.

Although the majority of games receive chiefly positive reviews, there will be situations in which mixed reviews or more extreme, mostly negative reviews will be received on the game platform. They can occur due to the game’s poor quality, sluggish gameplay, or frequent bugs, resulting in more than usual negative reviews. Examples are The Last of Us 2, the initial release of No Man’s Sky, and Lord of Rings: Gollum. Moreover, some of the games were under early access reviews, in which a demo of the game was released on the platform for eager players to test and provide feedback although the game was under development process. Since the game was unpolished and incomplete, it was expected for the developers to receive a mixed review. Therefore, considering the performance of models in both balanced and imbalanced validation datasets, models trained with balanced datasets were preferred. Referring to Table 3 and Table 4, the best performant model was BERT, achieving a perfect weighted average F1-score in both imbalanced and balanced validation datasets.

Since there was a tie in the weighted average F1-score of all BERT models, a different metric is required to select the best performant model with respect to the size of the training dataset. Receiver Operating Characteristic – Area Under the Curve (ROC-

AUC) was chosen to be the metric as it measures the performance of a model at different classification thresholds. A prediction from a model can be treated as a probabilistic prediction, with each class containing a value $[0, 1]$, and the sum of them equal to 1. With the information, the True Positive Rate (TPR) against the False Positive Rate (FPR) at different prediction thresholds can be plotted, creating the ROC curve. To find out the ROC-AUC value, the area under the ROC curve is calculated. A ROC-AUC value = 0.5 represents a random classifier, and a ROC-AUC value = 1.0 represents a perfect classifier. Classifiers with ROC-AUC value < 0.5 are considered worse than a random classifier, and vice versa. The higher the ROC-AUC value, the more performant the classifier is.

Referring to Figure 26, BERT fine-tuned with 240K balanced dataset scored the highest ROC-AUC among all three BERT models in the imbalanced validation set, achieving a 0.68 in ROC-AUC. A similar result on a balanced validation set was also observed. Therefore, the BERT model fine-tuned on a 240K balanced training set was the best performing model. Its ROC curves on both imbalanced and balanced validation sets are displayed in Figure 27.

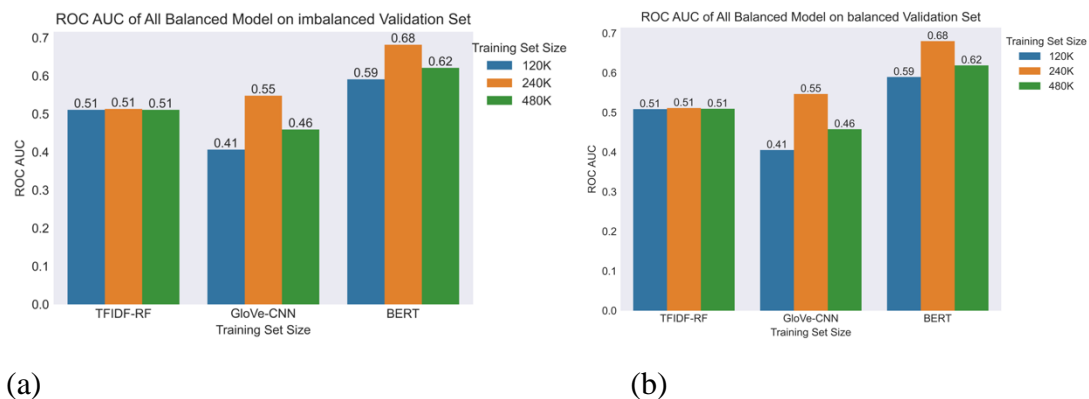


Figure 26: ROC-AUC of all models trained with the balanced dataset. (a): on the imbalanced validation set. (b): on the balanced validation set.

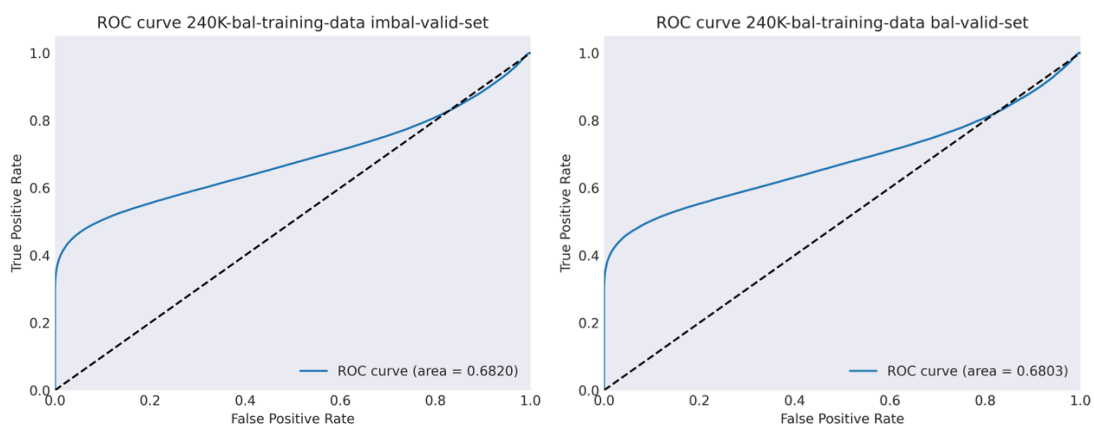


Figure 27: ROC curve of BERT fine-tuned on 240K balanced training set. (a): on the imbalanced validation set. (b): on the balanced validation set.

Regarding the required inference time for one review, it was obvious that speedup was recorded for all three models on both machines. The median inference time, lower and upper quartile were plotted in Figure 28. Among the three models, TFIDF-RF recorded the largest speedup on both machines, followed by GloVe-CNN, and lastly BERT (See Table 5). The large variation in inferencing times on BERT was attributed to its higher complexity compared to the other two models.

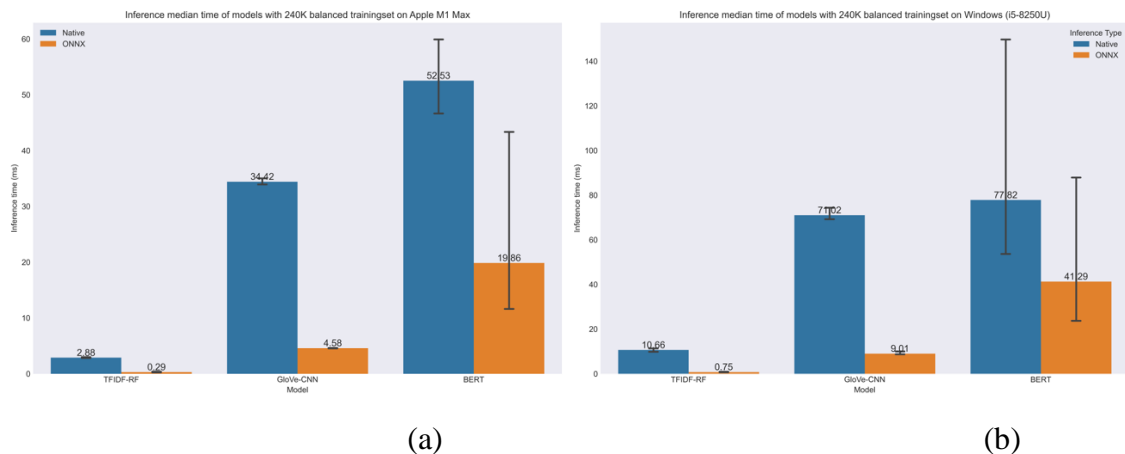


Figure 28: Median inference time of original and ONNX model on different machines. (a): Machine 1 (Windows i5-8250U). (b): Machine 2 (Apple M1 Max)

	Machine 1 (i5-8250U)	Machine 2 (M1 Max)
TFIDF-RF	14.21	9.93
GloVe-CNN	7.88	7.51
BERT	1.88	2.65

Table 5: Median Speedup of inference time of both machines.

Therefore, considering the conclusion of three research questions and results on inference time evaluation, BERT finetuned on a 240K balanced training set, converted to ONNX format, was selected for deployment on the VM.

4.1.2 Topic Modeling

For simplicity's sake, the two genres that are most reviewed will be selected to train the topic models. After data scrapping was performed, we retrieved 4.05M of game reviews. After some processing tasks, a comprehensive dataset of all potential game genres was created. Based on the characteristic of the review dataset (See Figure 29), the top two genres with the highest amount of game review are action and indie, which contain 1.31M and 0.74M reviews, respectively. In total, they accounted for around 50% of all reviews, therefore, the topic models trained with these two genres should be able to handle most of the reviews.

	genre_id	description	f
0	1	Action	1314407
1	23	Indie	741913
2	25	Adventure	636492
3	3	RPG	545025
4	2	Strategy	409708
5	28	Simulation	260097
6	37	Free to Play	246372
7	4	Casual	209223
8	29	Massively Multiplayer	94777
9	9	Racing	25863
10	18	Sports	24616
11	51	Animation & Modeling	7431
12	58	Video Production	6876
13	57	Utilities	2651
14	53	Design & Illustration	1943
15	74	Gore	1470

Figure 29 List of game genres and the review frequencies

To evaluate the performance of the topic models trained with the action and indie genres and test the capacity of the models. We applied some reviews from unseen games with the same genre to the model. Only reviews written since 2023 were selected. For different games, the number of reviews selected ranged from 2 to around 290000. Usually speaking, indie games created by small developer teams are inclined to have few reviews, whereas triple-A titles published by renowned game publishers tend to have hundred thousand reviews.

As mentioned in the methodology section of the topic modelling (section 3.1.2). The topic model will be evaluated both quantitatively and qualitatively.

For the quantitative evaluation, as discussed before, the topic coherence and diversity of the topic models will be measured using NPMI score and the inverted RBO score, respectively. Three topic models, each using a different architecture (which include

BERTopic, LDA and CTM) were trained. Regarding architectures that tackle sentence embedding (BERTopic and CTM), the models were trained with a special technique called token splitting, where the reviews were split if the length exceeded the limited tokens length of sbert, which is 384 tokens. This is to guarantee that the model will be able to learn different topics from the same review. According to the analysis result of the NPMI score on the models trained with reviews data from action genre, the BERTopic architecture performed significantly better than LDA and CTM in terms of topic coherence, and CTM performed slightly better than LDA (See Figure 30).

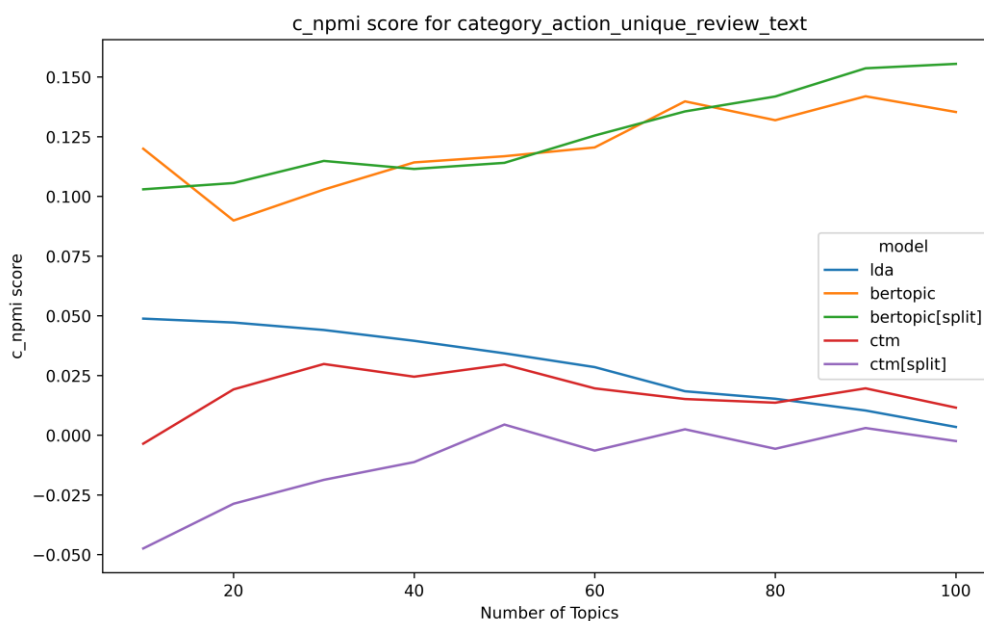


Figure 30 NPMI score of models trained with different architectures and reviews on games in the action genre

Based on the quantitative analysis results (refer to Michael report for the full results), we selected the top two architectures that performed the best, which are BERTopic and LDA, to undergo further qualitative analysis.

Since the quantitative evaluation is unable to measure the human readability of the results generated by the topic models, which often is the most important metric to the users, we recognize the significance of manual examination in evaluating the topic models through the qualitative manner.

Regarding the qualitative analysis process, cross model comparison was performed on the topics of “crashes and bugs” and “horror puzzle game”. The results showed that while both architectures perform similarly in terms of the generated keywords. BERTopic performed much better than LDA in extracting meaningful representative documentation from all of the reviews. This result is expected as LDA performs

lemmatization and stemming, causing the modal trained with LDA to favor reviews with repetitive words, whereas BERTopic focuses on contextual meaning of the sentence and unlikely to affect by the repetitive review context (see Table 6 and 7).

	LDA	BERTopic
Top 10 Keywords	'work', 'fix', 'play', 'crash', 'bug', 'issue', 'problem', 'computer', 'patch', 'window'	'crashes', 'windows', 'crashing', 'crash', 'account', 'fix', 'work', 'play', 'start', 'wont'
Top Representative Document	Crash! Crash! Crash! Crash! Crash! Crash! Crash! Crash! Crash! Crash! ...	Good game, but it CRASHES. TOO. MUCH.
	crashes, crashes, crashes, crashes, crashes, crashes, crashes, crashes, ...	Have Windows 8 or Windows 10? This game won't run on your computer without crashing every 5 minutes.

Table 6 Representative texts and topic keywords generated by the topic models on the topic of "crashes and bugs"

	LDA	BERTopic
Top 10 Keywords	'puzzle', 'music', 'art', 'style', 'atmosphere', 'gameplay', 'sound', 'horror', 'design', 'movie'	'puzzles', 'puzzle', 'horror', 'like', 'really', 'good', 'scary', 'great', 'play', 'characters'
Top Representative Document	Puzzles, inside of puzzles, inside of puzzles, inside of puzzles, inside of puzzles, ...	Great graphics, wonderful story and puzzles. Will make you think at the end.
	I couldn't stand playing anymore. SO SCARY!SO SCARY!SO SCARY!SO SCARY!SO SCARY!SO SCARY!SO SCARY!SO SCARY!SO SCARY!SO SCARY! ...	I've just played this game through and I can definitely recommend it to those, who enjoy casual adventure (specifically hidden- object/simple puzzle) games....

Table 7 Representative texts and topic keywords generated by the topic models on the topic of "horror puzzle game"

Based on the results of the quantitative and quantitative evaluation. BERTopic is selected for training the topic models, as this architecture offers the best performance out of the three.

Regarding the task of topic naming. Following the methodology, the use of LLM with prompt engineering to generate a concise topic name using the result of top 10 keywords and most representative texts are explored. It is discovered that the LLM with Llama-2-7b architecture is able to generate a sensible and concise topic name with prompting within a reasonable amount of time.

Moreover, experiments on transfer learning with BERTopic models are conducted. For the BERTopic model trained with game reviews for action and indie genres, it learned to identify topics specific to these genres. The aim of transfer learning is to use this pre-trained topic model on a new problem, specifically, to identify topic names in games with genre other than action or indie. It is found that though contextual embedding, the pre-trained BERTopic model can be applied on unseen games that belong to the other genre and generate meaningful topic names for analysis. The pre-trained topic model with action game reviews was applied on the game Starfield, an action RPG to generate meaningful topic names catering to that game with exceptional result (See Figure 31).



Figure 31 Topic name results after applying the pre-trained topic model on action game reviews to the game Starfield

4.1.3 Keyword Extraction and LLM Prompting

As mentioned in the methodology of keyword extraction (See Section 3.1.3). LLM with prompt engineering is selected to implement the keyword extraction task rather than a certain keyword extraction algorithm. This is because these algorithms are unable to provide human comprehensible names for the generated keywords and topics, which is considered as the most important element in understanding its meaning. Tools of LangChain, Mistral AI and chroma, along with the prompt techniques of RAG, generated knowledge, one-shot prompting, use of delimiters and role prompting are incorporated for this task. The purpose of prompt engineering is to reduce the likelihood of LLM producing hallucinations, and to assist the LLM in generating more consistent and concise outcomes. This leads to the creation of keywords of superior precision and quality related to the review. It also streamlines the process of extracting, formatting, and saving the results to the database, making it easier to incorporate them into the front-end application.

The keyword extraction process, as depicted in Figure 32, operates in the following manner: Firstly, when a new review is created by the user, a message with the review context will be directed to our system through RabbitMQ, triggering the keyword extraction task. Then the LLM is tasked to analyze the review to determine if it is spam. If the review is indeed spam, then the process is halted, and no further keyword extraction task is carried out as it would be pointless. If not, the LLM is prompted sequentially to analyze the review based on different review aspects. These aspects, ten in total, include graphics and art design, price, gameplay, accessibility, performance, sound, bug, narrative, suggestion and overall. Notably, the 2-4-4 approach is adopted where the LLM first extracts keywords from the initial two aspects, then the next four, and finally the last four. This strategy minimizes token usage and costs, as each prompt consumes tokens, and reduces the likelihood of hallucinations, which are more probable when the LLM is asked to extract keywords from too many aspects at once. During this analysis, two subtasks are executed: keyword extraction and sentiment classification of these keywords in relation to the aspect being analyzed. Finally, if the review context exceeds 50 words, the LLM is prompted to generate a summary of the review.

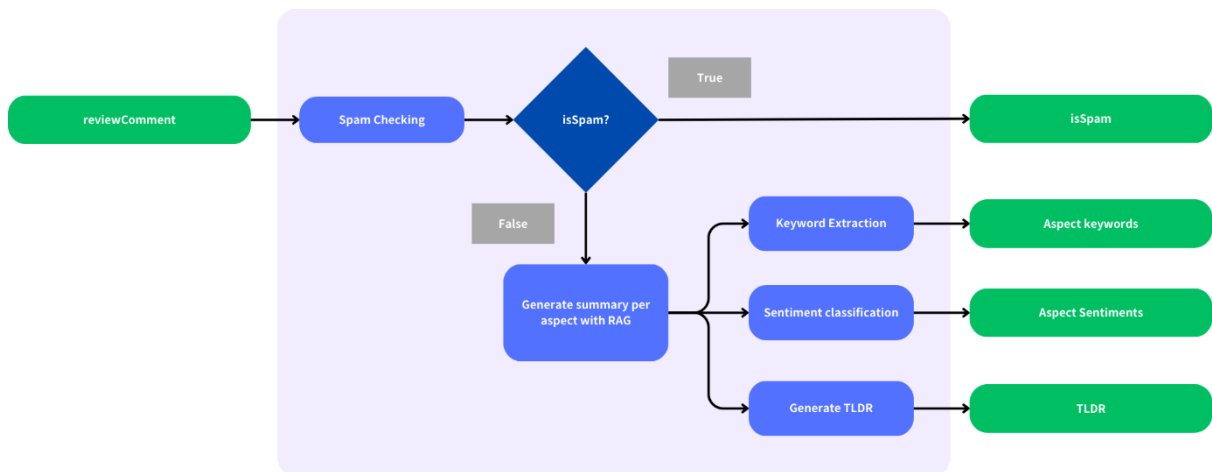


Figure 33 Procedure of keyword extraction with LLM and prompting

Another implemented LLM Prompting task involves generating a summary of all reviews for a specific game on the platform. The process, as depicted in Figure 33, begins with the use of the gameID to fetch the results of sentiment analysis and topic modeling. Following this, the LLM accesses a vector database containing game reviews written by game critics, and is tasked with summarizing each aspect. In the course of this process, the LLM also creates a brief description based on the sentiment analysis results. Finally, the LLM uses the aspect information, sentiment analysis results, and the most prevalent topics from all reviews to create a concise and comprehensive summary for the game.

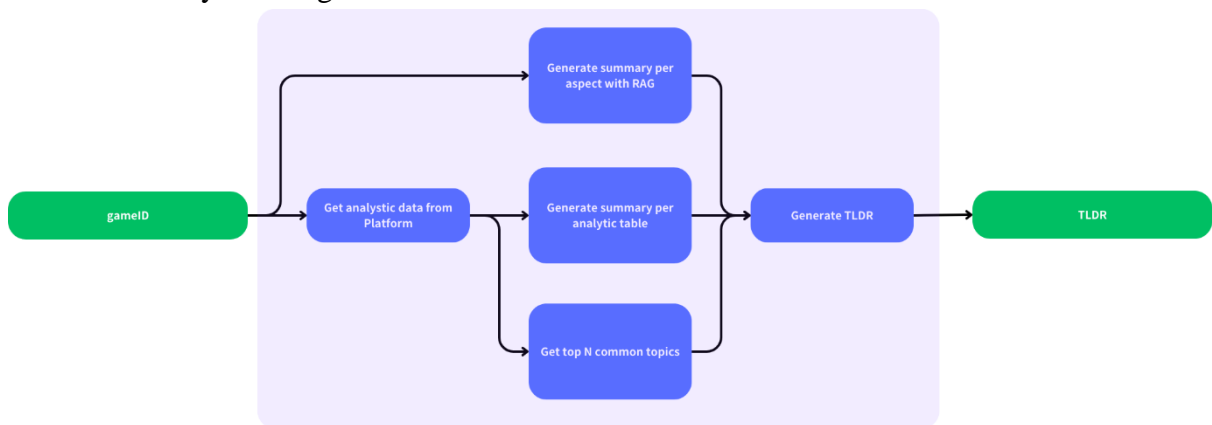


Figure 32 Procedure of aggregated review summary generation with LLM and prompting

This section further explores the use of prompt engineering with the LLM, demonstrating how this method can minimize the generation of hallucinated results and aid the LLM in producing outputs with a consistent format.

The subsequent picture presents the template of the LLM prompting for extracting keywords from the reviews (See Figure 33). The text marked in green denotes the role prompting technique, where the role of a player reading game reviews is assigned to

the LLM. This assignment helps control the context's format and style generated by the LLM.

The text highlighted in yellow represents the one-shot prompting techniques, providing an example of the desired output format for the LLM. This method guides the LLM to adhere to our provided output schema, resulting in more relevant and reliable outputs.

The text in purple shows the prompt restrictions used to minimize hallucination. In this instance, it prevents the LLM from outputting results in formats other than JSON and from presenting inaccurate keyword information when the answer is unknown because the review context does not discuss certain aspects.

The text in pink indicates the use of delimiters, specifically a special character combination of `\`\``, to define the boundaries that include the review context. This technique helps eliminate ambiguity, which could lead the LLM to produce hallucinated responses or structures with inaccuracies.

Lastly, the text in blue denotes the RAG technique, where the LLM is provided with review summaries extracted from the vector database as content and prior knowledge. This technique is vital in preventing hallucination.

```
System: You are a player reading reviews of a game to understand the characteristics of the game. Use the following pieces of context to answer user's question.

User: You are reading reviews of a game to understand the characteristics of the game. Extract the following aspect of the game from the reviews. The aspects are {aspects}. For each aspect, output a paragraph with less than 50 words. Then create a JSON with aspects name as key and the paragraph as value.
Output the JSON as a single line with no spaces between the key, value pairs. {For example, if the aspects are {aspects list}, the JSON should be: {"aspect01": "...", "aspect02": "...", "aspect03": "..."} }
Only output the JSON. Do NOT output other text.

The reviews are as follows:
\`\`
{summaries}
\`\`
If you don't know the answer, output only \`NA\`. Do NOT try to make up an answer. Do NOT output other text.
```

Figure 34 Prompting template to extract keywords from the reviews for the LLM

With the use of this prompting template, the LLM can accurately extract keywords from reviews and generate responses in a consistent, predefined format. This assists in converting the response into a JSON object for future operations and integration with both backend and frontend systems (refer to Figure 35).

Moreover, the response generated by the LLM successfully prevents hallucinated responses. If aspects are not discussed in the review context, the LLM responds with "N/A".

Regarding the token spent for this request. The total token usage is 597, where the prompt requires 520 tokens and the completion cost 77 tokens.

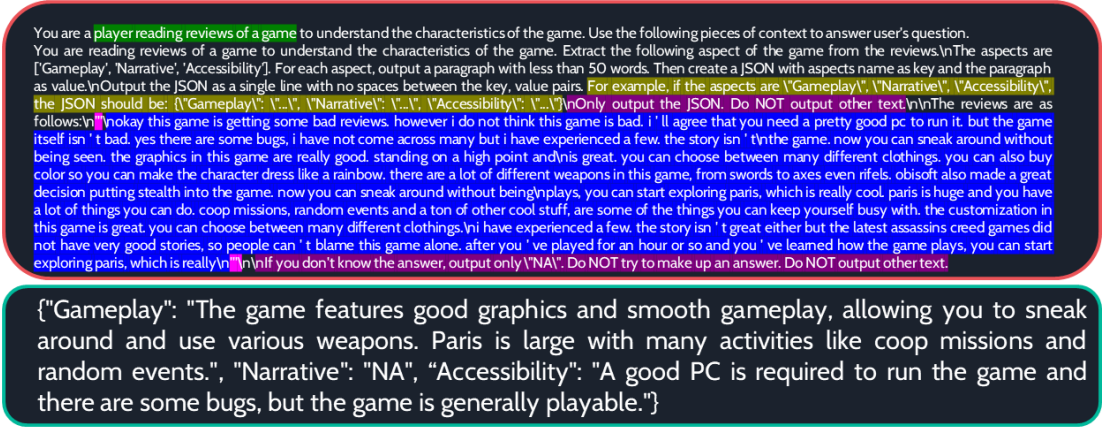


Figure 35 An example prompt and the returned response from the LLM

4.2 Web Application

This section explains the design, features, and implementation of the web application. The web application was developed using the technologies, framework and design approach specified in the proposed methodology (Section 3.2). The pages and important user interface elements that have been developed are the toolbar (Section 4.2.1), register and login popup modal (Section 4.2.2), forget password and reset password pages (Section 4.2.3), landing page (Section 4.2.4), search result page (Section 4.2.5), game page (Section 4.2.6), game reviews page (Section 4.2.7), review page (Section 4.2.8), game analytics page (Section 4.2.9), and profile page (Section 4.2.10). The following sub-sections will discuss and explain the mentioned pages and components in detail. As outlined in the section on design methodology (See Section 3.2.2), our web application adheres to the principles of Responsive Web Design (RWD). This ensures that our application adapts to the user’s viewport. In the subsequent section detailing the results of our web application, we will provide screenshots from both desktop and mobile viewports to illustrate how the user interface adapts to different devices.

4.2.1 Toolbar

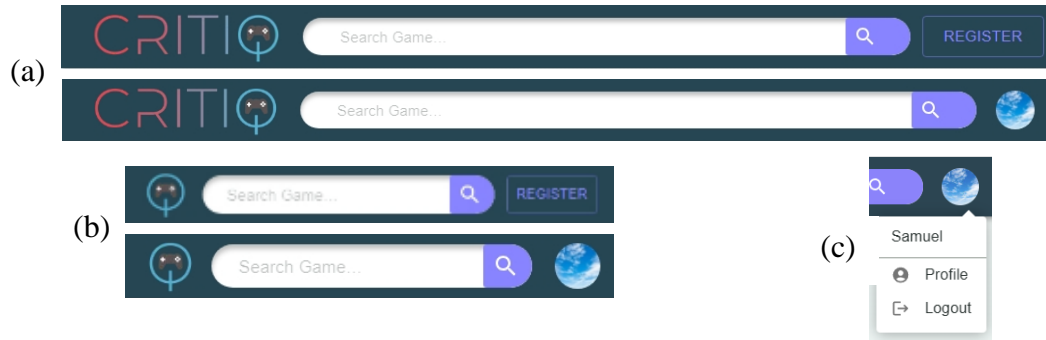


Figure 36 Web application's toolbar design. (a): Toolbar design for desktop viewport. (b): Toolbar design for mobile viewport. (c): Avatar icon button drop down menu.

The toolbar was implemented using the AppBar and Toolbar components from MUI. The layout of the toolbar was customized to suit the needs of our web application and consisted of three major sections (See Figure 36 above).

On the left is the app icon button that redirects the user to the landing page when clicked. In the middle, there is a search bar that allows the user to search for games with game title. If no input is given, it will search for all the games in the database. The user can initiate the search by clicking the search button or pressing the enter key while typing the input field, after that, it will take them to the search results page.

Finally, on the right, there is either a register button or a user avatar button, depending on the user's login status. The register button opens a popup modal that enables the user to create a new account or sign in to their existing one. The details of the popup modal's implementation, features, and design will be discussed in Section 4.2.2. The user avatar button opens a menu (See Figure 36(c)) that displays the username and two button options: the profile button and the logout button. The profile button takes the user to the profile page, while the logout button signs the user out of the web application.

The toolbar follows the principles of Responsive Web Design (RWD) in its design and implementation. For the viewport of mobile devices (See Figure 36(b)), the app icon button is substituted by a simplified version of the icon, the spacing between the three components is minimized, and the dimensions and font size of the register button are scaled down. These measures ensure that the toolbar can adapt to the smaller viewport and offer the optimal user experience.

The toolbar also adopts a dynamic display strategy to enhance the website's visual clarity and information density. The toolbar disappears when the user scrolls down and reappears when the user scrolls up.

4.2.2 Login and Registration

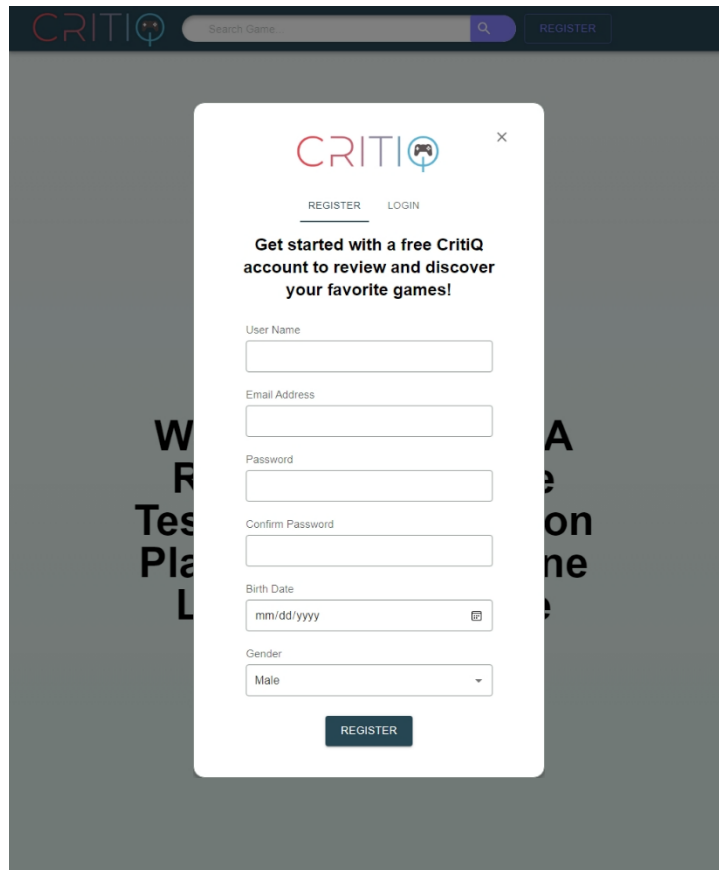


Figure 37 Web application's register modal popup

User Name

Your username must be 4 to 14 characters long with no spaces or @ symbols.

Email Address

Email address cannot be empty

Password

Your password should have:

1. A minimum of 8 and a maximum of 16 characters
2. Contains both numbers and letters

Confirm Password

Your password should have:

1. A minimum of 8 and a maximum of 16 characters
2. Contains both numbers and letters

Birth Date

You must be at least 13 years old to register an account

Figure 38 Register modal input validations

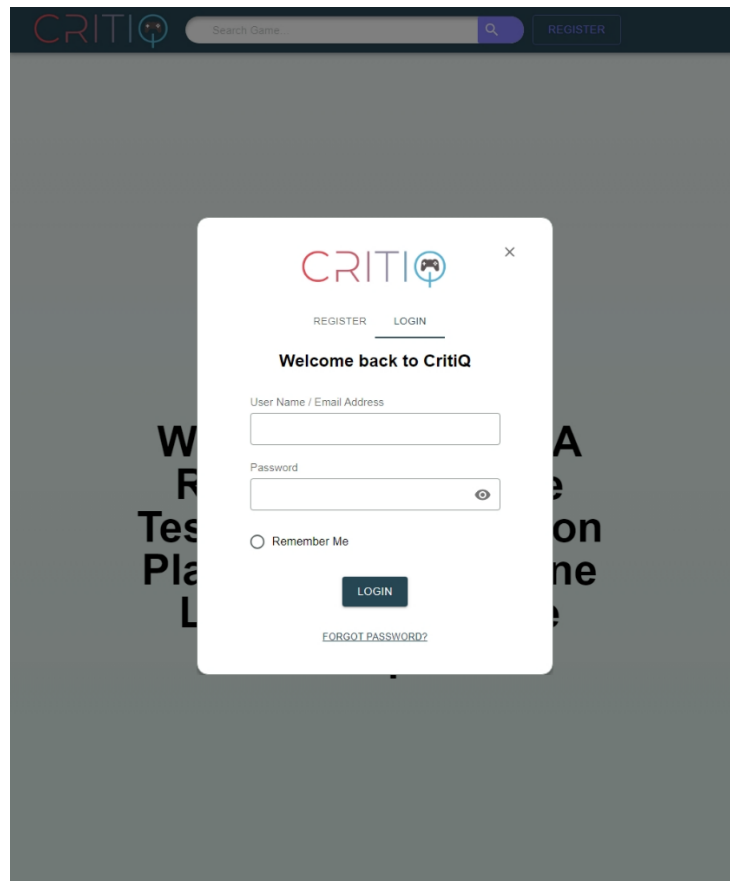


Figure 39 Web application's login modal popup

As mentioned in the previous sections, the popup modal can be accessed by the users through clicking the register button on the toolbar. This modal enables the users to either register a new account or log in to their existing account. The Modal component from MUI is utilized to implement this popup modal and the input fields are a customized version of the InputBase component from MUI. The layout of the popup modal consists of the web application icon and an icon button to close the modal on the top, and the tab bar below the icon. The Tabs and Tab components from MUI are employed to implement the tab bar. The users can toggle between the register modal and the login modal by selecting the corresponding tab.

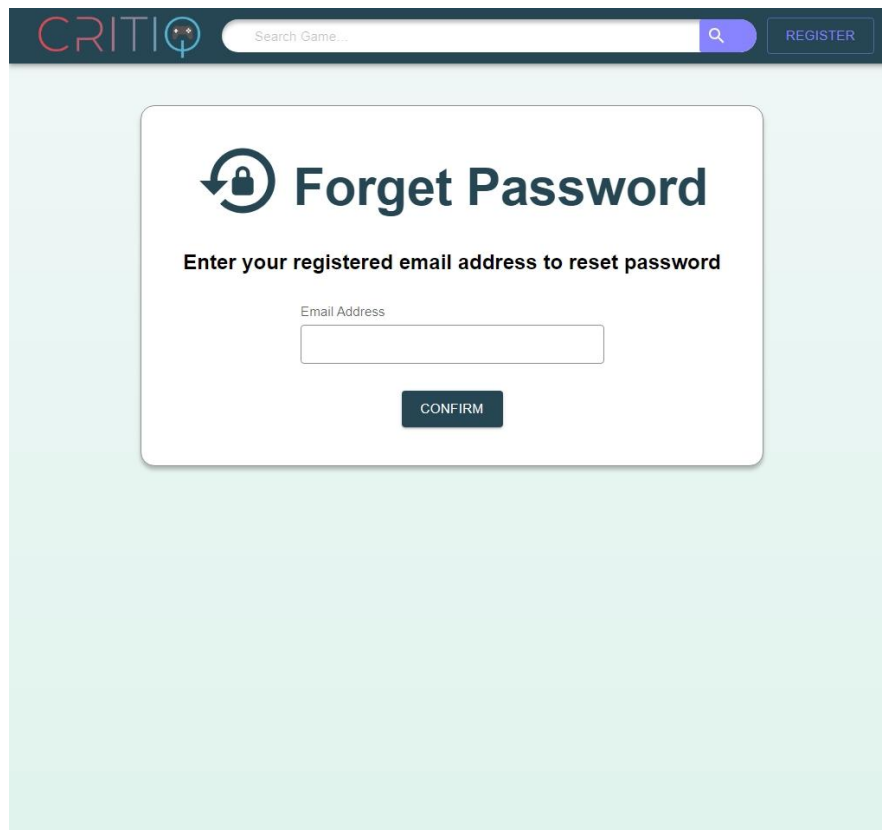
The register modal (See Figure 37 above) requires the users to enter all the fields to create a new account, which comprise username, email address, password, confirm password, birth date and gender. The front-end application performs validations on all the input fields using regular expressions (See Figure 38) based on the following 2 rules. This is crucial as it reduces the network load for the backend system by preventing the users from making invalid requests.

1. The username must have a length of 4 to 14 characters with no spaces or @ symbol, as this symbol is designated for the email address detection.
2. The password must have a length of 8 to 16 characters with both number and letter to ensure its security.

The backend system also validates the register request to avoid duplicate usernames or email addresses in the database.

The login modal (See Figure 39) requires the users to enter the username or email address and password to sign into their account. The icon button on the right of the password input field allows the users to hide or show the password by changing the type of input field between text and password. The “Remember Me” checkbox below the two input fields determines how the refresh token cookies are stored, as explained in Section 3.2.2. The backend system is responsible for the validations, and it will return an error message with a description of the issue for the invalid login request. The modal will display this error message to inform the user. If the user has forgotten their password, they can click the forget password button below the login button, which will redirect them to the forgot password page. The next section will describe the implementation of this page.

4.2.3 Forgot Password Page and Reset Password Page



The screenshot shows the 'Forgot Password' page of the CritiQ web application. At the top, there is a dark navigation bar with the 'CRITIQ' logo on the left, a search bar in the center, and a 'REGISTER' button on the right. The main content is a white card with a rounded border. It features a circular icon with a lock and a refresh symbol, followed by the heading 'Forgot Password'. Below the heading, the text reads 'Enter your registered email address to reset password'. There is a text input field labeled 'Email Address' and a dark 'CONFIRM' button below it.

Figure 40 Web application's forgot password page design.

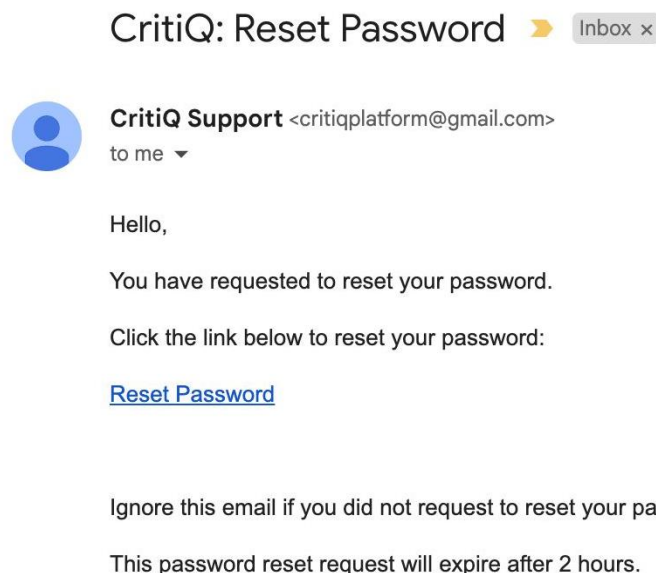
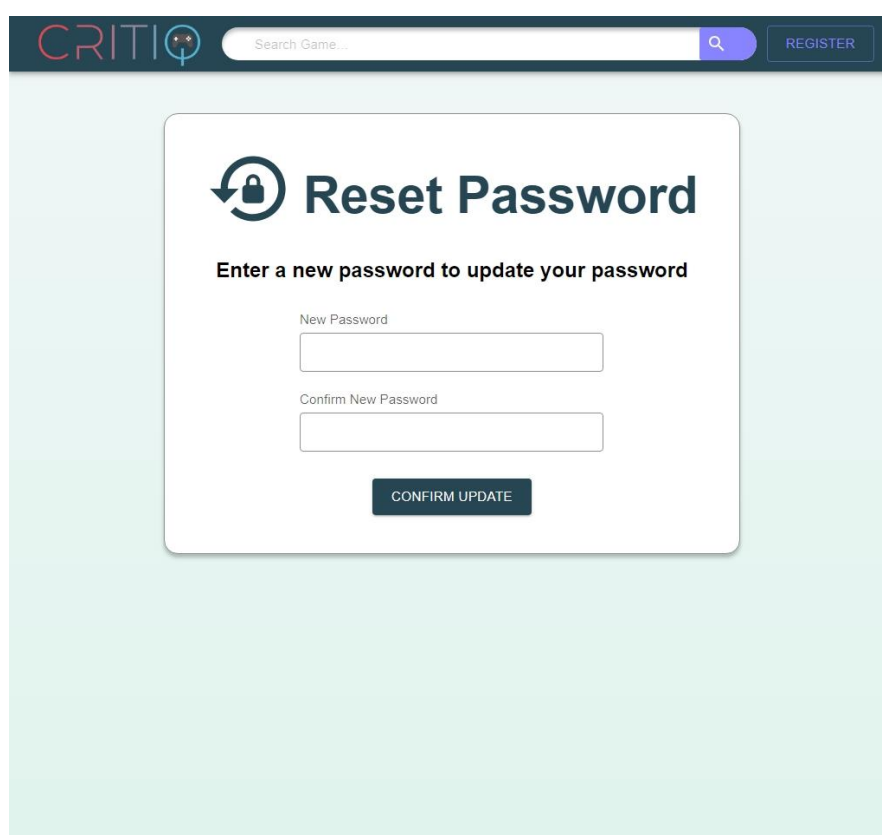


Figure 41 Reset password email

The user can access the forgot password page (See Figure 40 above) by clicking the forgot password button on the login page. On this page, the user can input the email address associated with their account. Validations will be performed by both the frontend application and the backend system. The frontend application uses regular

expressions to validate the email address format, and the backend system queries the database to check the email address existence. If the email address is valid, the system generates an email with a link to the reset password page (See Figure 41) and sends it to the user after they click the confirm button. The link contains a unique token that corresponds to a specific account, and this token is passed as a prop to the reset password page to enforce authentication and authorization. The backend system also persists the token in the database for verification purposes and sets it to expire after 2 hours for increased security. To prevent email spamming, the web application disables the confirm button for 60 seconds after sending the email.



The image shows a web application interface for resetting a password. At the top, there is a dark blue navigation bar with the 'CRITIQ' logo on the left, a search bar in the center, and a 'REGISTER' button on the right. The main content area has a light green background. In the center, there is a white rounded rectangle containing the following elements: a circular icon with a padlock and a refresh arrow, the heading 'Reset Password', the instruction 'Enter a new password to update your password', two text input fields labeled 'New Password' and 'Confirm New Password', and a dark blue button labeled 'CONFIRM UPDATE'.

Figure 42 Web application's reset password page design

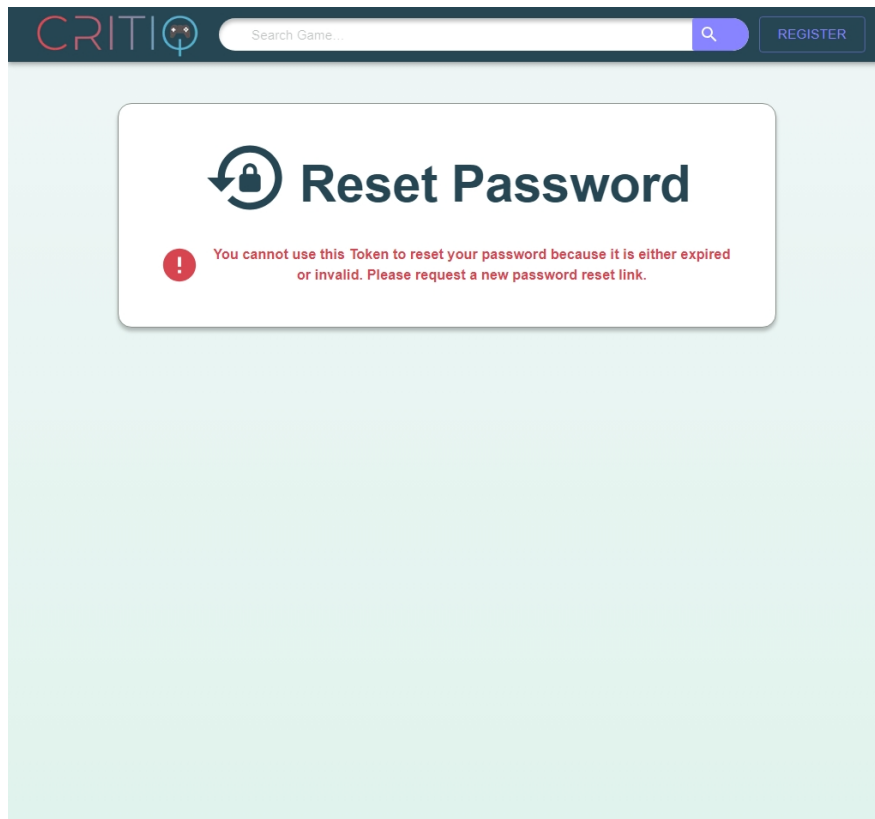
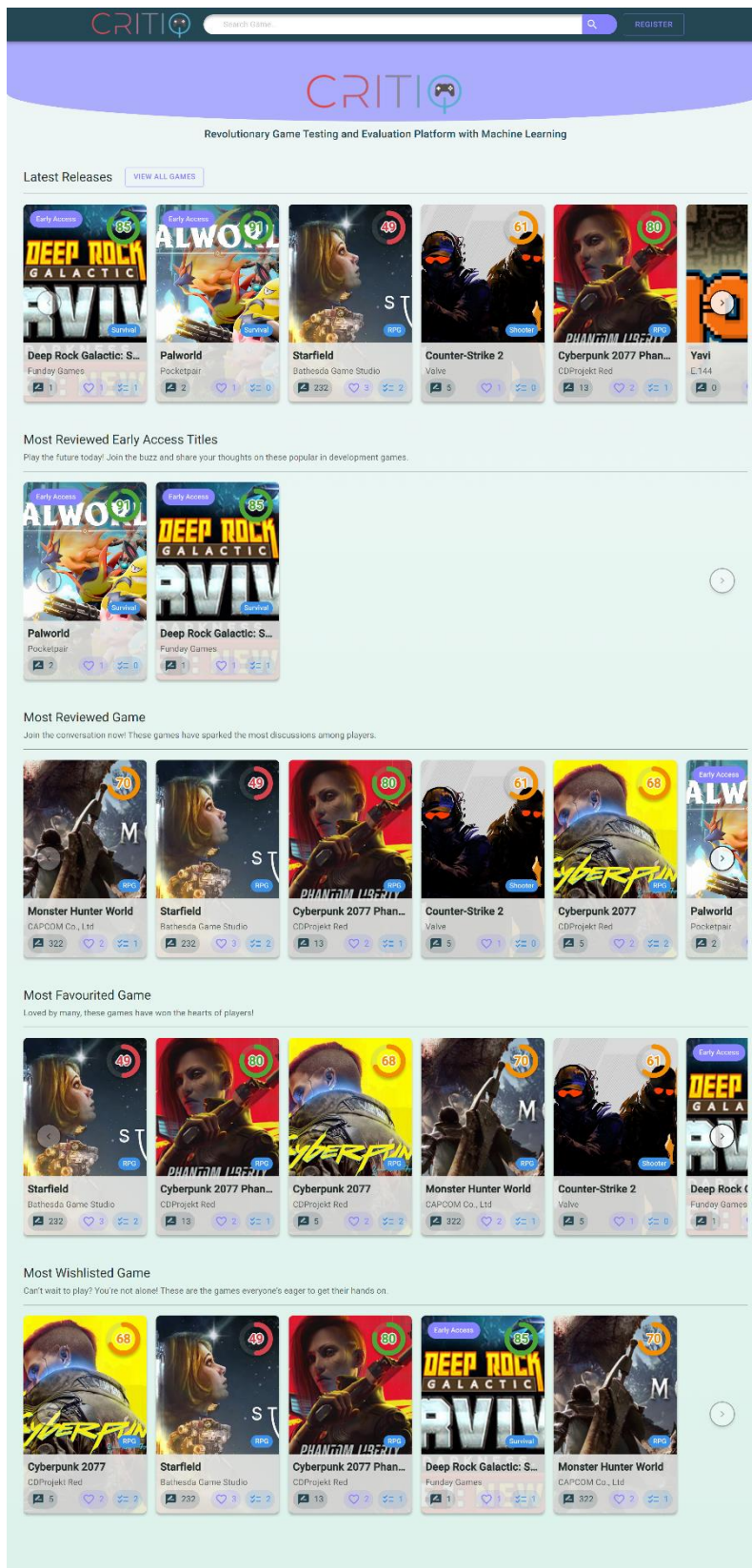


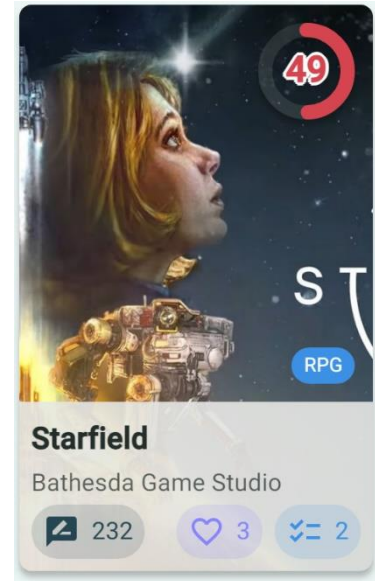
Figure 43 Web application's reset password page with invalid token

On the reset password page, the web page displays a form to reset a new password (See Figure 42) if the token in the link is valid. Otherwise, an error message is shown to inform the user of the invalid or expired token (See Figure 43 above). To update the password, the user must enter a new password and confirm it in the respective input fields in the form. The system enforces the same password rules as for registering a new account. After the user clicks the confirm update button, the web application sends the update password request to the backend system. Validation is performed by the backend system to ensure that the new password is different from the old password. If the request is invalid, the backend system will return the error to be displayed by the frontend system, otherwise, the backend will modify the password in the database.

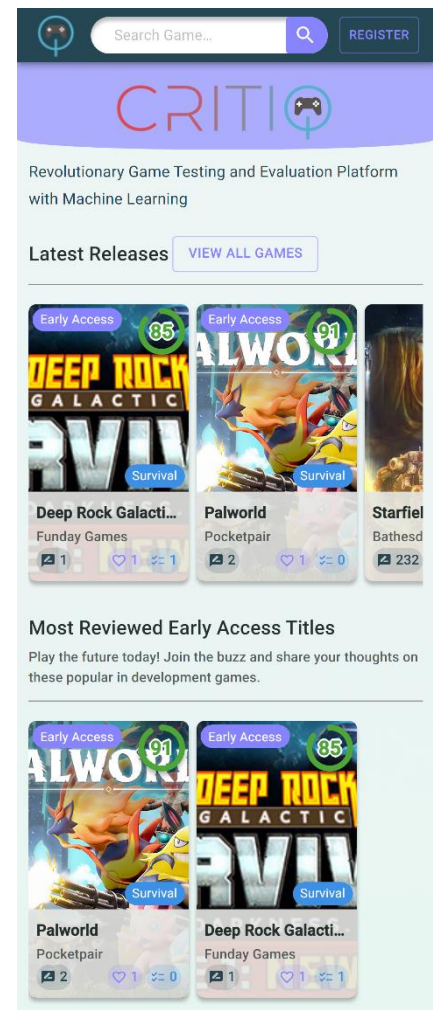
4.2.4 Landing Page



(a)



(c)



(b)

Figure 44 Web application's landing page design. (a): Landing page design for desktop viewport. (b): Landing page design for mobile viewport. (c) Game card component

Upon entering our web application, users are greeted by our landing page (See Figure 44(a) above). On the top of the page is the logo of the CritiQ platform, accompanied by a brief description of our platform below.

This page showcases five rows of game card sliders, each displaying the top 20 games based on different filtering, and ordering criteria, using the game card component. The slider is implemented using the Embla Carousel components and package, which offers advanced features such as responsive layout based on viewport and thumbnails navigation. A detailed discussion about an image slider, which utilizes this library, will be discussed when we present the review page in future section (See Section 4.2.7). Each game card outlines essential game details (See figure 44(c)). The game icon serves as the card's background. If the game is still under development, an 'Early Access' chip appears on the card's top left corner. The game's scores are displayed on the top right corner, while the game genre is displayed near the middle and right below the score. The lower segment shows the game's title, the developer's name, and three chips representing the number of reviews, favorites, and wishlists from our users. Clicking the game card components will redirect to user to the game page of that specific game.

The 'Latest Releases' row orders the games by their releases date in reverse chronological order and includes a 'View All Games' button that leads to the Search Result Page, whose design and functionality are elaborated in the subsequent section. The 'Most Reviewed Early Access Titles' row ranks in-development games by their release date. The 'Most Reviewed Game' row orders games by the number of reviews created. The 'Most Favourite Game' and 'Most Wishlisted Game' rows sort games by the number of favorites and wishlists, respectively. Through these interactive game card sliders, we aim to offer users an effortless means to discover games that suit their interest.

A mobile-responsive design has been developed for the landing page (See Figure 44(b)), which is distinct from the desktop version. Adjustments have been made to the font size, spacing between elements, and the downsizing of the game card to accommodate the limited width of a mobile screen, resulting in a more compact layout. Moreover, the navigation methods for the game card sliders are different between the desktop and mobile versions. Desktop users can navigate the slider by clicking the next and previous

buttons situated at the far right and left of the slider. On the other hand, mobile users, who typically operate a phone by scrolling, can navigate the slider by simply swiping left or right on the slider to transition between slides.

4.2.5 Search Result Page

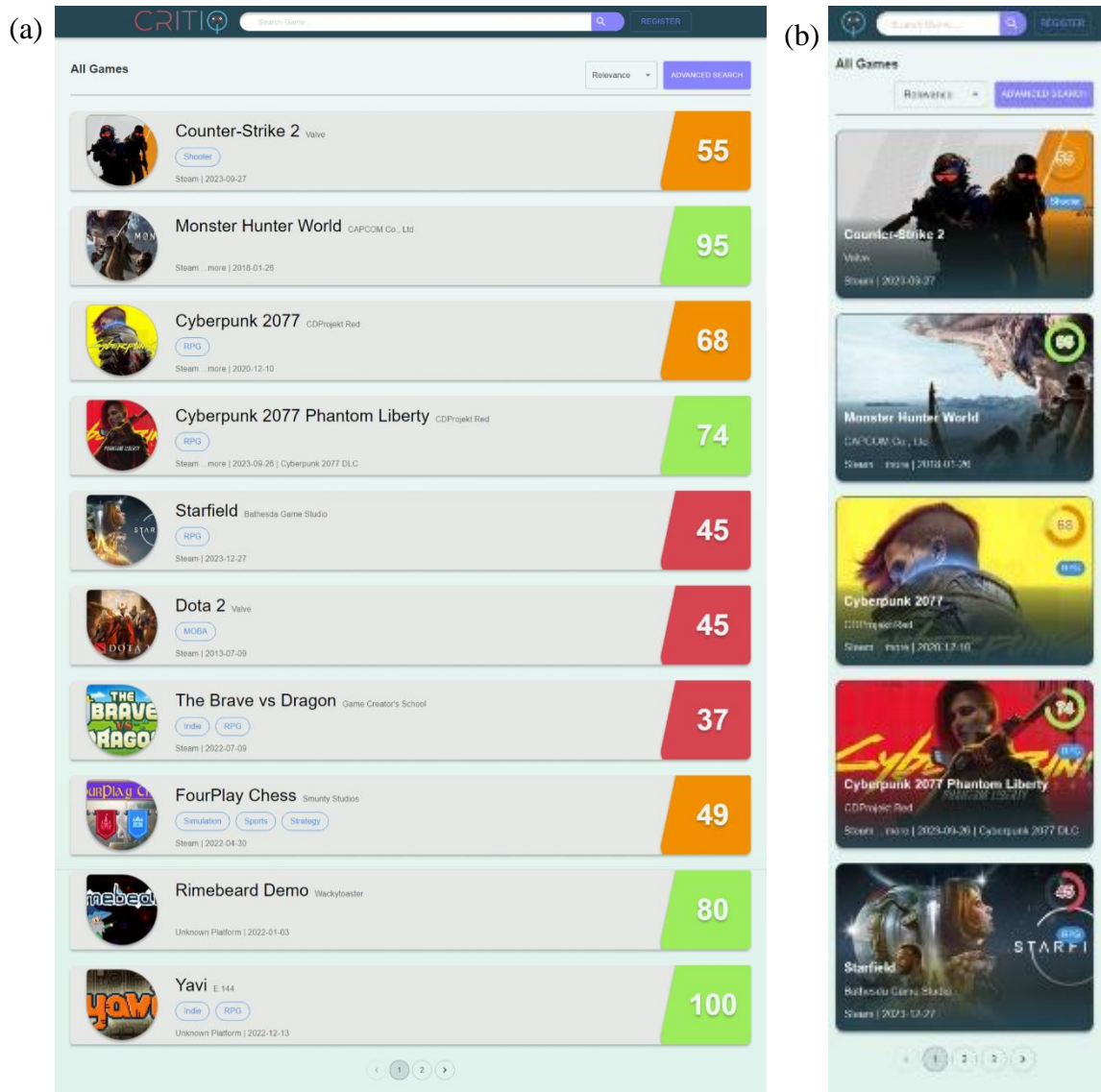


Figure 46 Web application's search result page design. (a): Search result page design for desktop viewport. (b): Search result page design for mobile viewport

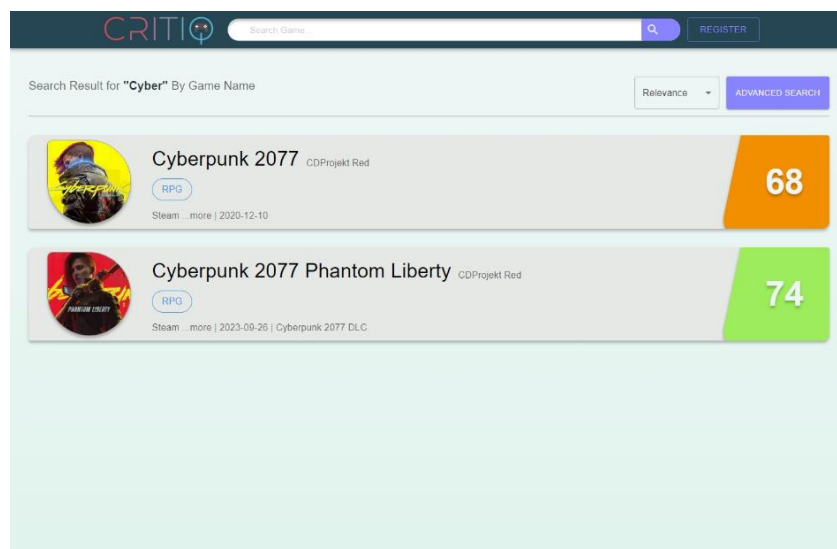


Figure 45 Search result page searching example

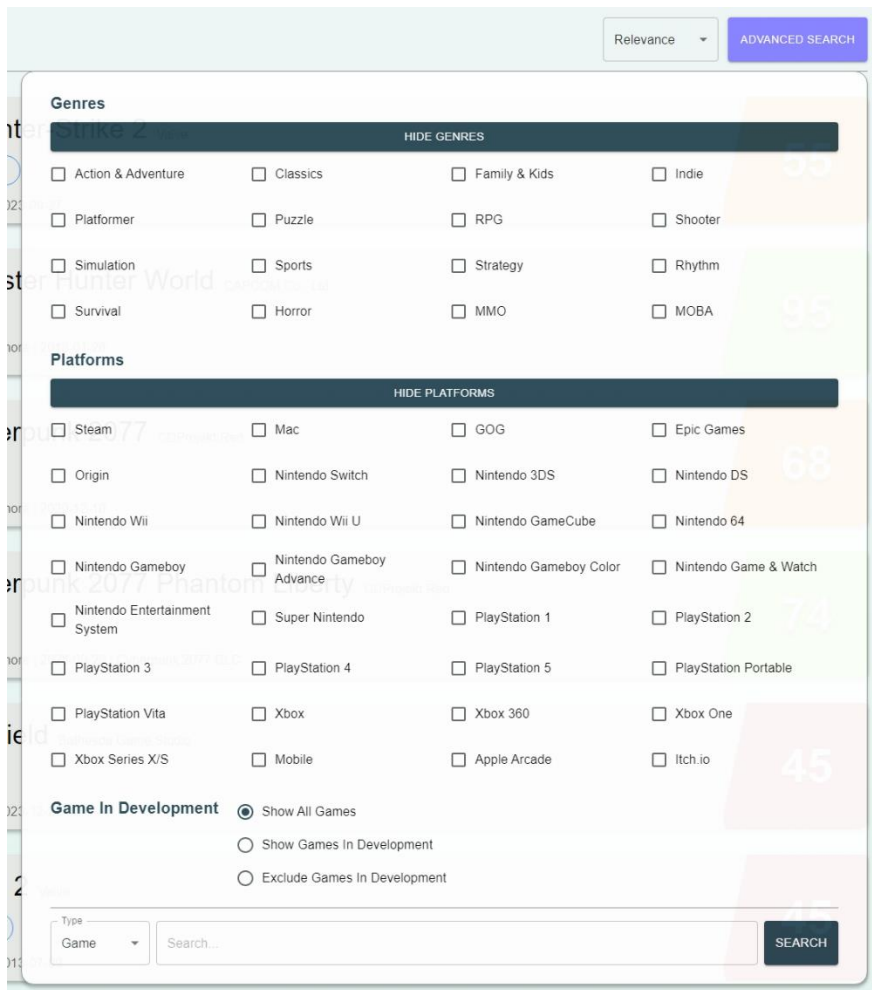


Figure 47 Advanced search modal for search result page

Section 4.2.1 and Section 4.2.4 describes how the user can access the search result page via the search bar, the search button in the toolbar and game card. The search button has a conditional functionality based on the state of the search bar. If the search bar is empty, the search button redirects the user to the search result page that displays all games in the database (See Figure 45(a) above). If the search bar has an input, the search button redirects the user to the search result page that displays the games that match the input (See Figure 46). The default search method for the search bar is search by game name.

The search result page has a description at the top that specifies the search method and the search input used to generate the results. Three different search methods are available to the user through the advanced search feature: all games search, search by title, and search by developer. The description changes accordingly to reflect the chosen search method and input.

The user can see a select button and an advance search button on the right of the description. The select button opens a menu that allows the user to choose the sorting method available for the game results: relevance, score, or release date. The advance search button opens a popup menu (See Figure 47) that enables the user to perform an advanced search by applying various filtering criteria, such as genres, platforms, and development state. The user can also choose between two search methods in this menu, which are search by title or developer.

The front-end application determines the search type and filtering criteria based on the query parameters appended to the URL and sends a search API request with the appropriate body. The query parameter **gamename** is used for search by title, while **developername** is used for search by developer. Other query parameters are **genre**, **platform**, and **isInDevelopment**, which are incorporated in the body of the API request to retrieve the filtered game results.

A URL example that searches by the developer's name "valve", with the genre of shooter, the platform of steam, and the exclusion of games in development is <https://critiq.itzjacky.info/result?developername=valve&genre=7&platform=0&isInDevelopment=false>. In this example, the genre of shooter is mapped as 7 and the platform of steam is mapped as 0 by the application.

The game search results are displayed below the description. Each search result card component displays the basic information of the game if it exists in the database. This information includes game icon, game name, developer name, game genres, game platforms, development state, and game release date. Additionally, the score of the game is also displayed, which is computed by the average score of all reviews. The game is classified as bad, average, or good based on its percentile rank among all games in the database. Games ranked above the 75th percentile are considered good, games ranked below the 30th percentile are considered bad, and games ranked between the 30th and 75th percentile are considered average. Good games are displayed with a green score, average games are displayed with an orange score, and bad games are displayed with a red score. Clicking on the search result card will redirect the user to the game page, the implementation and design of the game page will be explained in the next section.

Two distinct versions of search result cards are developed to cater to the diverse range of devices. Both versions present identical game information as mentioned before. However, the desktop version (See Figure 45(a)) is designed to accommodate the information in a horizontal layout, capitalizing on the greater width available on desktop interfaces. Taking into account the limited width of mobile viewports, the mobile version (See Figure 45(b)) adopts a more compact design with a reduced font size and a vertical composition.

The pagination at the bottom of the page allows the user to navigate through the search results. This is implemented using the Pagination component from MUI. The number of results shown on each page depends on the viewport layout. For the desktop layout, up to 10 results are shown on each page, while for the mobile layout, up to 5 results are shown on each page.

The search result page adapts to the user's viewport by displaying the layout of the page and the search result card component differently to accommodate the various screen sizes. The font size for the mobile viewport is also reduced to ensure that all information is displayed properly (See Figure 37 (a) & (b)). This design approach ensures that users will have the optimal user experience regardless of the devices they use to access the website.

4.2.6 Game Page

CRITIQ Search Game... REGISTER

FAVORITE WISHLIST ANALYTICS MORE

MONSTER HUNTER WORLD

RPG, Action & Adventure

70

Monster Hunter World

Welcome to a new world! In Monster Hunter: World, the latest installment in the series, you can enjoy the ultimate hunting experience, using everything at your disposal to hunt monsters in a new world teeming with surprises and excitement.

Platform(s): Steam, PlayStation 4, Xbox One

Released On: 2018-01-26

Developed By: CAPCOM Co., Ltd

Published By: CAPCOM Co., Ltd

Aggregated Review

Monster Hunter: World is a highly ranked game in the franchise, with an overall rating of 9.5 out of 10. It features an engaging gameplay loop with satisfying combat and loot systems, offering a wide appeal to various types of gamers. The game's narrative is built around a single-player campaign, serving primarily as a gate to unlock features and expand the map. Improved multiplayer accessibility is seen in the PC version, although the tutorial and grouping up with friends might take some time to get used to. The game's sound effects are noteworthy, and the graphics and art design are impressive, with detailed monsters, lush environments, and quality animations. The performance on PS4 Pro can reach 60fps in 4K with some settings adjusted, while the PC version has minor pop-in issues during graphically intensive cutscenes. The new verticality of levels occasionally leads to frustrating AI pathing glitches. The most mentioned topics in the reviews are "Fun Monster Hunting" and "Great Monster Hunter Game - Fun and Addictive", highlighting the game's success in delivering an enjoyable monster hunting experience. Some issues with AI pathing and end game difficulty spikes exist, but overall, Monster Hunter: World is a highly recommended game for MMO fans, action fans, RPG fans, and open-world fans who enjoy combat and loot systems.

DLC

Monster Hunter World: Iceborne

CAPCOM Co., Ltd

N/A

User Reviews (322)

Sign in to write a new review LOG IN

ALL REVIEWS POSITIVE NEGATIVE Filter Spam Recency

shawwendy4713 2024-02-26 **67**

so much to do, so engaging, fun, overall great game, can't wait for Wilds to come out

12 Days 9 Hours Played, Platform: Steam

AI Sentiment: Positive

stephensnancy8 2024-02-26 **71**

loopy star train mechanic, excessive UI, less gear options, looks nice

2 Days 2 Hours Played, Platform: Steam

AI Sentiment: Negative

candacecook343 2024-02-26 **77**

yeah

29 Days 15 Hours Played, Platform: Steam

AI Sentiment: Positive

petersfrancis6 2024-02-26 **63**

An incredible game that I discovered thanks to my friend and as well as ark mode believe it or not. I recommend this to be played by people that are ACCEPTING to be defeated from time to time and that don't have so much frustration over losing. Game itself is peak overall while I do prefer multiplayer WAY more than solo you can for sure solo this game...

2 Days 5 Hours Played, Platform: Steam

AI Sentiment: Positive

leejoel5684 2024-02-26 **92**

best monster

13 Hours 26 Minutes Played, Platform: Steam

AI Sentiment: Positive

adamkim1562 2024-02-26 **91**

yes

4 Days 14 Hours Played, Platform: Steam

AI Sentiment: Positive

gabriele00800 2024-02-26 **86**

The current state of the game is very good. Plenty of community QOL mods available to fix most issues you may have with the game. POSITIVES - The gameplay is hard to master and each weapon has its own identity. The environments are beautiful with a lot of different creatures to discover. The character customization is really good. NEGATIVES -

1 Day 21 Hours Played, Platform: Steam

AI Sentiment: Positive

scott861249 2024-02-26 **89**

Came back after several years playing on Console. The framerate and graphics update is beyond anything I could've hoped for. Playing through with a whole new weapon and it everything feels fresh once again. Highly recommended.

2 Days 11 Hours Played, Platform: Steam

AI Sentiment: Positive

markmartin6734 2024-02-26 **82**

Such a good game. As with the other Monster Hunter games, the story is only mildly interesting, but that's all it needs to be. The main focus is of course fighting the monsters and exploring the environments, and it is such a joy to do so. Originally I got this game at launch on Xbox as a birthday present and played through the whole thing and then later.

1 Day 19 Hours Played, Platform: Steam

AI Sentiment: Positive

duncanwalter76 2024-02-26 **56**

I'm not sure how to even describe this game. I would say I'm primarily a PvE player. The most hours I've put into games in the last couple of years are PvE and recently SO3. I've got maybe 2-300 hours in SO3 and probably around 900-1k hours in PvE. I consider PvE my favorite game. But then I had the pleasure of finally playing MHW. I thought it back...

8 Days 18 Hours Played, Platform: Steam

AI Sentiment: Positive

zgalloway8342 2024-02-26 **75**

monster hunt

2 Days 6 Hours Played, Platform: Steam

AI Sentiment: Negative

VIEW MORE REVIEWS

(a)

Search Game... REGISTER

RPG, Action & Adventure ANALYTICS MORE

MONSTER HUNTER WORLD

RPG, Action & Adventure

70

Monster Hunter World

FAVORITE WISHLIST

Welcome to a new world! In Monster Hunter: World, the latest installment in the series, you can enjoy the ultimate hunting experience, using everything at your disposal to hunt monsters in a new world teeming with surprises and excitement.

Platform(s): Steam, PlayStation 4, Xbox One

Released On: 2018-01-26

Developed By: CAPCOM Co., Ltd

Published By: CAPCOM Co., Ltd

Aggregated Review

Monster Hunter: World is a highly praised game in the franchise, with an overall rating of 9.5 out of 10. It features an engaging gameplay loop with satisfying combat and loot systems, offering a wide appeal to various types of gamers. The game's narrative is built around a single-player campaign, serving primarily as a gate to unlock features and expand the map. Improved multiplayer accessibility is seen in the PC version, although the tutorial and grouping up with friends might take some time to get used to. The game's sound effects are noteworthy, and the graphics and art design are impressive, with detailed monsters, lush environments, and quality animations. The performance on PS4 Pro can reach 60fps in 4K with some settings adjusted, while the PC version has minor pop-in issues during graphically intensive cutscenes. The new verticality of levels occasionally leads to frustrating AI pathing glitches. The most mentioned topics in the reviews are "Fun Monster Hunting" and "Great Monster Hunter Game - Fun and Addictive", highlighting the game's success in delivering an enjoyable monster hunting experience. Some issues with AI pathing and end-game difficulty spikes exist, but overall, Monster Hunter: World is a highly recommended game for MMO fans, action fans, RPG fans, and open-world fans who enjoy combat and loot systems.

DLC

Monster Hunter World: Iceborne

CAPCOM Co., Ltd

N/A

User Reviews (322)

Sign in to write a new review LOG IN

ALL REVIEWS POSITIVE NEGATIVE Filter Spam Recency

shawwendy4713 2024-02-26 **67**

so much to do, so engaging, fun, overall great game, can't wait for Wilds to come out

12 Days 9 Hours Played, Platform: Steam

AI Sentiment: Positive

(b)

Figure 48 Web application's game page design. (a): Game page design for desktop viewport. (b): Game page design for mobile viewport.

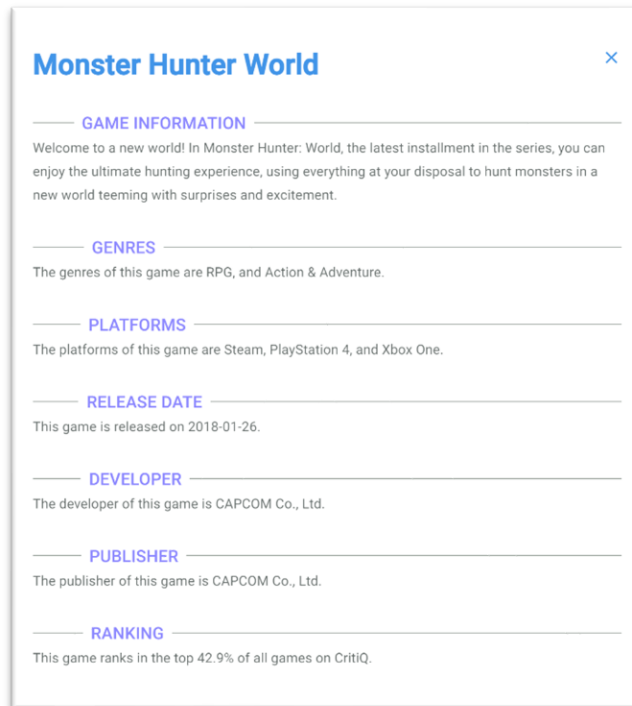


Figure 49 Game detailed information popup modal

As stated in the previous sections, the game page (See Figure 48 above) can be accessed by selecting the search result card on the search result page, or by clicking on the game card in the landing page. This page includes four main sections, which are Information, Aggregated Review, Add Review and All reviews.

Firstly, the information section is located at the top of the game page, it displays all the game information that was previously presented on the search result card. On the top of this section is a container with the game icon as the background. To access the full description of the game, the user can click on the “More” button located on the top right of this container, which will open the game detailed information modal (See Figure 49). On top of the “More” button is the “Analytics” button, clicking this button will redirect the user to the game analytics page. The game analytic page will be discussed in the future section (See Section 4.2.10). For the desktop version (See Figure 48(a)), two buttons are positioned next to the “More” button on the left side, which are the “Favourite” and “Wishlist” Buttons, for user who have signed into their account, they can favorite or wishlist a game using these buttons, and the favorite and wishlist data will be collected and used for the game analytic page to help the developer to analyze the game. Below the “More” button is a blue chip displaying the genres of the game. For the Mobile version (See Figure 48 (b)), the layout is altered, where the Favourite”

and “Wishlist” Buttons and game genres chip are positioned differently to accommodate the limited width of the mobile phone. Moreover, below the icon container description box, it provides a brief overview of the game, the name of the publisher, and a comprehensive list of platforms that support the game. The brief overview is restricted to three lines for desktop and five lines for mobile, and the overflow text will be concealed by ellipsis.

The information section is followed by the aggregated review section and the DLC section. The aggregated review section utilizes LLMs with prompt engineering to generate an aggregated review that summarizes all the reviews of the game made by our platform users, and display the summary information on this section. The DLC section will only be visible for games that have DLC, and all the DLCs will be presented in a slider format with an individual DLC card. The DLC card shows the name, developer, the release date, and the score of the LDC, clicking on the card will redirect the user to the game page of the DLC. The slider is implemented using the React Slick library, a popular React carousel that offers a simple, lightweight, and customizable carousel component.

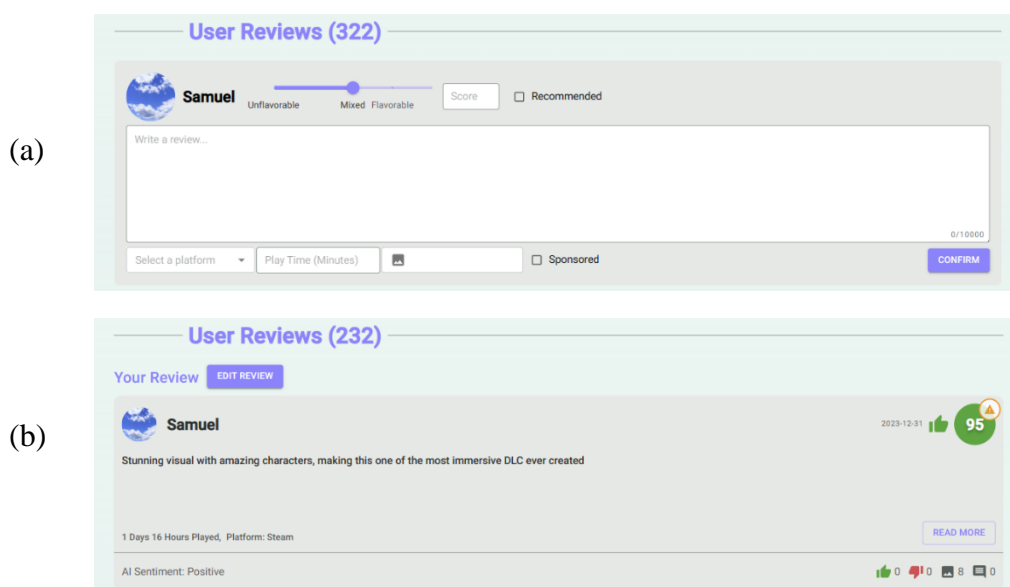


Figure 50 Add review section design. (a): Implementation for authenticated user that have not made a review to the game. (b): Implementation for authenticated user that made a review to the game

Following the aggregated review section is the add review section. Unauthenticated users will be prompted to log in before they can create a review. Otherwise, users can create new reviews by completing the input fields in the add review form (See Figure 50(a)). They are required to provide mandatory information, including a numerical score (from 0 to 100), the recommendation status, the source of the game (self-bought

or sponsored), the review content, the platform, and the total playtime. Additionally, users can optionally attach images to their reviews through the file input field, this is implemented using the MUI file input component from the MUI file input library. The submitted images are limited by quantities and size, with a maximum of 10 images and a size of less than 3 Megabytes per image. This constraint aims to prevent users from uploading excessive number of high-quality images to our storage bucket and hindering the performance of page load. When the user clicks on the confirm button, both the frontend and backend perform validations to ensure that all required fields are filled, and the attached images do not surpass the size limit. If the new review is successfully created, the web application will redirect the user to the review page.

For users who have already made a review for this game, this section will display their review (See Figure 50(b)). Furthermore, they can modify their review by clicking on the “Edit Review” button, which brings up the add review form pre-filled with their previous review details, allowing them to make necessary changes. However, for the sake of simplicity, users are not permitted to change the platform and images in their review. To prevent spam, users are limited to editing their review once per week, as each update triggers a regeneration of the automated review analysis. The system checks the time both on the frontend and backend, and the frontend will display the remaining time before the user is eligible to edit their review again.

Finally, the game review section exhibits the reviews that other users have written for the game. A tab bar for filtering and a select button for sorting are located at the top. Users can choose to display based on the review sentiment values by selecting the respective tab in the tab bar. They can also filter out possible review spam by checking the filter spam check box next to the tab bar. Clicking the select button opens a menu for users to select the sorting criterion for the reviews, which comprises recency or score.



Figure 51 Game review card design. (a): Game review card under desktop viewport. (b): Game review card under mobile viewport

The review information is displayed using the game review card component, it presents the essential information about the game review and the reviewers (See Figure 51(a) above).

A mobile version with slight changes to the layout, icon size and font size is implemented to reduce the width of the card (See Figure 51(b)). The cards are situated below the tab bar. The first row in the game review card displays the avatar icon and name of the reviewers, the date and time the game review was created, a thumb up or thumb down icon indicating the recommendation status of the review, and the score that the reviewer assigned to the game. The score color is computed dynamically: scores above 75 are deemed as good and shown in green, scores below 50 are deemed as bad and shown in red, and scores between 50 and 75 are deemed as average and shown in orange. The second row shows the review content, which is limited to four lines and the excess content is concealed by ellipses. The left side of the third row reveals the total playtime, the platform, and the game version that the reviewer played the game on, and the right side has the read more button, which will redirect the user to the review page upon clicking. The review page will be explained in detail in the subsequent section. Lastly, the fourth row displays the sentiment result of our NLP model on the left, and the number of likes, dislikes, images, and review comments on the right. We used LLMs and prompt engineering to analyze each individual review. The model identifies possible review spam, and places an orange alert on the top right the corner accordingly to alert the user.

Moreover, the game review cards are structured using the Grid component from MUI, which enables the adjustment of the card per row according to the viewports. The desktop layout displays two game review cards per row, while the mobile layout shows

one game review card per row. The game page will only show 12 reviews at most, if more than 12 reviews are created, the “View More Reviews” button will be displayed on the bottom on the game page. Clicking this button redirects the user to the Game Review Page, which will be discussed in the next section.

4.2.7 Game Reviews Page

322 Reviews for **MONSTER HUNTER WORLD**

Recency [ADD REVIEW](#)

ALL REVIEWS POSITIVE NEGATIVE

Filter Spam

shawwendy4713 2024-02-26 **67**
so much to do, so engaging, fun, overall great game, can't wait for Wilds to come out
12 Days 9 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

stephensnancy8 2024-02-26 **71**
lousy roar stun mechanic, excessive ui, lifeless gear options, looks nice
2 Days 3 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Negative

candacecook343 2024-02-26 **77**
yeah
26 Days 15 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

petersfrancis6 2024-02-26 **63**
An incredible game that I discovered thanks to my friend and aswell as ark mods believe it or not. I recommend this to be played by people that are ACCEPTING to be defeated from time to time and that dont have as much frustration over losing. Game itself is peak overall While i do prefer multiplayer WAY more than solo you can for sure solo this game...
2 Days 9 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

leejoel5684 2024-02-26 **92**
hunt monster
13 Hours 26 Minutes Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

adamkim1562 2024-02-26 **91**
yes
9 Days 16 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

gabriele00800 2024-02-26 **86**
The current state of the game is very good. Plenty of community QOL mods available to fix most issues you may have with the game. POSITIVES: The gameplay is hard to master and there each weapon has its own identity - The environments are beautiful with alot of different creatures to discover - The character customization is really good. NEGATIVES...
1 Days 21 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

kevinmoss3158 2024-02-26 **82**
Highly recommend this game. Its challenging but rewarding. You easily lose hours playing without you realizing it.
8 Days 14 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

scott861249 2024-02-26 **89**
Came back after several years playing on Console. The framerate and graphics update is beyond anything I could've hoped for. Playing through with a whole new weapon and it everything feels fresh once again. Highly recommended.
2 Days 11 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

markmartin6734 2024-02-26 **82**
Such a good game. As with the other Monster Hunter games, the story is only mildly interesting, but that's all it needs to be. The main focus is of course fighting the monsters and exploring the environments, and it is such a joy to do so. Originally I got this game at launch on Xbox as a birthday present and played through the whole thing (and then later...
1 Days 18 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

duncanwalter76 2024-02-26 **56**
I'm not sure how to even describe this game. I would say I'm primarily a PoE player. The most hours I've put into games in the last couple of years are PoE and recently BG3. I've got maybe 2-300 hours in BG3 and probably around 900-1k hours in PoE. I consider PoE my favorite game. But then I had the pleasure of finally playing MHW. I bought it back in...
8 Days 18 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

zgalloway8342 2024-02-26 **75**
mointer hunt
2 Days 8 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Negative

< 1 2 3 4 5 - 27 >

(a)

322 Reviews for **MONSTER HUNTER WORLD**

Recency [ADD REVIEW](#)

ALL REVIEWS POSITIVE NEGATIVE

Filter Spam

shawwendy4713 2024-02-26 **67**
so much to do, so engaging, fun, overall great game, can't wait for Wilds to come out
12 Days 9 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Positive

stephensnancy8 2024-02-26 **71**
lousy roar stun mechanic, excessive ui, lifeless gear options, looks nice
2 Days 3 Hours Played, Platform: Steam [READ MORE](#)
AI Sentiment: Negative

(b)

Figure 52 Web application's game reviews page design. (a): Game reviews page design for desktop viewport. (b): Game reviews page design for mobile viewport

As mentioned in the previous section, The game reviews page can be accessed by clicking on the “View More Reviews” button in the game page. This page displays the game reviews in the same way as the game page with the sentiment and spam filtering, and sorting by recency or score (See Figure 52 above). The creation of this page aims to declutter the game page, which is already information-dense. A pagination feature is included at the bottom of this page for easy review navigation, implemented in the same manner as on the game result page (See Section 4.2.5).

4.2.8 Review Page

The desktop review page for Palworld features a clean, structured layout. At the top, the Critic logo and search bar are visible. The review title 'Palworld' is prominently displayed, followed by the reviewer's name 'SAMUEL'. The reviewer's profile, including a 'Recommended' badge and a score of 89, is shown. The review text is organized into sections: AI Sentiment (Positive), Main Topics (Friendly Fun), Key Words (Graphics & Art Design, Price, Gameplay, Accessibility, Performance, Sound, Bug, Narrative, Suggestion, Overall), and a Summary. A large, vibrant screenshot of the game is featured, with a 'VIEW IMAGE' button and a '1' indicator. Below the screenshot, there are thumbs-up and thumbs-down buttons for feedback. A comment section at the bottom shows one comment from Samuel2, dated 2024-04-03 17:56 PM.

(a)

The mobile review page for Palworld is a condensed version of the desktop page. It maintains the same core information but with a more compact layout. The reviewer's name 'Samuel' and score '89' are clearly visible. The review text is shortened to fit the mobile screen, with key sections like AI Sentiment, Main Topics, Key Words, and Summary being present but more concise. The game screenshot is also smaller and includes a 'VIEW IMAGE' button. The feedback buttons and comment section are also adapted for the mobile format.

(b)

Figure 53 Web application's review page design, review context has been shortened to reduce the length of the screenshots. (a): Review page design for desktop viewport. (b): Review page design for mobile viewport.

This section presents the review page, which was mentioned in the previous sections. The review page can be accessed by clicking on the read more button on the game review card and consists of three sections: the header section, the review body section, and the review comment section (See Figure 53(a) above). This page implements RWD. The layout for the mobile version is most similar to the desktop version. However, modifications have been made to adapt to the narrower width of mobile devices. These include reducing the font size, adjusting the spacing between elements, and reordering components (See Figure 53(b)).

The header section, located at the top of the page, displays the game icon and a location-based breadcrumb navigation. The breadcrumb navigation allows the user to return to the previous level in the website's hierarchy, which is the game page, by clicking on the game icon or the game name. The Breadcrumbs component from MUI is used to implement the breadcrumb navigation.

The review body section, which follows the header, comprises four sub-sections.

The first sub-section displays some basic information about the review and the reviewer. On the left, it shows the avatar icon and the name of the reviewer, the creation time of the review, the platform, the total play time, and the version of the game that the reviewer played. On the right, it shows the recommendation status of the review and the score that the reviewer assigned to the game.

The second sub-section consists of the review analysis information. Users can expand or collapse this section by clicking on the arrow button situated at the center of the section's bottom. The content within this section is automatically generated using Natural Language Processing (NLP) models. The analysis includes sentiment, topics, keywords, and a summary. The sentiment, which can be either positive or negative, is determined using a sentiment analysis model, aiding in understanding the overall tone of the reviews. The primary topics are identified using Topic Modelling and LLMs, identifying prevalent themes or subjects in the reviews. Keywords are extracted using keyword extraction techniques combined with LLMs and prompt engineering according to different predefined aspects. These aspects, with positive or negative keywords, are marked with green or red, respectively. This identifies the most commonly used words or phrases in the reviews according to different aspects. The summary is produced using LLMs and prompt engineering, where a summary will only

be generated if the review contains more than 50 words. With this review analysis information, we hope to provide a concise yet comprehensive overview of the review and assist in reducing the time required for developers to analyze the review, enabling them to devote more resources to enhance their game and create games that better align with player expectations and preferences. The information aids the developer in understanding what players like or dislike about different aspects of the game and make informed decisions to enhance their game.

The third section presents the review context and attached images. The section displays the review text body and an image slider that shows the images associated with this review. The image slider is implemented using the Embla Carousel component, which was described on the landing page (See Section 4.2.4). The height of slider is reduced for the mobile viewport to avoid the slider from taking up the entire screen. The user can navigate the images in various ways, such as dragging on the image, clicking the left and right arrow buttons, and clicking on the individual thumbnail below the slider. The final section allows the user to evaluate the review by clicking on the like or dislike buttons, depending on whether they found the review helpful or not. The buttons and the game review card show the number of likes and dislikes, which provide an initial impression of the review to other users.

The final section of the review page is the comment section, which appears below the review body section. This section allows the users to add new comments and interact with other users about the review. The section displays the total number of comments at the top, followed by the comment box, which is only visible to logged-in users. The comment box shows the user's avatar and a text input field for entering the comment. The user can submit the comment by pressing enter while typing. The comments of other users are shown below the comment box, sorted by the creation time in ascending order. The user can browse the comments using the pagination at the bottom, which is implemented in the same way as the search result page (See Section 4.2.5).

4.2.9 Game Analytics Page

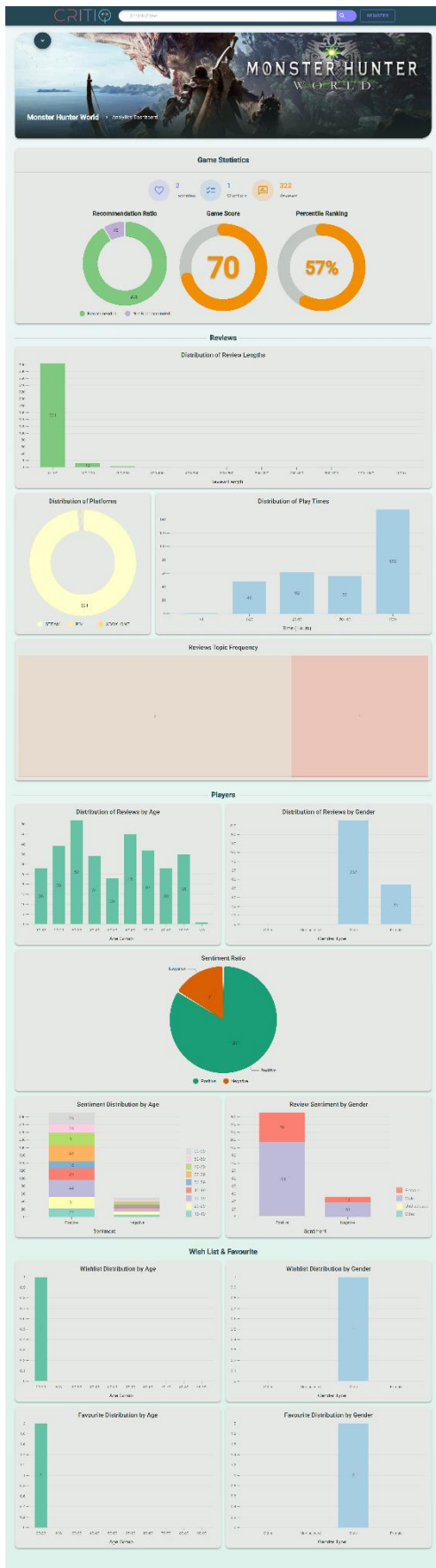


Figure 54 Game analytics page design for desktop viewport

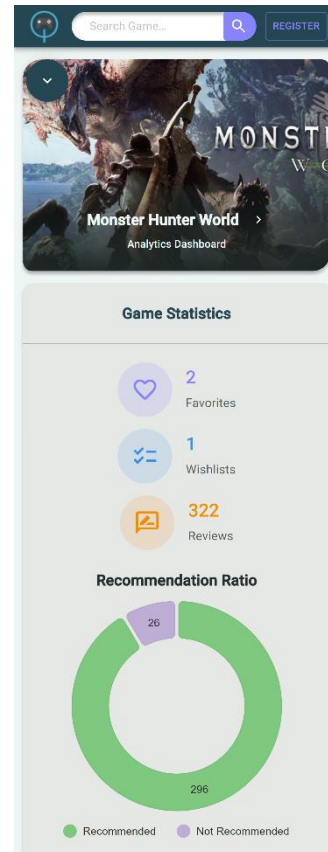


Figure 55 Game analytics page design for mobile viewport. The image is cropped as the page is too long to display

This section outlines the design and implementation of the game analytics page. As referenced in section 4.2.6, users can navigate to the game analytics page by clicking the “Analytics” button on the game page. When this page is accessed by a user, an API is triggered to fetch the analytical data from the backend, which is compiled from the existing data from our users and reviews. We employ an on-demand processing approach, as opposed to batch processing, for managing the analytical data due to its superior efficiency and lower cost. Consequently, we evaluate the timestamp of the last update to the analytical data. If more than 24 hours have passed since the last update, the analytical data will be refreshed when a user visits the page and triggers the API.

The game analytics page is divided into five sections: the game header, game statistics, reviews, players, and wishlist and favorite (See Figure 54 above). This page also implements RWD. In the mobile view, due to the restricted width, all charts and components will occupy the full row (See Figure 55). The layout of the charts and components and implemented using the Grid component from MUI. This component was discussed in Section 4.2.6.

Firstly, the game header shows the name of the game, its icon, and the name of the current page. It uses the game icon as its background. It incorporates breadcrumb navigation similar to the one implemented on the review page (refer to Section 4.2.8). Within the breadcrumb navigation, there is a button labeled with the game’s name. Clicking this button will navigate the user back to the game page. Additionally, there is a typography component present to signify that this page serves as the analytics dashboard for the game.

Secondly, the game statistics section displays the basic analytics information regarding the game, which includes the number of reviews, favorites and wishlist for the game. It features a pie chart of recommendation ratio that illustrates the proportion of user reviews that recommend the game versus those that do not. This is implemented using the ResponsivePie component from nivo, a renowned React library offering a variety of data visualization components such as charts and maps. Additionally, this section presents the game’s score and its percentile ranking compared to all games in the platform’s database, implemented using the CircularProgress component from MUI.

Thirdly, the reviews section visualizes the reviews related data through charts. It comprises three bar charts and a tree map. The charts depict the distribution of review lengths, the distribution across platforms, and the distribution of play times. This data is retrieved from the reviews submitted by users on our platform. The charts are constructed using the ResponsiveBar component from nivo. The purpose of these bar charts is to understand if users are providing detailed feedback or just brief comments, identify which platforms are most popular for this game and indicate if the game is engaging enough to hold the players' interest for extended periods. Additionally, the tree map showcases the frequency of topics generated for the reviews through topic modelling, with a maximum limit of 10 topics. It is created using the ResponsiveTreeMap component from nivo. The purpose of the tree map is to help game developer to identify common theme and the most frequently discussed topics in the reviews, understand what players like and dislike about the game, and prioritizing their development resources accordingly to improve the frequently discussed topic.

Fourthly, the players section provides analytics information regarding the reviewer of the game. It comprises four bar charts and one pie chart. The four bar graphs represent the distribution of reviews by age and gender, as well as the sentiment distribution of reviews by age and gender. These bar charts are created using the ResponsiveBar component from nivo. The pie chart, which is made using the ResponsivePie component from nivo, represents the ratio of review sentiments. The data for age and gender are obtained from our users, as these details are mandatory when registering a new account. The sentiment data is produced by our sentiment analysis model. Moreover, in the bar charts for sentiment distribution by age and gender, users have the option to include or exclude specific age groups or genders to analyze sentiment data according to various user groups and target audiences.

Finally, the wishlist and favorite section showcases the wishlist and favorite data of the game. As mentioned in the game page, authenticated users can add a game to their favorites or wishlist via the provided buttons. This section includes four bar charts representing the wishlist distribution by age and gender, and the favorite distribution of reviews by age and gender. These charts aim to understand the preferences of different demographic groups, thereby directing marketing efforts towards the interested audience. Furthermore, developers can customize their game to cater more to

demographics that show higher engagement with the game. If a particular demographic group exhibits increased interest in the game, it indicates the success of marketing strategies or game features that resonate with those groups.

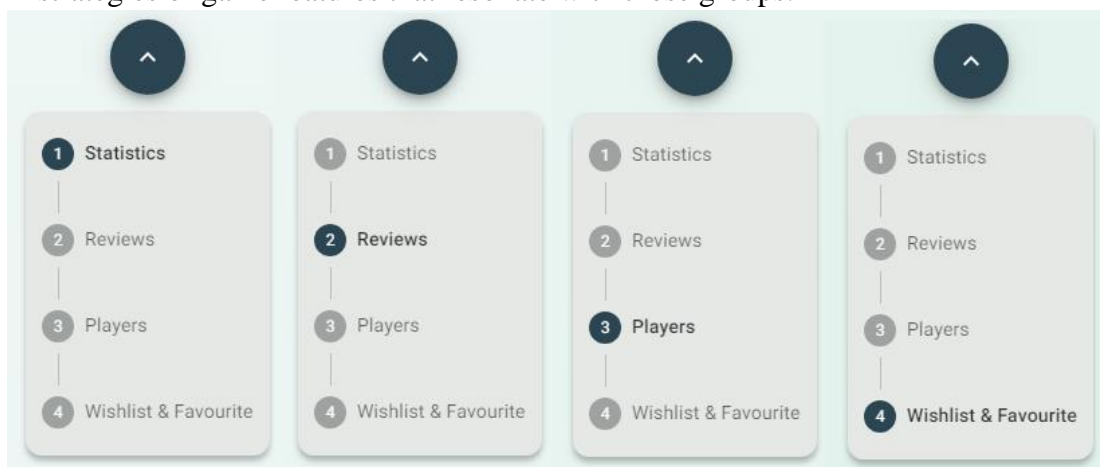


Figure 56 Game Analytics Page sections tracking feature

Furthermore, a navigation bar has been incorporated into the game analytics page to provide the users a convenient way to move between different sections of the page. Users can select a specific section from the content table, which will then scroll the page to that section. Moreover, as users scroll through the sections, the content table will highlight the current section the user is viewing by showing that section as active (See Figure 56 above). The navigation bar utilizes the Stepper components from MUI, while the content table is made up of the Step, StepButton, and StepLabel components from MUI. The section tracking feature is implemented using the intersection observer API and the IntersectionObserver interface, allowing for asynchronous observation of changes in the intersection of a target element with an ancestor element or with the viewport of the top-level document. The active Step is set whenever the user's screen interacts with a specific section on the page.

4.2.10 Profile Page

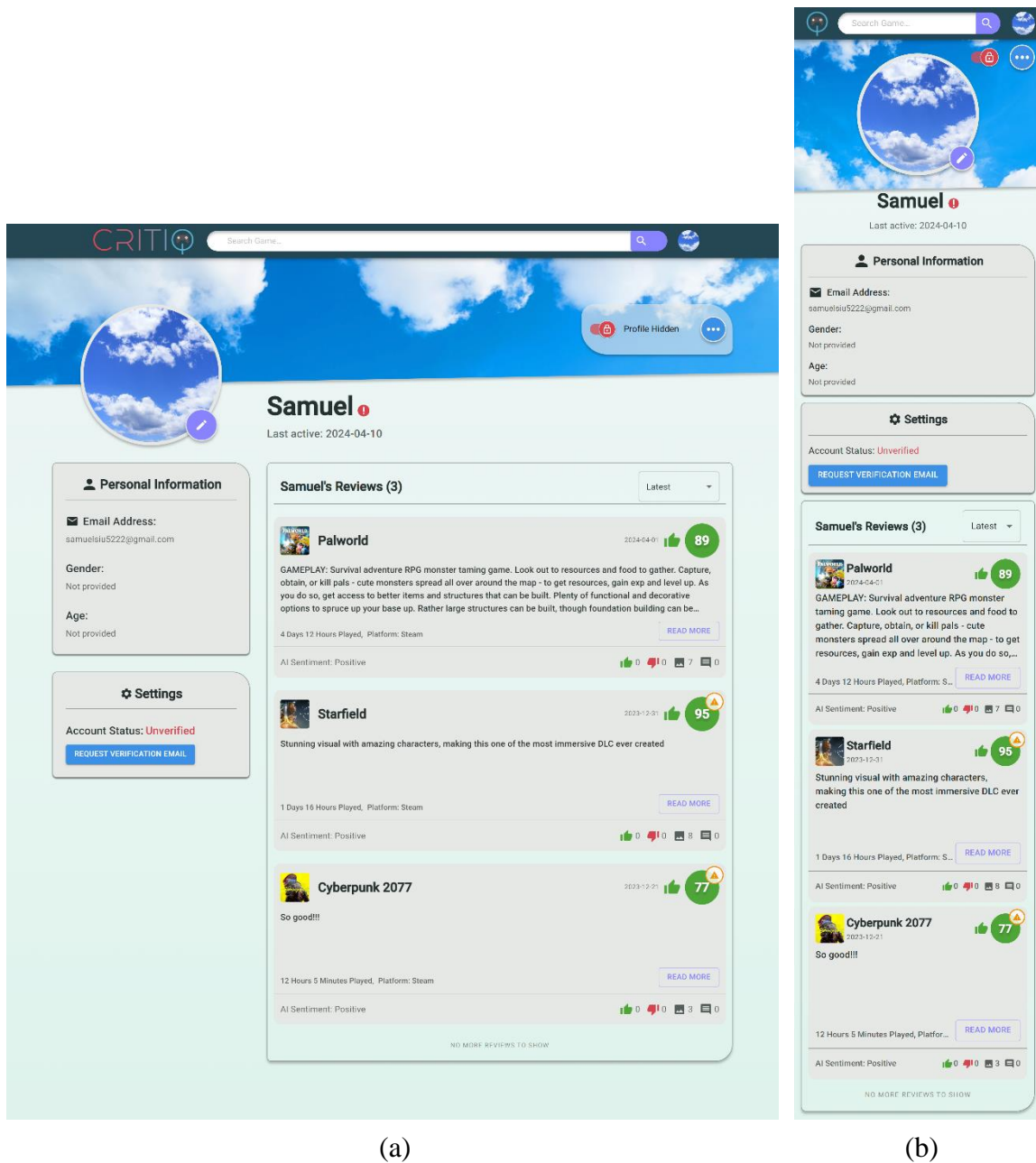


Figure 57 Web application's profile page design under the owner view. (a): Profile page design for desktop viewport. (b): Profile page design for mobile viewport.

As outlined in Section 4.2.1, users can navigate to their own profile page by selecting the profile button located in the toolbar menu. If they desire to view another user's profile page, they can do so by clicking on the user's avatar icon or name, which can be found on the review page or within a review card.

There are three major sections in the profile page (See Figure 57(a) above). For mobile viewports, the layout and spacing are adjusted to suit mobile devices. Some components, like the profile privacy switcher, are simplified to ensure that no component overflows or overlaps with another (See Figure 57(b)).

The profile page will be explained with the desktop version for the sake of simplicity. On the top of the profile page is the user banner. If the profile page is under the owner's view, they will also see a control panel on the right side, with the profile privacy switcher and a button to open a menu to update username of their profile banner (See Figure 60). Clicking on the switch will switch the profile page between public and private. Selecting the button on the menu will open the respective modals for the user to update username or their profile banner (See Figure 58 and 59 below). The username update follows the same rules as for registering an account.

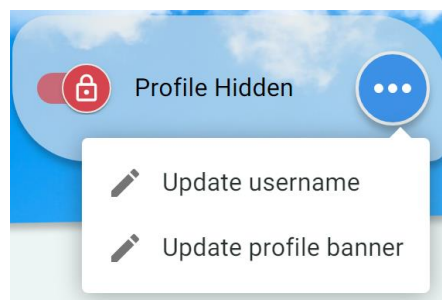


Figure 60 Profile page control panel for the profile owner

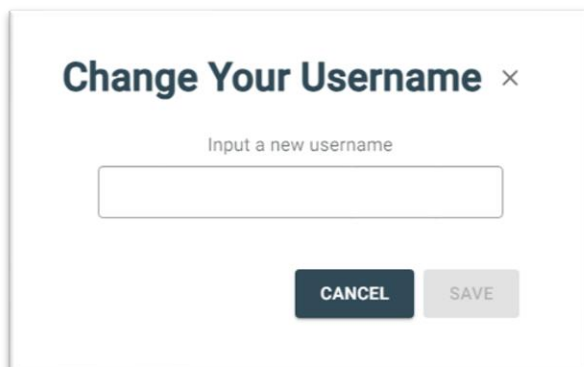


Figure 58 Update profile banner modal

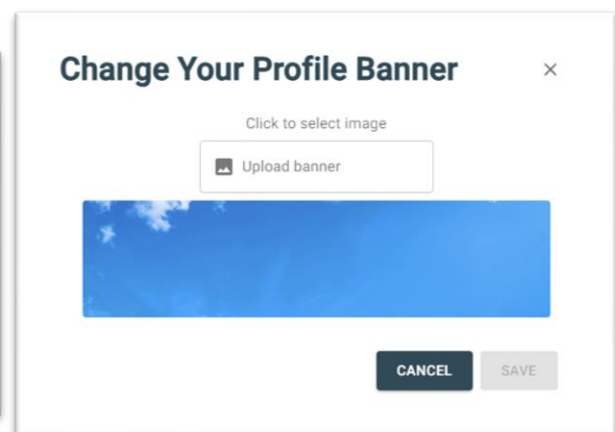


Figure 59 Update username modal

Next, on the left-hand side of the profile page, you'll find three sub-sections, which are the user avatar box, the personal information box, and the settings box. In the desktop viewport, this section is assigned a sticky position in the CSS. This means that the section will remain affixed to the top of the page and will not be scrolled past by the user, ensuring its constant visibility.

In the user avatar box, for the profile owner, a small purple edit icon will appear at the bottom right of the user avatar. Clicking this icon opens the 'Update User Avatar' modal (See Figure 61(a)). After an image is uploaded via the file input bar, an interface

backdrop featuring the avatar editor will appear. Here, users can crop, rotate, and set the image as their avatar icon (See Figure 61 (b)). The updated avatar icon will then be displayed on the modal, and users can click ‘Save’ to confirm the avatar changes (See Figure 61 (c)).

Directly below the user avatar box is the personal information box, which displays the email address, gender, and age of the user. Lastly, the settings box is situated beneath the personal information box and is only visible to the profile owner. This box displays the account verification status and includes a button to resend the verification email to your email address every 60 seconds.

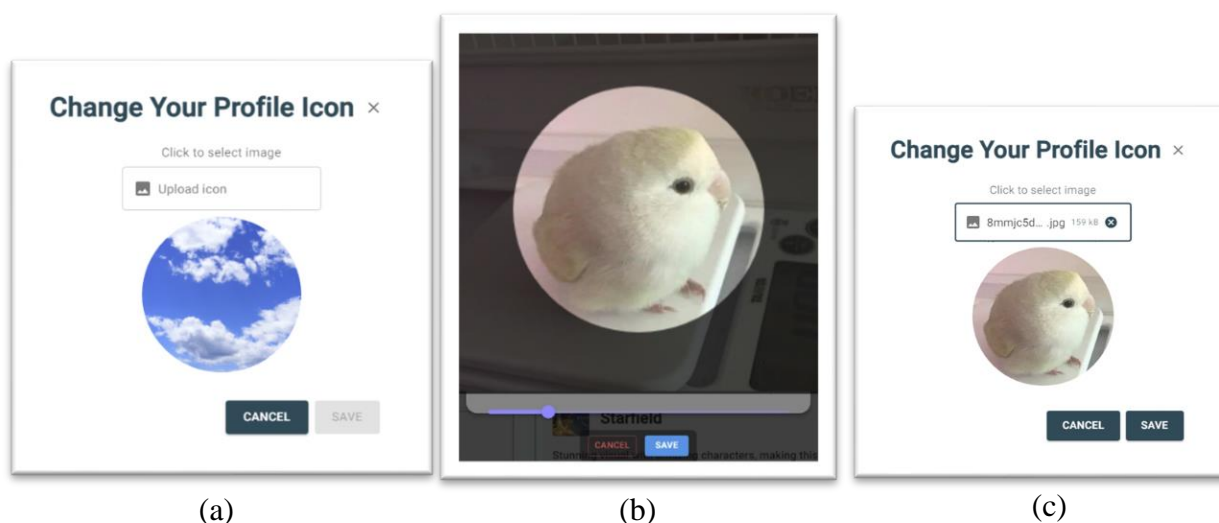


Figure 61 Update user avatar modal. (a) Modal before image upload. (b) Modal after image upload, with avatar editor backdrop to set the icon. (c) Modal after the image is uploaded and the icon is set

Next to the previous section and on the right-hand side, it displays the username, the user’s last active time, and a box for user reviews. A verification icon is placed adjacent to the username, indicating the verification status of the user’s email address. An unverified status is represented by a red exclamation mark, while a green tick denotes a verified status. The user reviews box presents all the reviews created by the user, with options to sort them by recency or score. The review fetching feature employs an infinite scroll, automatically retrieving new reviews as the user scrolls to the bottom of the review box, until all the user’s reviews have been fetched. This is accomplished using the Intersection Observer API and IntersectionObserver interface, implemented in a manner similar to the section tracking in the game analytics page (refer to the previous section). The fetch API is triggered whenever the user scrolls and interacts with the bottom of the review box.

As mentioned previously, profile users can set their profile page to private or public. Under the private view. When a user sets their profile to the public, other users are granted access to view their personal information, which includes their email address, gender, and age. Additionally, other users can see the last time the profile owner was active and read all the reviews they have created (See Figure 62 (a) and (b) below).

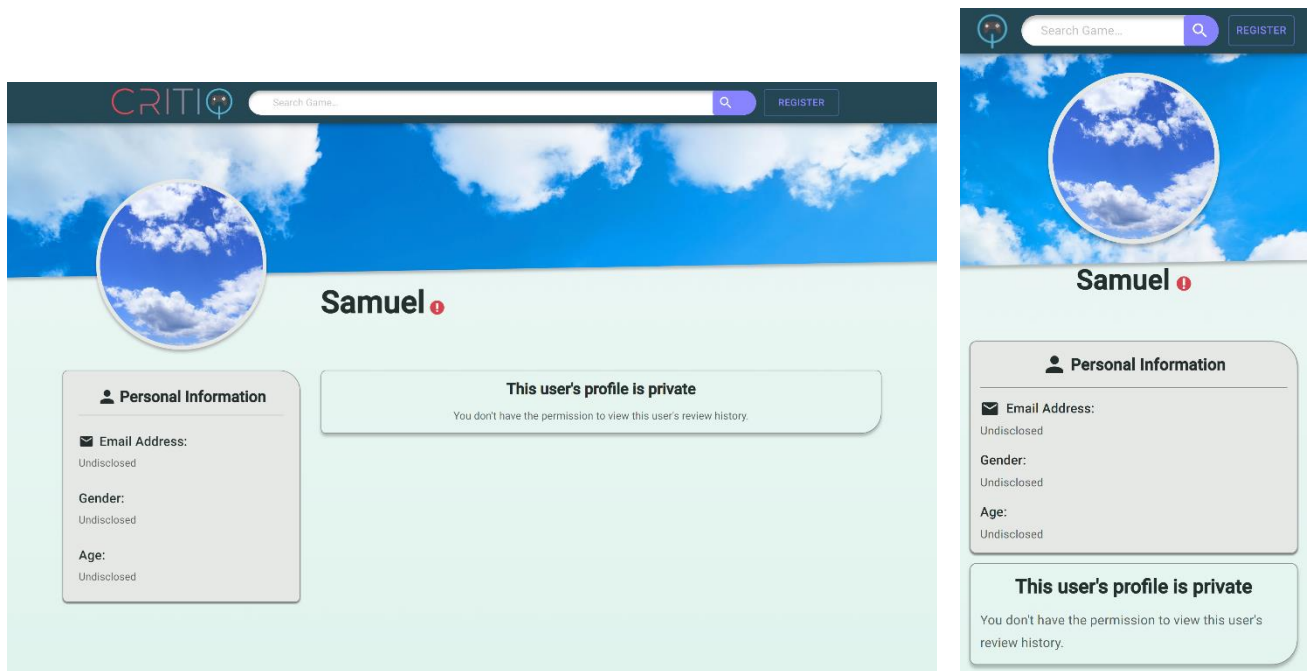


Figure 62 Web application's profile page design under visitor view and private profile. (a): Private profile page design for desktop viewport. (b): Private profile page design for mobile viewport.

4.3 Web Scraping

Web scraping was adopted to extract information of existing games on the Steam platform. Using Steam Web API (Valve, 2023), 157068 games in total were discovered, and data such as description, developers, genres, categories, and release dates were scraped from Steam in October 2023. The data was then saved in JavaScript Object Notation (JSON) format.

Subsequently, a Python program was used to parse the data and modified them to match the format used in the database (See Figure 63). This data was utilized to establish the foundation of data for our platform. Furthermore, these data will be utilized in topic modeling experiments, employing various methods to construct distinct topic models, including topic models trained with top categories and genres.

Valve	0	CS2	Valve	0	2023
Testing	0	Testing	Testing	[NULL]	2023-09-22
[NULL]	0	Monster Hunter World	Capcom	[NULL]	July 2018
CDProjekt Red	0	Cyberpunk 2077	CDProjekt Red	[NULL]	2020
CDProjekt Red	0	Cyberpunk 2077 Phantom Liberty	CDProjekt Red	0.5	2023
Bathesda Game Studio	0	Starfield	Bathesda	[NULL]	2023
Valve	0	Dota 2	Valve	[NULL]	2013
Twilight Sonata Studio	0	大富翁少女 18DLC	Twilight Sonata Studio	[NULL]	10 Jan, 2022
Game Creator's School	0	The Brave vs Dragon	Game Creator's School	[NULL]	8 Jul, 2022
Smunty Studios	0	FourPlay Chess	Smunty Studios	[NULL]	30 Apr, 2022
DigitalDream	0	Scratch Girl	DigitalDream	[NULL]	14 Dec, 2021
Wackytoaster	0	Rimebeard Demo	Wackytoaster	[NULL]	11 Nov, 2021
E.144	0	Yavi	E.144	[NULL]	13 Dec, 2021
Quentin Edel	0	Find The Cat	Quentin Edel	[NULL]	6 May, 2022
Pawel Wiecha	0	Cosmos Conquer	Iguana Mercenary	[NULL]	4 Mar, 2022
	0	Oasis: Dark Forest Playtest		[NULL]	10 Nov, 2021
Immure Creations®	0	The Infecting 3	Immure Creations®	[NULL]	30 Nov, 2021

Figure 63 Sample Database Records of Scraped Games from Steam

4.4 Backend System

This section will discuss the preliminary results of the backend solution, including CI/CD (Section 4.4.1), API Endpoints and Database (Section 4.4.2), API Security (Section 4.4.3), S3 Bucket (Section 4.4.4) and Stability and Testing (Section 4.4.5).

The backend solution has been set up according to the planned Methodology with Spring Boot being the main application and CI/CD handled by Jenkins and Docker (See Figure 64).

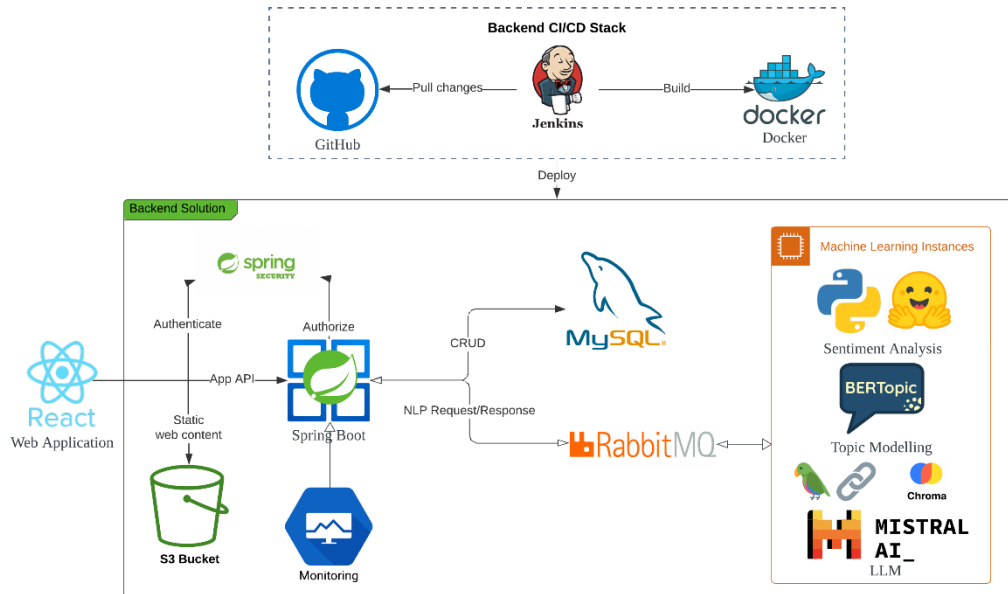


Figure 64 Backend Solution Architecture Graph

4.4.1 CI/CD

Uptime and Stability are key for modern web applications. And our backend architecture is designed to provide high uptime and stability without relying on additional backup nodes. Our pipeline is written to only deploy changes to the modified service, reducing the overall system downtime for long-starting service, including our Sentiment Analysis Model Server.



Figure 65 Jenkins deployment User Interface with different stages of deployments.

By using a Custom CI/CD pipeline (See Figure 65 above), we have reduced the downtime for our backend system during deployment to approximately 10 seconds with minimal impact on the end-user experience of our platform. Moreover, Jenkins also offers an intuitive UI that displays the status and duration of each deployment with failed deployments marked in red.

4.4.2 API Endpoints and Database

The APIs required by the frontend application have been developed and tested to optimize for performance and stability.

By creating API endpoints based on the need of frontend applications and optimizing database performance, the Round-Trip-Time (RTT) of the most commonly used API endpoints has been lowered to within 300ms.

Database optimization techniques including Indices on commonly queried entities and lazy load, which is only loading information associated with the entity when needed, significantly reduced the RTT of most API endpoints, improving the responsibility of the application. The most common API, `/findGameById`, is used to display information regarding a selected game and can be executed in 65 milliseconds (See Figure 66).

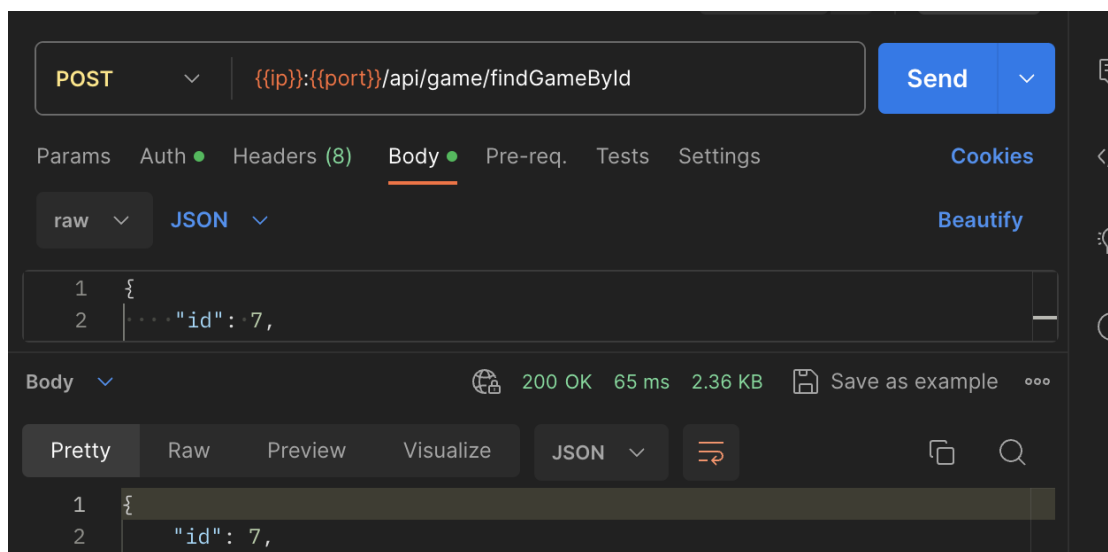


Figure 66 API call to `/findGameById` to fetch specific game information finishes in 65ms

For API calls that perform exhaustive searches, including the Advanced Search feature, the RTT depends on the size of the returned result. Testing has shown that the worst case's RTT still falls below 300ms (See Figure 67).

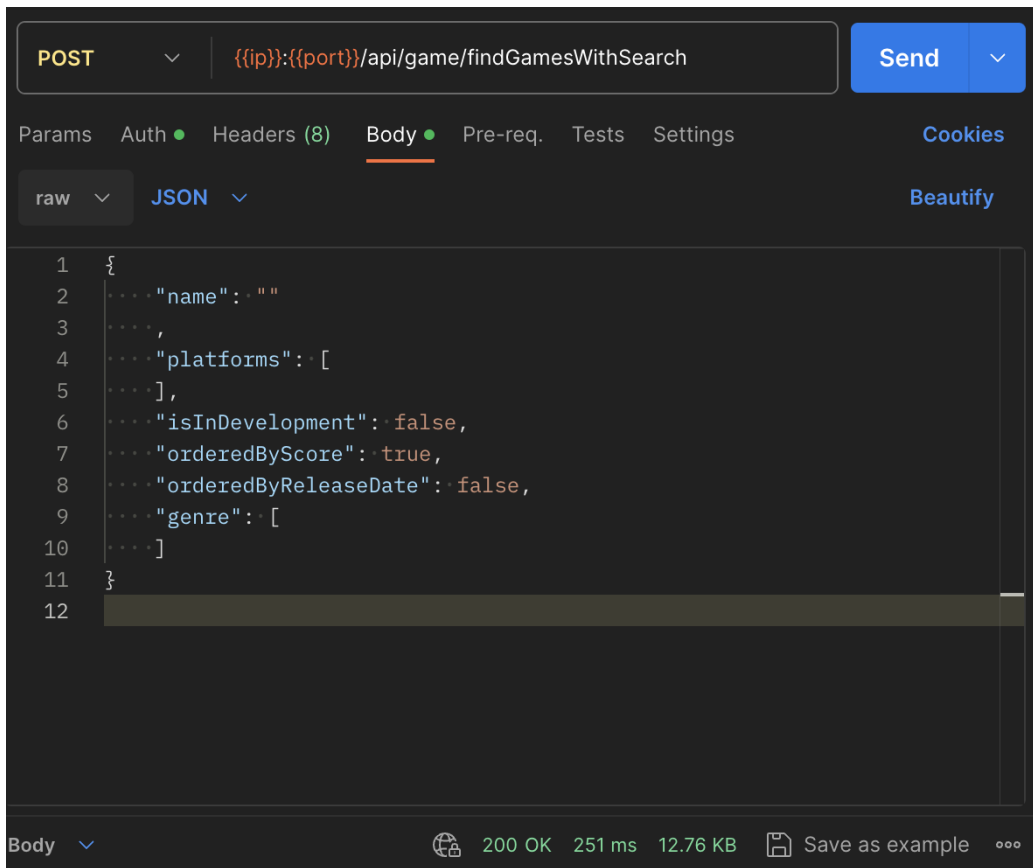


Figure 67 API call to /findGamesWithSearch to perform exhaustive game search finishes in 251ms

Our database plan offers a connection limit of 150 concurrent connections from any location (See Figure 68). To improve the performance of complex queries, including Advanced Game Search or Data Analysis, setting a larger connection size from Spring Boot had improved the query times by over 100%. Further testing has shown that 70-100 concurrent connection to the database in the production environment yielded the best result without utilizing too many connections (See Figure 69). A small portion of connection pool is reserved for local development and testing purposes.

1 vCPU / 2 GB RAM / Storage minimum: 30 GB / Connection limit: 150

Figure 68 Current Database Plan with 2GB RAM offer a maximum of 150 concurrent connections



Figure 69 Spring Boot utilizes 80 concurrent connections to the database

4.4.3 API Security

API Security is crucial in maintaining the confidentiality, integrity, and availability of our platform and its data by only permitting data access and modifications. The system uses the `@AuthenticationPrincipal` annotation in the REST API endpoint of the backend application to obtain the user based on the username or email from the JWT in the HTTP request. The system can also restrict access to the API call by verifying the user's information. The figure below illustrates how the system checks if the user's ID matches the requested user's ID before sending the verification email (See Figure 70).

```
@PostMapping(Ⓜ"/sendVerifyEmail")
public ResponseEntity<Void> sendVerifyEmail(@RequestBody UserRequest userRequest, @AuthenticationPrincipal User u){
    if (u == null || !Objects.equals(u.getId(), userRequest.getId())) {
        throw new AccessDeniedException("Access Denied");
    }
}
```

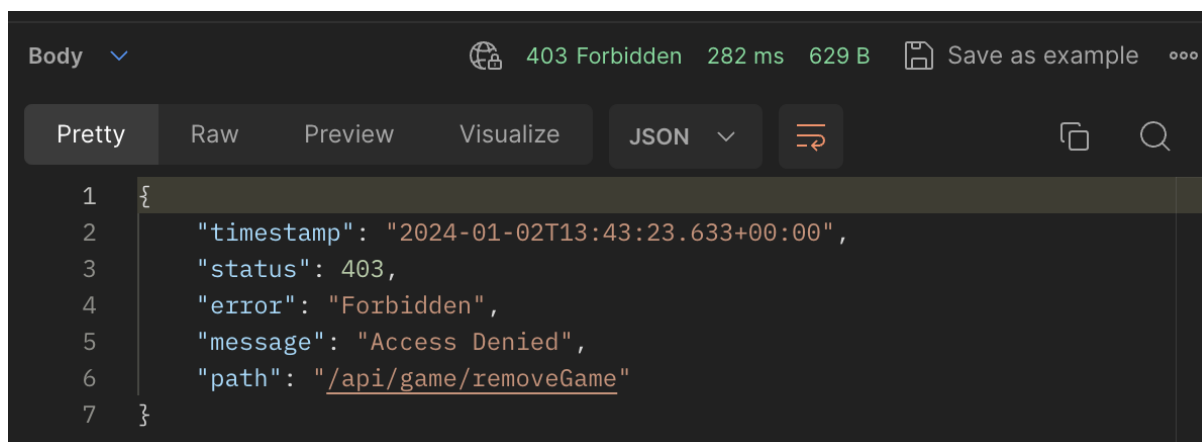
Figure 70 Access Control by verifying the user based on the JWT sent in HTTP requests using the `@AuthenticationPrincipal` annotation.

An alternative annotation `@PreAuthorize` can directly access the user making the requests based on the JWT token and make use of the Spring Expression Language (SPeL). This annotation will check for permission based on the SPeL evaluation. In our application, it is used to check for the role of the user prior to method invocation. The figure below showcases that the `/removeGame` API endpoints should only be accessed by user with the ADMIN role and will automatically return a 403 Forbidden Error otherwise (See Figure 71).

```
@PreAuthorize("hasAuthority('ROLE_ADMIN')")
@PostMapping(Ⓜ"/removeGame")
public ResponseEntity<Void> removeGame(@RequestBody GameRequest gameRequest, @AuthenticationPrincipal User u) {
    try {
        gameService.removeGame(gameRequest);
        return ResponseEntity.noContent().build();
    } catch (Exception e) {
        throw new ResponseStatusException(HttpStatus.valueOf( code: 400), e.getMessage());
    }
}
```

Figure 71 Access Control by verifying the user's role based on JWT sent in HTTP requests prior to method invocation using the `@PreAuthorize` annotation.

Attempting to access a protected API without a valid JWT, or without being the authorized user or having the necessary permission, will result in a 403 Forbidden error (See Figure 72)



The screenshot shows a REST client interface with a dark theme. At the top, it displays 'Body' with a dropdown arrow, a globe icon, '403 Forbidden', '282 ms', '629 B', a save icon, 'Save as example', and a menu icon. Below this is a toolbar with 'Pretty', 'Raw', 'Preview', 'Visualize', 'JSON' (with a dropdown arrow), and a refresh icon. The main area shows a JSON response with line numbers 1 through 7 on the left. The JSON content is: { "timestamp": "2024-01-02T13:43:23.633+00:00", "status": 403, "error": "Forbidden", "message": "Access Denied", "path": "/api/game/removeGame" }

Figure 72 403 Forbidden Error on Unauthorized Access to Protected API endpoints

4.4.4 S3 Bucket Storage

The S3-compatible storage Solution provided by Digital Ocean offers high availability, To optimize the user’s experience, a Content Delivery Network (CDN), is a network of edge servers that serve the content to the user based on the user’s geographic location to minimize network traffic time, provided by Digital Ocean is used. By using a CDN, we can minimize the data fetching time of common User Interface elements in the web application, including User Icon, Game Images, and Review Images.

Together with browser and server caches, images presented on the web application can be loaded efficiently and quickly without hindering the user experience during page navigation or exploration (See Figure 73).

image?url=%2Flo...	GET	200	h3	https	104.21....	webp	1:0	2...	24 ms
image?url=https...	GET	200	h3	https	104.21....	webp	1:0	1...	31 ms

Figure 73 Fetching of Cached Image(s) can be performed within 50ms

To ensure integrity of files uploaded to the storage, any modification to the storage bucket is only permitted through Spring Boot secured API endpoints, preventing unwanted access and modification of meta-data, including uploader, upload time and data. In addition, additional information including the uploader of the files will be tracked during file upload for security purposes and tracking (See Figure 74) while the uploader for files uploaded by the Backend system will be marked as “System”.

```
metadata.setHeader("uploader", uploader);
```

Figure 74 Uploader Header is set to the user's name during file upload

4.4.5 Stability and Testing

API testing using Gatling, a scalable load-testing tool, has shown scalable performance in terms of basic and complex CRUD operations (See Figure 75). The system can sustain 60 concurrent users performing queries while maintaining an acceptable performance of around 0.2 second of response time per database query. The increase in users has not led to any depreciation of application performance and stability.

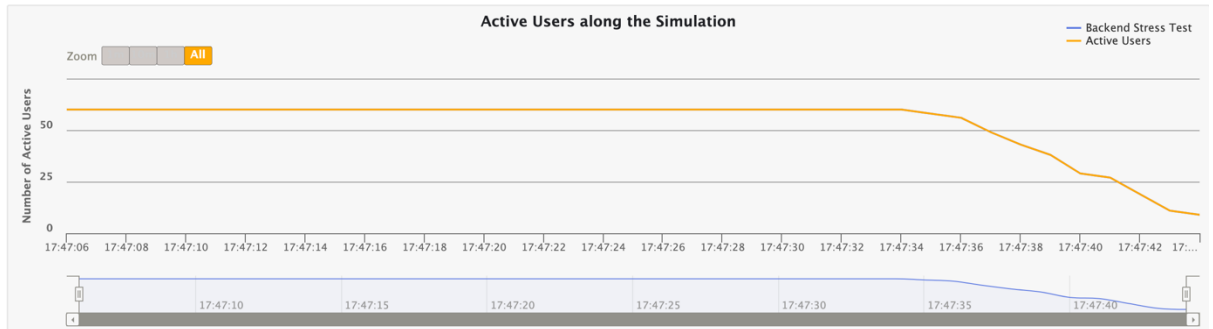


Figure 75 Backend Load-Testing using Gatling, showing the ability to sustain 60 active users performing complex queries.

5 Difficulties and Limitations

This section discusses the primary challenges and constraints experienced throughout the development process. The structure of this section is as follows: Section 5.1 discusses HTTPS and SSL certificates. Section 5.2 addresses the issue of Message Queue Disconnection. Section 5.3 explores the compatibility issues of Nest.js with other libraries. Section 5.4 examines the issues related to dataset noise.

5.1 HTTPS and SSL Certificate

Our backend server was hosted on a Virtual Private Server (VPS) from Contabo, which did not provide a Secure Sockets Layer (SSL) certificate without an extra charge. However, modern browsers that prioritize security would automatically convert any HTTP requests from a site using HTTPS to HTTPS requests.

Since HTTP and HTTPS are two distinct origins for a backend service, HTTPS requests to an HTTP-only server will result in a 404 Not Found error. This resulted in the backend server being unable to receive any requests from the frontend application. To adhere to modern security standards, our frontend application was hosted on Vercel, a cloud website hosting platform, which automatically provided an SSL certificate to the application using Let's Encrypt, an open Certificate Authority (CA) from the Internet Security Research Group (ISRG).

We initially attempted to self-sign a digital SSL certificate. However, this was not a viable option, as modern browsers such as Google Chrome, Firefox, and Safari would not trust the certificate since it was not signed by a CA and would show an error page (See Figure 76).

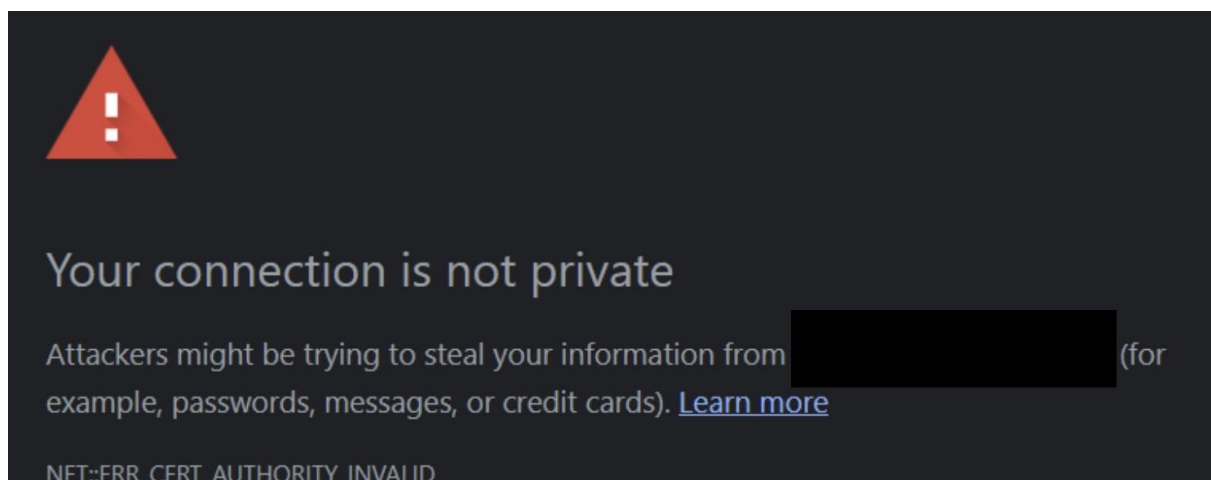


Figure 76 Google Chrome Browser Error Page on Visiting Website with a self-signed digital certificate

Our solution was to install Let's Encrypt locally on the VPS using CertBot to set up the SSL certificate to enable HTTPS for the traffic going to the Spring Boot backend services. Let's Encrypt performs domain validation using challenges where only the domain owner can perform, making the certificate official compared to a self-signed certificate where no domain validation is performed. We faced some challenges when using the signed certificate file to enable HTTPS on Spring Boot. Firstly, the certificate generated was either in the PEM or Java Keystore (JKS) format, which were both incompatible with Spring Boot. We had to convert the keystore file into a PCKS12 format, which could be read by Spring Boot. Secondly, enabling HTTPS on Spring Boot required the front end to use HTTPS calls even in the local development environment by default. However, Spring Boot would not accept the certificate that was signed on the VM when it was run on localhost. To overcome this and facilitate local development, we disabled HTTPS during local development with different environment variables and allowed HTTP requests to be made to the local backend service.

However, certificates issued by Let's Encrypt are only valid for 3 months. And without an automated renewing service, manual renewal will be required prior to the expiration of the current certificate. By using a modern browser and visiting the backend service, the certificate issued by Let's Encrypt can be viewed, together with the certificate key (See Figure 77 below).

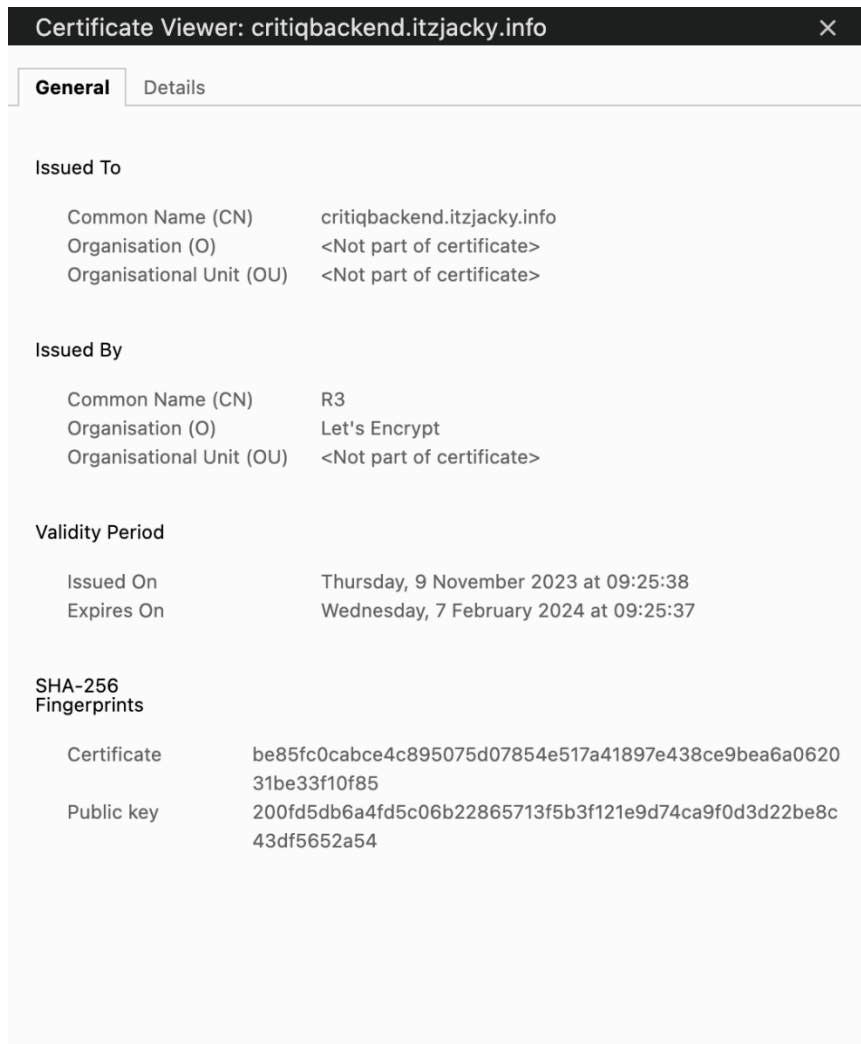


Figure 77 Certificate Viewer on Backend Domain Address using Google Chrome

5.2 Message Queue Disconnection

The Pika Python library, the official Python library for working with RabbitMQ, was used to connect to the Message Queue Server, which supports Machine Learning services. The incoming reviews were read from the message queue and processed by the deployed ML Models. However, random disconnections were experienced, without any error messages from both the Python client and the Message Queue Server. The logs revealed that the disconnection frequency varied from 10 minutes to 3 hours after redeployment. There were no error messages from either the Python client or the Message Queue Server. Logs reveal that disconnection occurs at random frequencies, ranging from 10 minutes to 3 hours after redeployment.

Many solutions for common problems were attempted to fix the issue, including:

- **Enforcing Mandatory Delivery Confirm (Acknowledgement):** This solution aimed to prevent data loss due to disconnections by using the transactional mode of Pika. The producer would send a batch of messages and wait for the broker to confirm that they were received and persisted. The producer would block until the broker responded with a **commit_ok** message, indicating that the transaction was successful. If the connection was closed or the broker returned an error, the producer would catch the exception and retry the transaction.
- **Reverting to Single Threading:** This solution aimed to reduce the complexity and the overhead of managing multiple connections and channels by using a single-threaded approach for both the producer and the consumer. The producer and the consumer would use the *pika.BlockingConnection* class, which provides a simple and synchronous interface for interacting with the broker. The producer and the consumer would use the *channel.basic_publish()* and *channel.basic_consume()* methods to send and receive messages, respectively. The *channel.basic_consume()* method would block until a message was delivered, and then invoke a callback function to process the message.
- **Mandatory Flag (Return message on Failure):** This solution aimed to handle any messages that could not be routed to a queue by using the mandatory flag of Pika. The producer would instruct the broker to return any unroutable messages, which could happen if the queue did not exist, or if the queue was full or had reached its limit. The producer would set the mandatory parameter to True when calling the *channel.basic_publish()* method. The producer would also register a callback function with the *channel.add_on_return_callback()* method, which would be invoked when the broker returned a message. The callback function would then handle the returned message, such as logging it, retrying it, or discarding it.

However, none of the mentioned solutions solved the disconnection issue.

The final solution that resolved the unpredictable disconnection issue was to disable the heartbeat check (See Figure 78). The heartbeat check is a mechanism that allows the broker and the client to detect and close stale connections. The broker and the client exchange heartbeat frames at regular intervals, and if either side does not receive a heartbeat frame within a specified timeout, it will close the connection. However, this mechanism can also cause problems if the network is unreliable, or the client is busy processing messages. To use this solution, the Blocking Connection for the consumer was set to not use the heartbeat

check. This would tell the broker not to expect any heartbeat frames from the client. With further testing, disabling the heartbeat check did not lead to any additional packet loss or drop.

```
connection = pika.BlockingConnection(pika.ConnectionParameters(  
    host= , port= , credentials=credentials, heartbeat=0))
```

Figure 78: Code snippet to initiate the RabbitMQ connect with heartbeat check disabled.

5.3 Next.js Compatibility and Debugging Issues

It is challenging to navigate the compatibility issues in Next.js, especially when dealing with dependencies that use different JavaScript module systems. During the development of the React application using Next.js, it's observed that certain external React libraries conflict with Next.js due to the complicated relationship between various JavaScript module systems and Next.js's server-side rendering (SSR) nature.

For instance, when an external package that is written in ES6 syntax or later is imported, if the library maintainers have not transpile their package to ES5 or CommonJS, errors of **“SyntaxError: Cannot use import statement outside a module”** will occur. This happens because Next.js, which runs on both the server and client side, requires the appropriate module styles for each. The server side runs on Node.js, which uses the CommonJS module system, while the client side can handle the ES6 imports. If a library does not export CommonJS files, it fails on the server side, leading to such errors.

This situation was encountered when using the MUI file input component from the MUI file input library, when implementing the game page (See Section 4.2.6). The creator of the library did not consider the compatibility issue with different module systems for their package, resulting in error. A workaround was discovered by adding the problematic package to the `transpilePackages` option in the `next.config.js` file. This instructs Next.js to transpile the specified package, converting its modern JavaScript code into a version compatible with all browsers and preventing syntax errors. However, this comes at the cost of performance as transpiling unnecessary packages can slow down the build time.

Nonetheless, adding the library to `transpilePackages` option does not always resolve the issue. A similar issue was encountered when using the `nivo` package to implement the game analytics page (See Section 4.2.9). Despite Next.js 13's native support for ECMAScript Modules (ESM), problems may still arise with packages like `nivo` that depend on other packages using ESM, in our case, the `d3-scale` library. Adding `nivo` to the `transpilePackages` option in the `next.config.js` file in this case does not resolve the issue. This is because the

problem is not with the nivo package itself, but with how its dependency d3-scale is being imported. We tried to switch to loose mode in the Next.js configuration as suggested by the Next.js documentation, which Next.js will try to automatically detect whether the imported package is ESM or CommonJS, and correct the error accordingly. This fails to resolve the issue as the nivo package contains dependencies on both ESM and CommonJS module. Out of solution, we resorted to downgrading the nivo package to the version of 0.80.0, where the d3 packages used are not ESM only and support CommonJS imports. This difficulty highlights the complexities involved in managing dependencies and module systems in Next.js.

Another challenge in Next.js is the complexity involved in debugging `getServerSideProps`. This server-side function is a key part of Next.js's page routing implementation, fetching data, and enabling Next.js to pre-render the page at build time, thereby enhancing performance and reducing page load time. However, its SSR nature poses a hurdle for traditional browser-based debugging tools, which are unable to capture any output or error messages within this function. This makes troubleshooting issues related to data fetching particularly difficult.

5.4 Dataset Noise

Dataset noise is a significant challenge that encountered during the development of machine learning models. It refers to the presence of irrelevant or inconsistent information in the dataset that can impair the performance of models. In the context of game reviews, three main types of dataset noise were observed, which are the non-relevant text, spam of words and language inconsistency.

Firstly, non-relevant text refers to personal stories, tangential details, or entirely unrelated data within the review, implying that the review does not genuinely address the game. This kind of noise can misguide models during training, leading to incorrect predictions.

Secondly, word spam is related to the repetition of specific words. Common examples of repeated words include acronyms like "GOAT" (Greatest Of All Time), game references such as "Cake" (a term from the 2007 video game Portal, as in "The cake is a lie"), and sentiment spam of words like "good" and "bad. These types of data noise can lead to skewed word frequency, causing a model to overemphasize these words during training. This can impact the outcome of topic modeling as word spam can overshadow other significant but less commonly used words. Furthermore, it can influence sentiment analysis.

For example, repeated use of words like “good” or “bad” lead a model to misclassify the sentiment of a review, resulting in misleading sentiment results.

Finally, language inconsistency is another form of dataset noise. Game reviewers from diverse countries, communities, and backgrounds exhibit different linguistic styles and prefer to use varying jargon, meme references, or languages. This inconsistency can lead to confusion for models, particularly those that depend on language patterns or structures. Therefore, dataset noise introduces a significant difficulty in data processing and model training. The application of suitable data cleaning and preprocessing techniques is crucial to mitigate the effects of such noise and enhance model performance.

5 Project Schedule

The following timetable outlines the progression of our project throughout its entirety, detailing the tasks accomplished within each respective phase. (See Table 8).

Period	Work done
Sep	<ul style="list-style-type: none"> - Define Requirements for Web App and ML Models - Literature Review on NLP and ML Architectures
Oct	<ul style="list-style-type: none"> - Designed Web App & DB Structure - Designed and implemented Search Result Page - Performed Data Scraping for Model Training - Performed Software Architecture Design - Designed Use-Case Diagram
Nov – Dec	<ul style="list-style-type: none"> - Designed and Implemented Sign Up & Sign In Page with JWT - Designed and Implemented Game Page - Designed and Implemented Review Page - Trained & Evaluated Sentiment Analysis Model
Jan – Feb	<ul style="list-style-type: none"> - Integrated the Sentiment Analysis model into our application. - Designed and Implemented Profile Page - Implemented Topic Modelling Model - Implemented Keyword Extraction model
Feb – Mar	<ul style="list-style-type: none"> - Designed and Implemented Game Analytic Page - Designed and Implemented Landing Page - Implemented Responsive Web Design for All Pages - Fine-tuned Topic Modelling Model - Fine-tuned Keyword Extraction model
Mar – Apr	<ul style="list-style-type: none"> - Fine-tuned of all models and their integration. - Debug and Refactor Code - Prepared for the Final Presentation - Prepared Demo and Demo Video

Table 8 Proposed Schedule for the project

6 Work Distribution

The contribution of each member to the project is listed in Table 9.

Student	Contributions
Lee Chi Ho	<ul style="list-style-type: none"> - Backend Architecture Design - Cloud Solution Design and Setup - Database Design and Implementation - Spring Boot Server Implementation - DevOps Design
Cheng Pak Yim	<ul style="list-style-type: none"> - Research, Implement, Evaluate all Sentiment Analysis models. - Research Topic Modeling models - Research Keyword Extraction models - Implement Python client side of the NLP message queue.
Siu Yuk Shing	<ul style="list-style-type: none"> - Web Application Design - Frontend Implementation - Frontend Libraries and Packages Setup

Table 9 Work Distribution Table of the project

7 Conclusion and Future Works

The progress NLP, particularly in the area of LLMs, has been remarkable. Yet, its application in the gaming sector and game reviews remains limited. Recognizing this gap, this project aims to create a web-based game reviews platform that leverages NLP. This platform assists game developers with analytic features such as automated review analysis, aggregation, filtering, and organization. Moreover, it visualizes analytic data related to the players and reviews through charts and maps. The outcomes of this project highlight the feasibility of integrating NLP into the gaming industry, especially in the realm of game review analysis. This integration can assist players in finding games that align with their preferences and enable developers to identify areas for enhancement in their games.

The outcomes of this project are categorized into three main areas: frontend, backend, and machine learning. The frontend has culminated in a responsive, user-friendly, well-designed, and highly accessible web application that is compatible with various devices. It is engineered to include game review features, automated review analysis with feedback of sentiment, primary topics, keywords, and summary, reviews aggregation for a specific game, and foster interaction between players and developers with the user through review commenting feature. The backend solution is hosted on cloud platforms, utilizing a comprehensive CI/CD pipeline to accurately reflect alterations made to the production environment, thereby enabling swift development and deployment processes. Regarding machine learning models, the sentiment analysis model, with careful tuning, can be effectively implemented while maintaining high precision and performance. The topic modeling models are capable of identifying the main topic from the review as indicated by the quantitative and qualitative evaluation, and generate a succinct, human-readable name through LLM. Finally, for the keyword extraction task, with the assistance of advanced LLM development tools and frameworks such as LangChain, Mistral AI, and Chroma, the model can capture keywords related to multiple aspects from the review, generate a concise summary for the review, and aggregate reviews created by the platform users and game critics to create a summarized review efficiently.

In conclusion, this project has demonstrated that, with a thoughtfully constructed system architecture, NLP tools can be smoothly incorporated into pre-existing systems with minimal expenditure and hardware prerequisites, thereby expanding the understanding of game developers and users about the strengths of the games and areas for improvement.

Regarding the future works of this projects, since I am mainly responsible for the frontend development, only future works related to the frontend will be discussed, if you are interested in the possible future works of the backend system and machine learning tasks, please refer to the report by my teammates.

While the current state of the frontend application has met the necessary requirements and provided an intuitive user interface, there are several areas identified for future development and enhancement.

Firstly, the potential benefits of investigating UI libraries other than MUI were recognized. The choice to use Material UI as the primary React component library was driven by its extensive range of UI components that cater to a wide variety of use cases. However, some web developers have criticized MUI as being generic and cheap due to its minimalist design and overuse in web applications. During the early development stages, the adoption of other UI libraries, such as Joy UI, was contemplated because we perceived its design approach to be more vibrant and distinctive. This consideration was eventually dismissed as Joy UI was still in its beta development phase, and the number of components it provided did not meet our expectations at that time. Once Joy UI reaches full development, we should contemplate migrating to other UI libraries to enhance the appearance and aesthetics of our web application.

Secondly, the potential upgrade from Page Router to App Router in Next.js requires further exploration. Currently, our project is using the Page Router, and the App Router built on React Server Components is introduced in Next.js 13. Transitioning to the App Router from Page Router could bring benefits such as accelerated page load times, as the App Router favors client-side component rendering, thereby reducing the volume of data transmitted over the network. The decision between employing the Page Router or App Router was debated during the project's early development stages. We gained experience with both the Page Router and App Router through a project undertaken last summer. However, we perceive the App Router as a relatively new feature that is currently lacking in both internal and external support. The official Next.js documentation for the App Router is not as comprehensive or clear as it could be, which complicates its usage. Furthermore, the App Router approach is not widely supported by many external React libraries, primarily due to its lack of out-of-the-box support for Server-Side Rendering (SSR). Once the App Router achieves greater stability and support,

consideration may be given to upgrading from the Page Router to the App Router for better performance.

Finally, enhancing frontend accessibility is a primary area of focus. The basic improvements include the implementation of Accessible Rich Internet Applications (ARIA) roles and properties, the facilitation of keyboard navigation, the support for trackpad usage, and ensuring appropriate color contrast for all elements. Regarding the charts and maps in the game analytics page, the use of color blind-friendly palette is planned to ensure data visualization is accessible to all users. Furthermore, the implementation of a language change functionality is proposed to cater to a wider audience, as our web application only display in English currently. To implement this feature, the machine learning model need to be tuned to accommodate for review that is not in English. In addition, the introduction of a dark mode theme is under consideration to enhance the accessibility of the web application across various environments. This feature can enhance the user experience by providing a visually relaxing interface and reducing eye strain in low-light conditions.

These proposed enhancements aim to improve the overall user experience, performance, and accessibility of the web application. Each represents a significant area of work and will require careful planning and implementation.

References

- Abdelrazek, A., Eid, Y., Gawish, E., Medhat, W., & Hassan, A. (2023). Topic modeling algorithms and applications: A survey. *Information Systems, 112*, 102131. <https://doi.org/10.1016/j.is.2022.102131>
- Al Mursyidy Fadhlurrahman, J., Herawati, N. A., Widya Aulya, H. R., Puspasari, I., & Utama, N. P. (2023, 2023/10/10/). Sentiment Analysis of Game Reviews on STEAM using BERT, BiLSTM, and CRF. 2023 International Conference on Electrical Engineering and Informatics (ICEEI),
- Bianchi, F., Terragni, S., & Hovy, D. (2021, 2021/08//). Pre-training is a Hot Topic: Contextualized Document Embeddings Improve Topic Coherence. ACL-IJCNLP 2021,
- Birjali, M., Kasri, M., & Beni-Hssane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems, 226*, 107134. <https://doi.org/10.1016/j.knosys.2021.107134>
- Churchill, R., & Singh, L. (2022). The Evolution of Topic Modeling. *ACM Comput. Surv.*, 54(10s), Article 215. <https://doi.org/10.1145/3507900>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: arXiv.
- Egger, R., & Yu, J. (2022). A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts [Methods]. *Frontiers in Sociology, 7*. <https://doi.org/10.3389/fsoc.2022.886498>
- Fachrul, K., Yuliana, R., Laila, Z., & Hammad, J. (2022). Comparing LSTM and CNN Methods in Case Study on Public Discussion about Covid-19 in Twitter. *International Journal of Advanced Computer Science and Applications*. <https://doi.org/10.14569/ijacsa.2022.0131048>
- Gan, L., Yang, T., Huang, Y., Yang, B., Luo, Y. Y., Richard, L. W. C., & Guo, D. (2024, 2024//). Experimental Comparison of Three Topic Modeling Methods with LDA, Top2Vec and BERTopic. Artificial Intelligence and Robotics, Singapore.
- Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. In: arXiv.
- Guzsvinecz, T., & Szűcs, J. (2023). Length and sentiment analysis of reviews about top-level video game genres on the steam platform. *Computers in Human Behavior, 149*, 107955. <https://doi.org/https://doi.org/10.1016/j.chb.2023.107955>

- Khan, M. Q., Shahid, A., Uddin, M. I., Roman, M., Alharbi, A., Alosaimi, W., Almalki, J., & Alshahrani, S. M. (2022). Impact analysis of keyword extraction using contextual word embedding. *PeerJ Computer Science*, 8, e967. <https://doi.org/10.7717/peerj-cs.967>
- Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In: arXiv.
- Li, X., Li, X., Rodolfo, C. R., & Shi, X. (2022). GloVe-CNN-BiLSTM Model for Sentiment Analysis on Text Reviews. *Journal of Sensors*. <https://doi.org/10.1155/2022/7212366>
- Lin, D., Bezemer, C.-P., Zou, Y., & Hassan, A. E. (2019). An empirical study of game reviews on the Steam platform. *Empirical Software Engineering*, 24(1), 170-207. <https://doi.org/10.1007/s10664-018-9627-4>
- Maisa, J. A.-K., Marwah, A., Mariam, M. B., & Bayan, A.-H. (2023). Sentiment Analysis for People's Opinions about COVID-19 Using LSTM and CNN Models. *Int. J. Online Biomed. Eng.* <https://doi.org/10.3991/ijoe.v19i01.35645>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In: arXiv.
- Morgenthaler, S. (2009). Exploratory data analysis. *WIREs Computational Statistics*, 1(1), 33-44. <https://doi.org/https://doi.org/10.1002/wics.2>
- Öğüt, M. S. S. (2021, 17-19 Nov. 2021). A Performance Study Depending on Execution Times of Various Frameworks in Machine Learning Inference. 2021 15th Turkish National Software Engineering Symposium (UYMS),
- Prusa, J., Khoshgoftaar, T. M., & Seliya, N. (2015, 9-11 Dec. 2015). The Effect of Dataset Size on Training Tweet Sentiment Classifiers. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA),
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training.
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: arXiv.
- Rogers, A., Kovaleva, O., & Rumshisky, A. (2021). A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8, 842-866. https://doi.org/10.1162/tacl_a_00349
- Ruseti, S., Sirbu, M.-D., Calin, M. A., Dascalu, M., Trausan-Matu, S., & Militaru, G. (2020, 2020). Comprehensive Exploration of Game Reviews Extraction and Opinion Mining Using NLP Techniques. *Advances in Intelligent Systems and Computing*

- Sobkowicz, A. (2017). *Steam Review Dataset (2017)*.
<https://doi.org/https://zenodo.org/doi/10.5281/zenodo.1000884>
- Srivastava, A., & Sutton, C. (2017). Autoencoding Variational Inference For Topic Models.
In: arXiv.
- Stepien, K. (2021). *Topic Modelling and Data Analysis: Impact of using topic modelling on game reviews* [https://eprints.lincoln.ac.uk/id/eprint/49978/1/Thesis -
Konrad Stepien Corrections 16-01-2022%20\(1\).pdf](https://eprints.lincoln.ac.uk/id/eprint/49978/1/Thesis_-_Konrad_Stepien_Corrections_16-01-2022%20(1).pdf)
- Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019, 2019). How to Fine-Tune BERT for Text Classification? *Lecture Notes in Computer Science*
- Tan, J. Y., Chow Sai Kit, A., & Tan, C. W. (2021). *Sentiment Analysis on Game Reviews: A Comparative Study of Machine Learning Approaches*.
- Valve. (2023). *Steamworks API Reference* <https://partner.steamgames.com/doc/webapi>
- Vigliato, M., Lin, D., Hindle, A., & Bezemer, C.-P. (2022). What Causes Wrong Sentiment Classifications of Game Reviews? *IEEE Transactions on Games*, 14(3), 350-363.
<https://doi.org/10.1109/TG.2021.3072545>
- Wenxuan, Z., & Yuxuan, W. (2022). Semantic sentiment analysis based on a combination of CNN and LSTM model. *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*. <https://doi.org/10.1109/mlke55170.2022.00041>
- Yu, Y., Nguyen, B. H., Dinh, D. T., Yu, F., Fujinami, T., & Huynh, V. N. (2022, 2022). A Topic Modeling Approach for Exploring Attraction of Dark Souls Series Reviews on Steam. AHFE (2022) International Conference,
- Zhang, Y., & Wallace, B. (2016). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In: arXiv.