

# COMP4801 Final Year Project



## Game Testing and Evaluation Platform with Machine Learning for Game Developers CritiQ

### Interim Report

Supervisor: Dr. Chim T. W.

#### Group Members:

Lee Chi Ho 3035785520

Cheng Pak Yim 3035784942

Siu Yuk Shing 3035779430

Date of Submission: 21/1/2024

## **Abstract**

The rapid development in Machine Learning (ML) and Natural Language Processing (NLP) has demonstrated evolutionary improvements in textual understanding and processing. However, such technologies have not been widely adopted for game review analytics on major review platforms, requiring manual data processing to extract valuable information. We aim to investigate the potential of automated review analysis by harnessing the power of Natural Language Processing.

This project aims to develop a full-stack web application using Spring Boot and React frameworks that incorporates NLP features, including Sentiment Analysis and Topic Modelling. The platform performance achieved at the current stage showcases great potential, offering high accuracy, efficiency, and performance. Future development and adoption in NLP could significantly reduce the time cost of manual data analysis and processing hence game-development lifecycles could be improved through eliminating the need for labor-intensive manual analysis and allowing game developers to make data-driven decisions.

A basic version of the frontend and backend application has been built and hosted. Development on the application will continue, together with further tuning of the Sentiment Analysis Machine Learning Model.

## **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>viii</i>
<b>1. Introduction</b> .....	<b>1</b>
<b>1.1. Background</b> .....	<b>1</b>
<b>1.2. Objectives</b> .....	<b>2</b>
<b>1.3. Deliverables</b> .....	<b>2</b>
<b>1.4. Outline</b> .....	<b>2</b>
<b>2. Related Work</b> .....	<b>3</b>
<b>3. Methodology</b> .....	<b>5</b>
<b>3.1 Machine Learning and Natural Language Processing</b> .....	<b>6</b>
3.1.1 Sentiment Analysis.....	6
3.1.2 Topic Modelling.....	20
3.1.3 Keyword Extraction.....	22
<b>3.2 Frontend Web Application</b> 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). .....	<b>24</b>
3.2.1 Technologies Involved.....	24
3.2.2 Design Approach.....	24
3.2.3 Frontend Authentication.....	25
<b>3.3 Backend Technologies</b> .....	<b>27</b>
3.3.1 Spring Boot Server Application.....	27
3.3.2 Authentication with JWT.....	28
3.3.3 Email Service.....	29
3.3.4 Database.....	30
3.3.5 Object Storage.....	31
3.3.6 Continuous Integration/Continuous Delivery (CI/CD).....	32
3.3.7 Message Queue.....	33
3.3.8 Hosting.....	35
3.3.9 Monitoring.....	36
<b>4 Current-Stage Result</b> .....	<b>37</b>
<b>4.1 Sentiment Analysis</b> .....	<b>37</b>
<b>4.2 Web Application</b> .....	<b>45</b>
4.2.1 Toolbar.....	45
4.2.2 Login and Registration.....	46
4.2.3 Forgot Password Page and Reset Password Page.....	49
4.2.3 Search Result Page.....	52
4.2.4 Game Page.....	55
4.2.5 Review Page.....	58
<b>4.3 Web Scraping</b> .....	<b>61</b>
<b>4.4 Backend System</b> .....	<b>62</b>

4.4.1	CI/CD .....	63
4.4.2	API Endpoints and Database .....	64
4.4.3	API Security .....	66
4.4.4	S3 Bucket Storage .....	68
4.4.5	Stability and Testing .....	69
<b>5</b>	<b><i>Difficulties Encountered</i></b> .....	<b>70</b>
5.1	HTTPS and SSL Certificate .....	70
5.2	Message Queue Disconnection .....	72
<b>6</b>	<b><i>Proposed Schedule</i></b> .....	<b>74</b>
<b>7</b>	<b><i>Work Distribution</i></b> .....	<b>74</b>
<b>8</b>	<b><i>Conclusion</i></b> .....	<b>75</b>

## 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. ....	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. ...	22
Figure 9: Two game reviews and the response from ChatGPT hosted by Azure when prompting to classify their sentiment. ....	23
Figure 10: JWT Claims extracted from the Access Token user received on login. ....	29
Figure 11: Database Entity Relations Diagram. ....	30
Figure 12: Sample Code to upload a file to the S3 Bucket. ....	31
Figure 13: Digital Ocean Spaces Dashboard. ....	32
Figure 14: Jenkinsfile pipeline written for deploying the Backend Server and NLP Server with the use of docker and dockerfiles. ....	32
Figure 15: Message Queue Structure for Supporting Inter-process Machine Learning Application. ....	34
Figure 16: RabbitMQ Management Panel showing all the queue connections. ....	34
Figure 17: Grafana Dashboard displaying uptime, CPU, and Memory Utilization by the Spring Boot Application. ....	36
Figure 18: Grafana Dashboard shows the database connection pool size, maintaining a stable connection to the database. ....	36
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. ....	37
Figure 20: Precision of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative. ....	38
Figure 21: Recall of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative. ....	39
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. ....	39
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. ....	39
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. ....	40

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 .....	41
Figure 26: ROC-AUC of all models trained with the balanced dataset. (a): on the imbalanced validation set. (b): on the balanced validation set. ....	43
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. ....	43
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) .....	44
Figure 29 Web application's toolbar design. (a): Toolbar design for desktop viewport. (b): Toolbar design for mobile viewport. (c): Avatar icon button drop down menu. ....	45
Figure 30 Web application's register modal popup .....	46
Figure 31 Register modal input validations .....	47
Figure 32 Web application's login modal popup .....	47
Figure 33 Web application's forget password page design. ....	49
Figure 34 Reset password email .....	49
Figure 35 Web application's reset password page design .....	50
Figure 36 Web application's reset password page with invalid token .....	51
Figure 37 Web application's search result page design. (a): Search result page design for desktop viewport. (b): Search result page design for mobile viewport. ....	52
Figure 38 Search result page searching example .....	52
Figure 39 Advanced search modal for search result page .....	53
Figure 40 Web application's game page design .....	55
Figure 41 Game detailed information popup modal .....	56
Figure 42 Web application's review page design .....	58
Figure 43 Sample Database Records of Scraped Games from Steam .....	61
Figure 44 Backend Solution Architecture Graph.....	62
Figure 45 Jenkins deployment User Interface with different stages of deployments. ....	63
Figure 46 API call to /findGameById to fetch specific game information finishes in 65ms ..	64
Figure 47 API call to /findGamesWithSearch to perform exhaustive game search finishes in 251ms.....	65
Figure 48 Current Database Plan with 2GB RAM offer a maximum of 150 concurrent connections .....	65
Figure 49 Spring Boot utilizes 80 concurrent connections to the database .....	65
Figure 50 Access Control by verifying the user based on the JWT sent in HTTP requests using the @AuthenticationPrincipal annotation. ....	66
Figure 51 Access Control by verifying the user's role based on JWT sent in HTTP requests prior to method invocation using the @PreAuthorize annotation. ....	66
Figure 52 403 Forbidden Error on Unauthorized Access to Protected API endpoints .....	67
Figure 53 Fetching of Cached Image(s) can be performed within 50ms.....	68
Figure 54 Uploader Header is set to the user's name during file upload .....	68
Figure 55 Backend Load-Testing using Gatling, showing the ability to sustain 60 active users performing complex queries. ....	69
Figure 56 Google Chrome Browser Error Page on Visiting Website with a self-signed digital certificate.....	70
Figure 57 Certificate Viewer on Backend Domain Address using Google Chrome .....	71
Figure 58: Code snippet to initiate the RabbitMQ connect with heartbeat check disabled. ....	73

## 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. ....	40
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.....	41
Table 3: Weighted Average F1-Score of all models on the imbalanced validation sets. ....	42
Table 4: Weighted Average F1-score of all models on the balanced validation sets. ....	42
Table 5: Median Speedup of inference time of both machines. ....	44
Table 6 Proposed Schedule for the project .....	74
Table 7 Work Distribution Table of the project.....	74



## 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
CTM	Contextualized Topic Model
EDA	Exploratory Data Analysis
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
RTT	Round-Trip-Time
RWD	Responsive Web Design
SDK	Software Development Kit
SSL	Secure Socket Layer
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), and Outline (Section 1.4).

### **1.1. Background**

Presently, game review platforms such as Steam and OpenCritic are predominantly designed to serve the needs of end-users. The functionalities offered by these platforms, particularly in terms of review analysis and filtering, are often limited. Consequently, this poses a challenge for developers seeking to derive meaningful insights from user opinions.

This project centers around the creation of 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 (Lin et al., 2019) for developers. By harnessing 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.
- 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. Outline**

This report will present a comprehensive analysis of the project, focusing on key sections, including Related Work (Section 2), Detailed Methodologies (Section 3), Current-stage Results (Section 4), Difficulties encountered (Section 5), Updated proposed schedule (Section 6), and a Conclusive summary (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 literatures were 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. Viggiato 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 the majority of 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. 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 comparison 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 analyze 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 provides 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 analyzing 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, and 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.

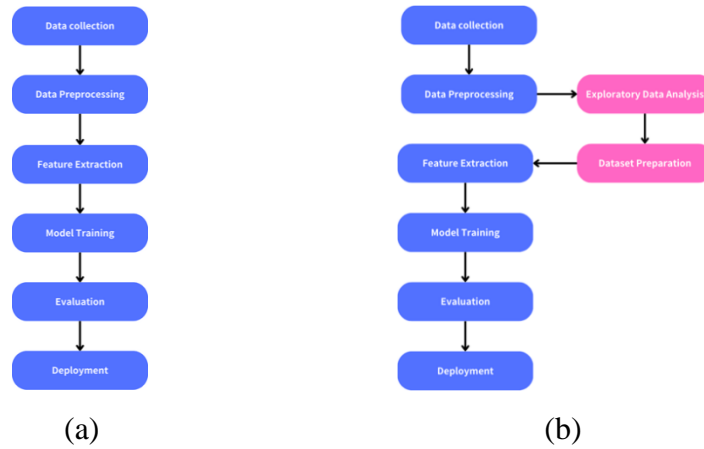


Figure 1: Usual stages in Sentiment Classification. (a): Six usual stages in Sentiment Classification. (b): Overall framework of section Sentiment Analysis of the project. Two additional stages were added and labeled in pink.

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 aforementioned 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 4.63M, a 27.8% reduction in terms of size.

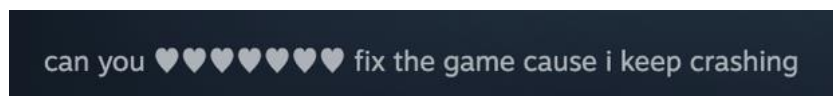


Figure 2: Steam automated comment filtering. The sensitive word was replaced by consecutive heart symbols.

### 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 “&gt;”, “&quot”, “<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.

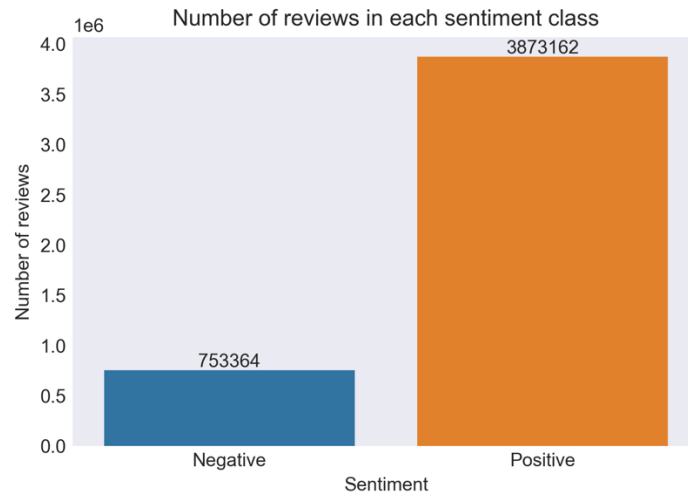


Figure 3: Number of reviews in each sentiment class in the cleaned dataset with a 5.14: 1 positive to negative ratio.

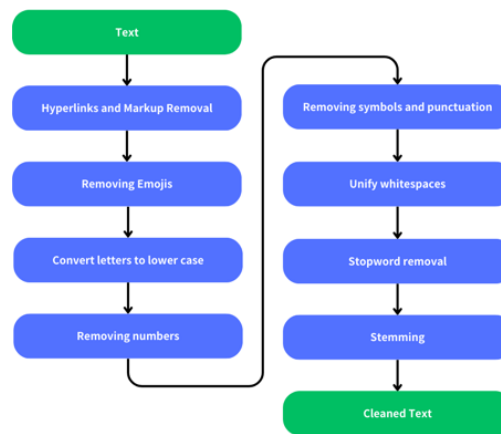


Figure 4: Procedure of further data cleaning on the cleaned dataset.

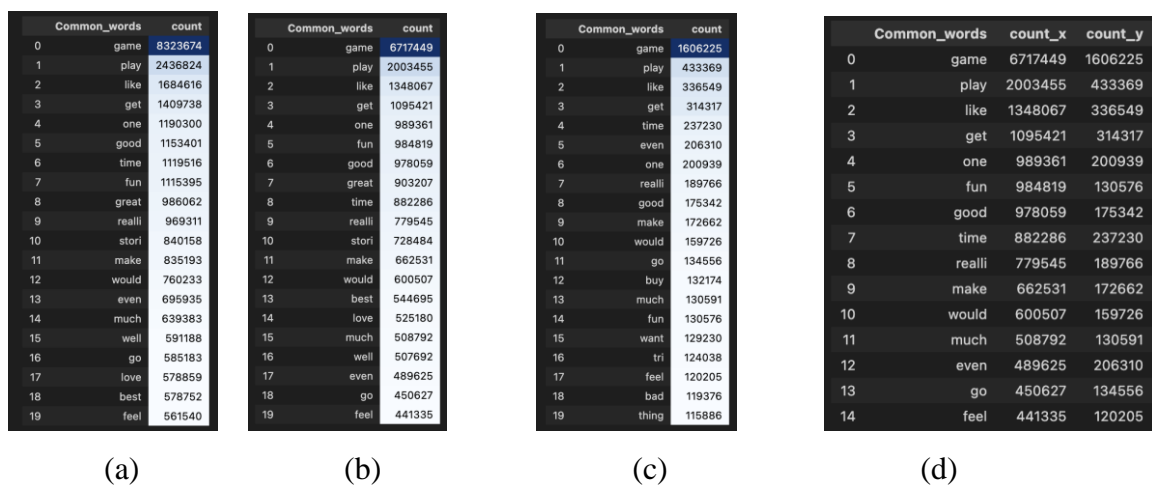


Figure 5: Number of appearances of top 20 frequent words in the cleaned dataset after further data cleaning. (a): Words in the cleaned dataset with both positive and negative reviews. (b): Words in the cleaned dataset with only positive reviews. (c): Words in the cleaned dataset with only negative reviews. (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 301344 reviews with the same number of positive and negative reviews, while the imbalanced dataset contained 925305 reviews with the ratio of number of positive and negative reviews approximated to 5.13: 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 5.14 : 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 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 to 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.

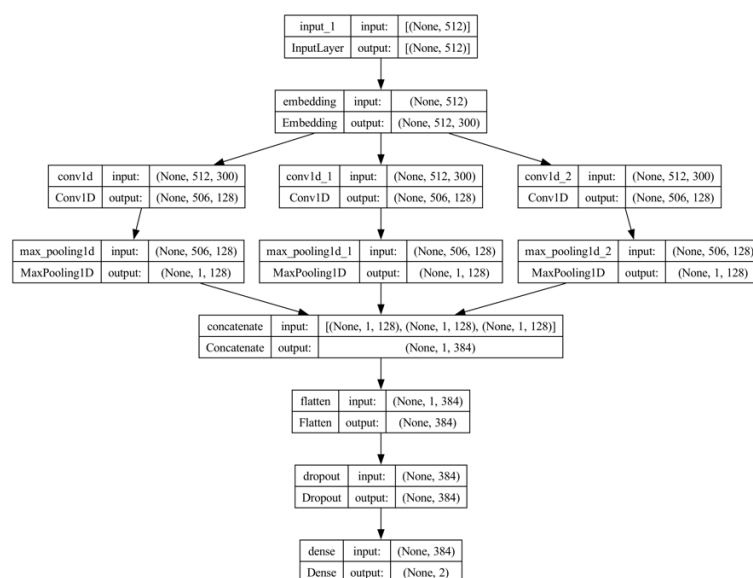


Figure 6: Model structure of CNN in model GloVe-CNN.



Regarding the third model, a training pipeline consisting of a tokenizer and a pre-trained BERT model was implemented. BERT<sub>BASE</sub> was selected instead of BERT<sub>LARGE</sub> 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<sub>BASE</sub> 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, punctuations were 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.

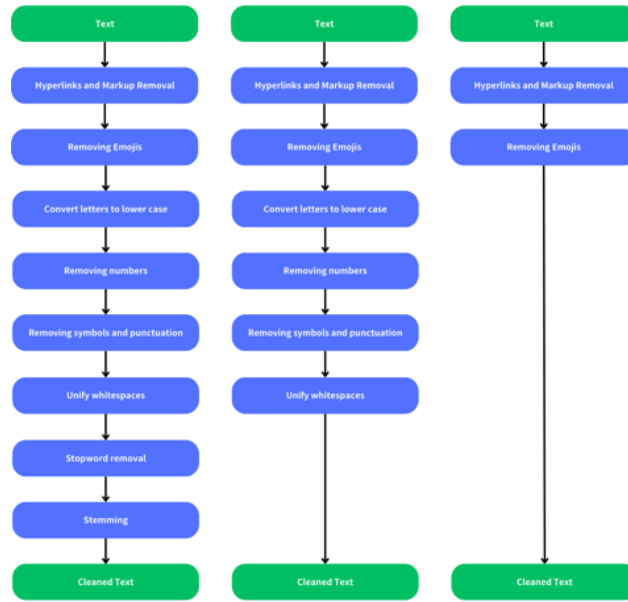


Figure 7: Customized further data cleaning to each model. Left: TFIDF-RF. Middle: GloVe-CNN. Right: BERT

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. 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<sub>BASE</sub> (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 on 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.

To evaluate the performance of different topic models, both quantitative and qualitative approaches were planned to be applied. Regarding qualitative analysis, visualization tools for all three topic modeling techniques were identified. Specifically, pyLDavis

will be used to analyze the result of LDA qualitatively, while similar visualization tools were identified for the remaining two models. Regarding quantitative analysis, measurement in terms of topic coherence and topic diversity will be taken using pre-defined metrics.

### 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 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	PHR	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, train...]	[Audio-Visual Speech Separation and Recognitio...]	[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...]	[crowdsourced, 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, vlsi, 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, or 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.



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

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 building user interfaces. To enhance the appearance and functionality of the user interface.

Second, 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.

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.

### **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 React's *useContext* hook. 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 setup 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 reduce the workload of our virtual machine and improve 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 result 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 a 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.

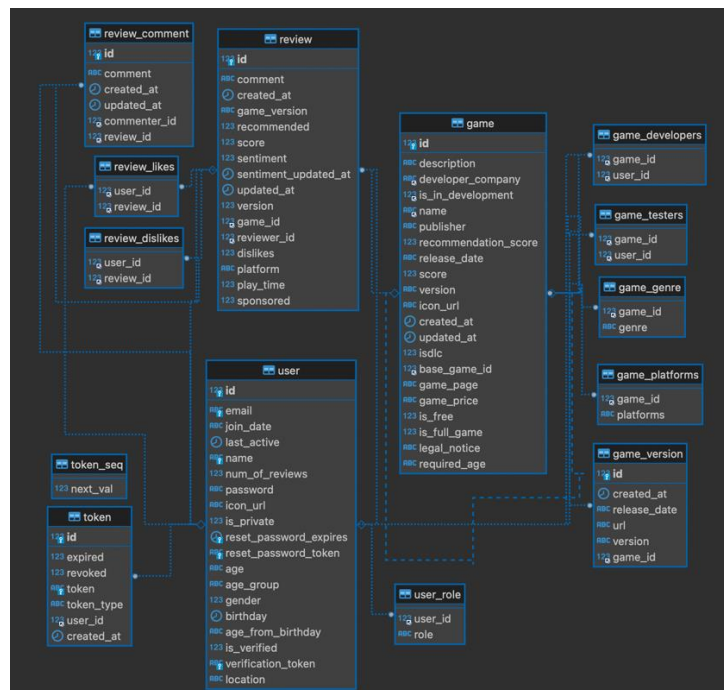


Figure 11 Database Entity Relations Diagram

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 maintains 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 offer 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.

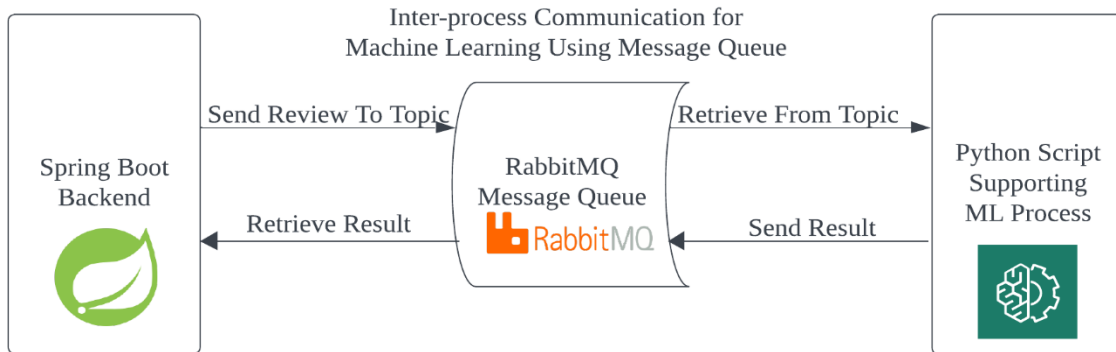


Figure 15 Message Queue Structure for Supporting Inter-process Machine Learning Application

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
109.10.10.10:49170 rabbitConnectionFactory#3555d804:0	FYP	running	o	AMQP 0-9-1	1	0 B/s	0 B/s
109.10.10.10:60836 ?	FYP	running	o	AMQP 0-9-1	1	2 B/s	2 B/s
42.10.10.10:57810 rabbitConnectionFactory#6c8e40fc:0	FYP	running	o	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.

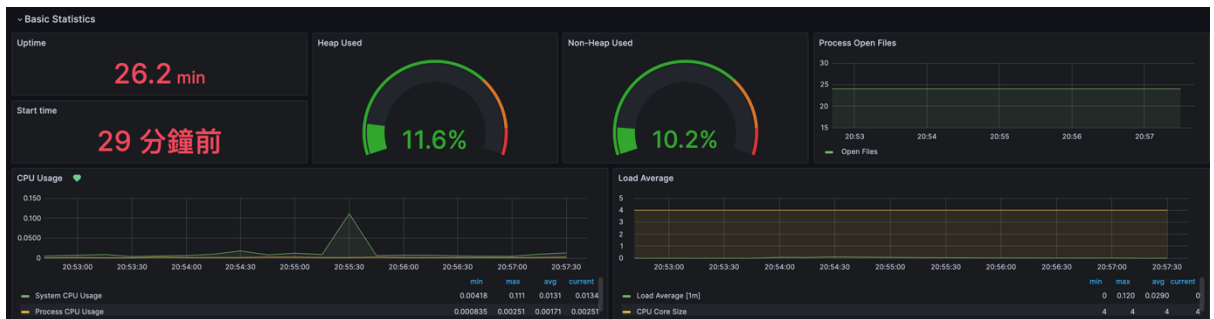


Figure 17 Grafana Dashboard displaying uptime, CPU, and Memory Utilization by the Spring Boot Application.

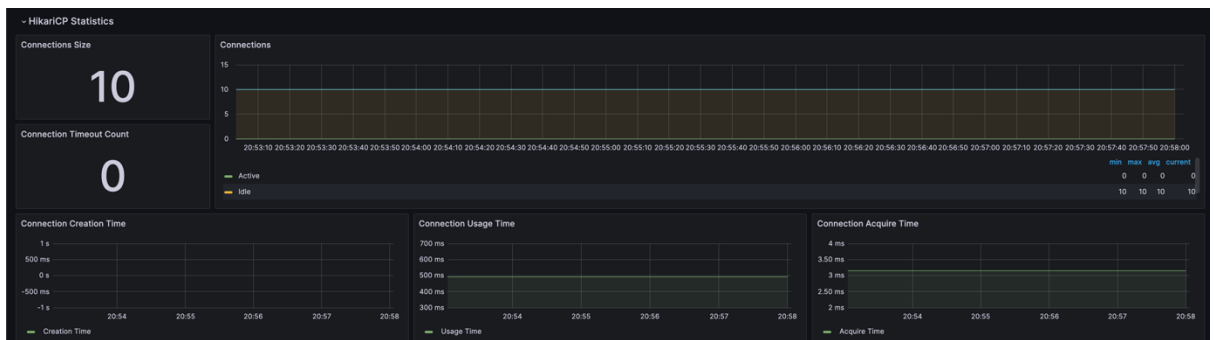


Figure 18 Grafana Dashboard shows the database connection pool size, maintaining a stable connection to the database.

## 4 Current-Stage Result

This section discusses the three main results at the current stage of development: Machine Learning and Natural Language Processing (Section 4.1), Frontend Web Applications (Section 4.2), and Backend Technologies (Section 4.3).

### 4.1 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.

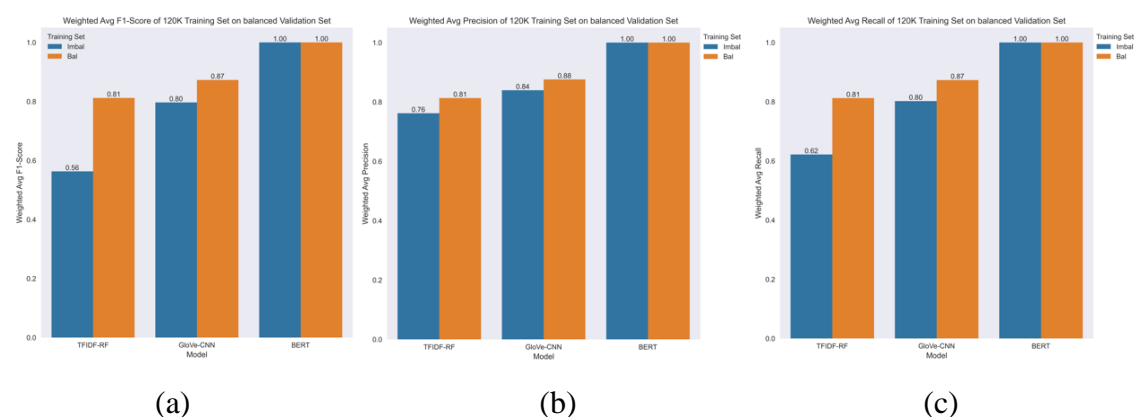


Figure 19: Results of all models trained with 120K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.

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.

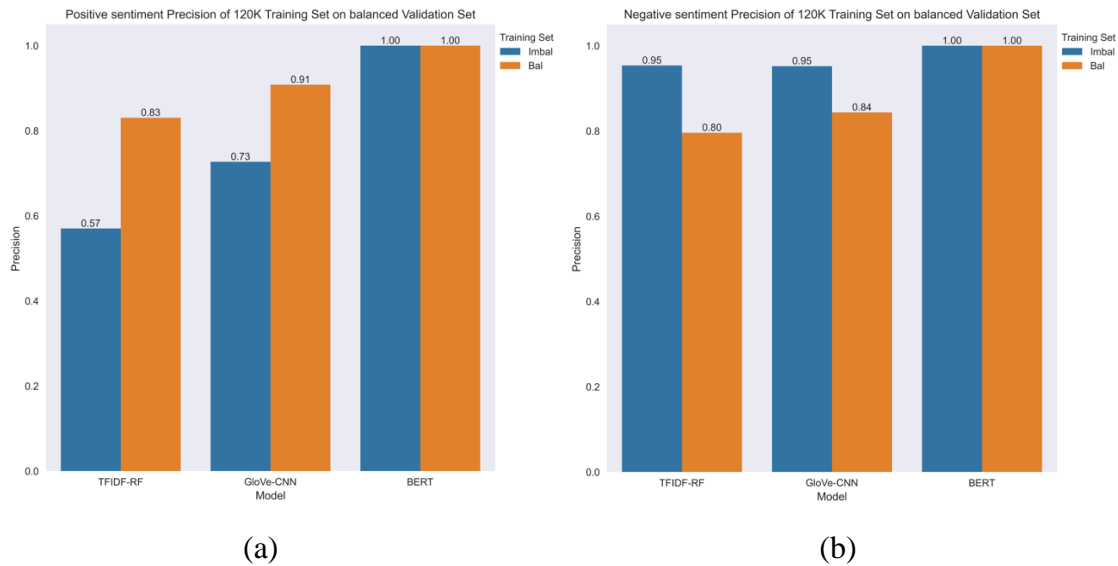


Figure 20: Precision of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative.

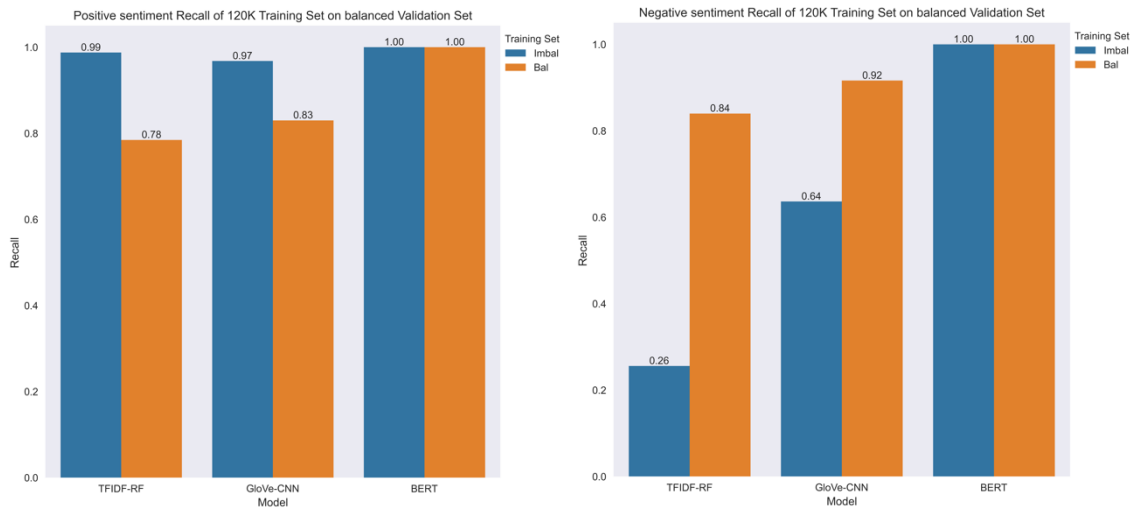


Figure 21: Recall of both sentiments on balanced validation set by models trained with 120K imbalanced and balanced datasets. (a): Positive. (b): Negative.

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.

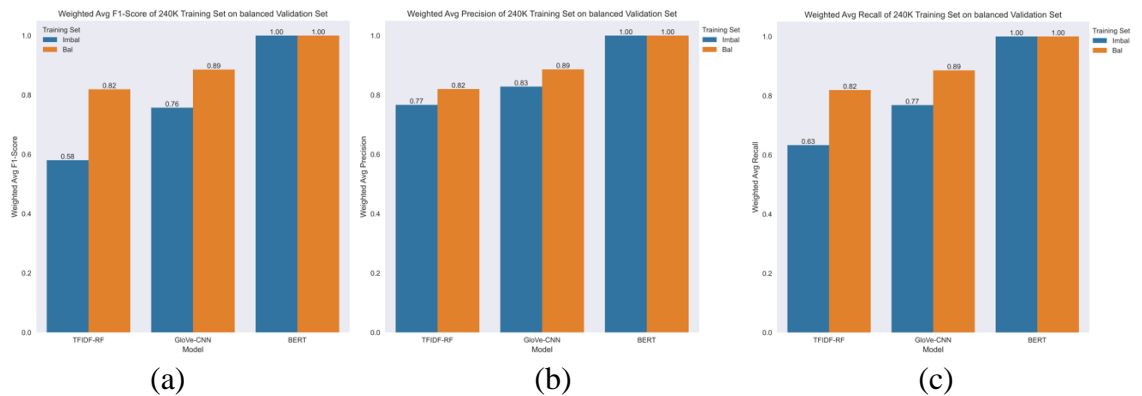


Figure 22: Results of all models trained with 240K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.

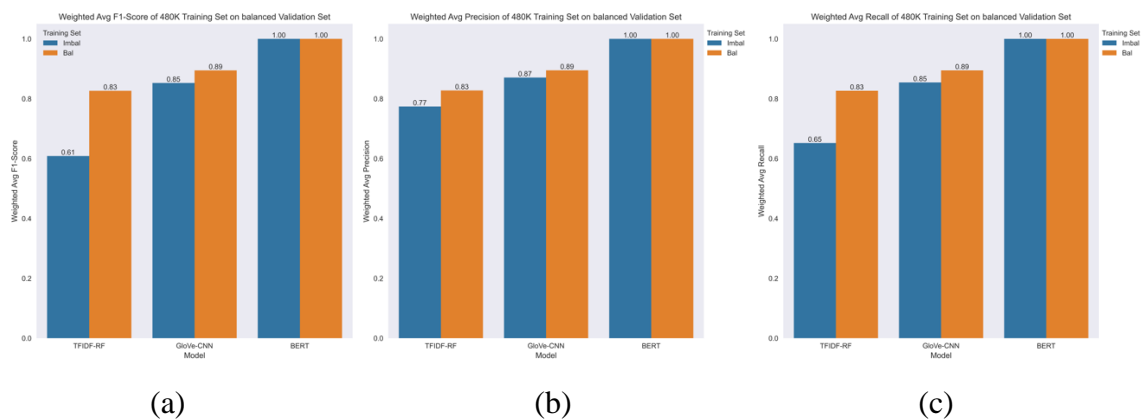


Figure 23: Results of all models trained with 480K balanced dataset. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models.



**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)).

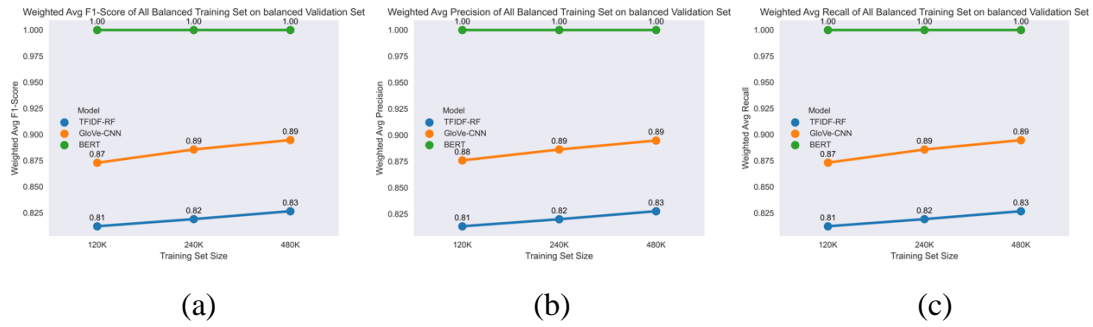


Figure 24: Results of all models trained with balanced datasets of all three sizes. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models

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)).

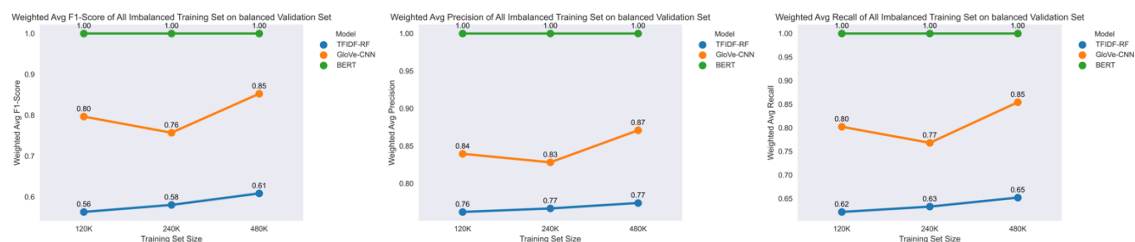


Figure 25: Results of all models trained with imbalanced datasets of all three sizes. (a): Weighted Average F1-score on all models. (b): Weighted Average Precision on all models. (c): Weighted Average Recall on all models

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	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>

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	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>	<b><u>1.00</u></b>

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 performant model. Its ROC curves on both imbalanced and balanced validation sets are displayed in Figure 27.

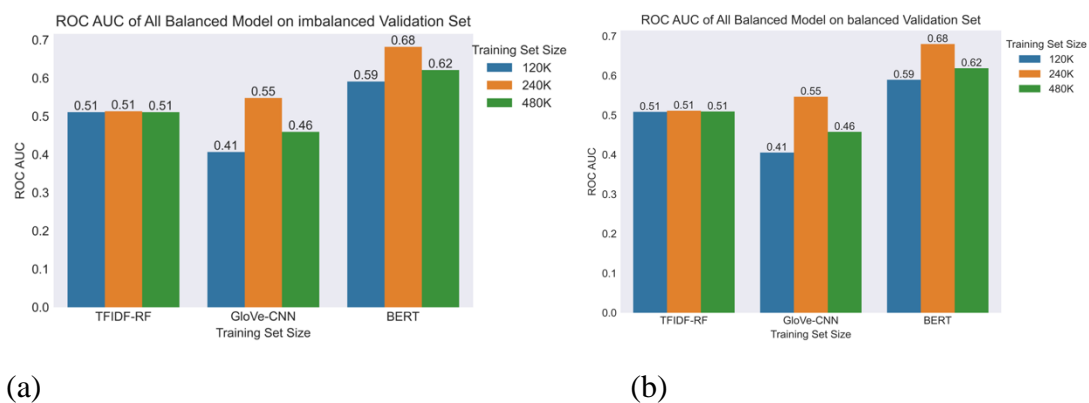


Figure 26: ROC-AUC of all models trained with the balanced dataset. (a): on the imbalanced validation set. (b): on the balanced validation set.

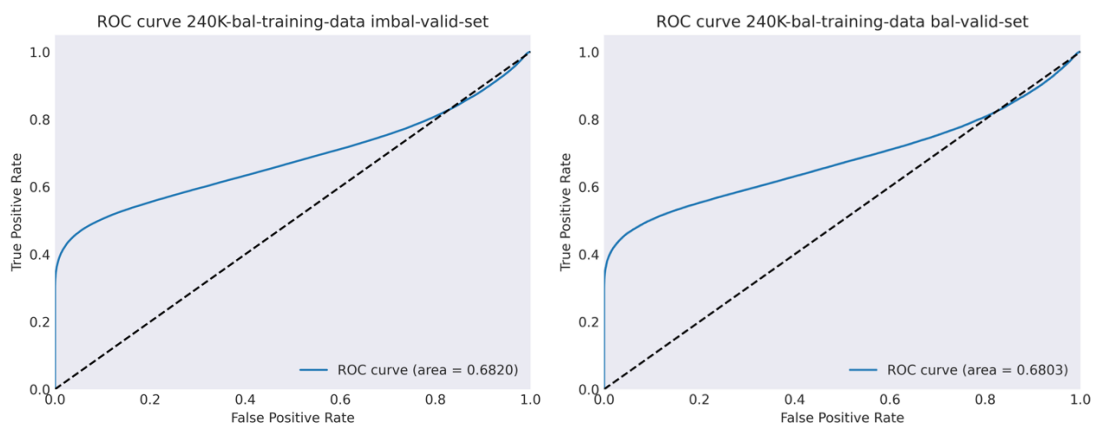


Figure 27: ROC curve of BERT fine-tuned on 240K balanced training set. (a): on the imbalanced validation set. (b): on the balanced validation set.

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.

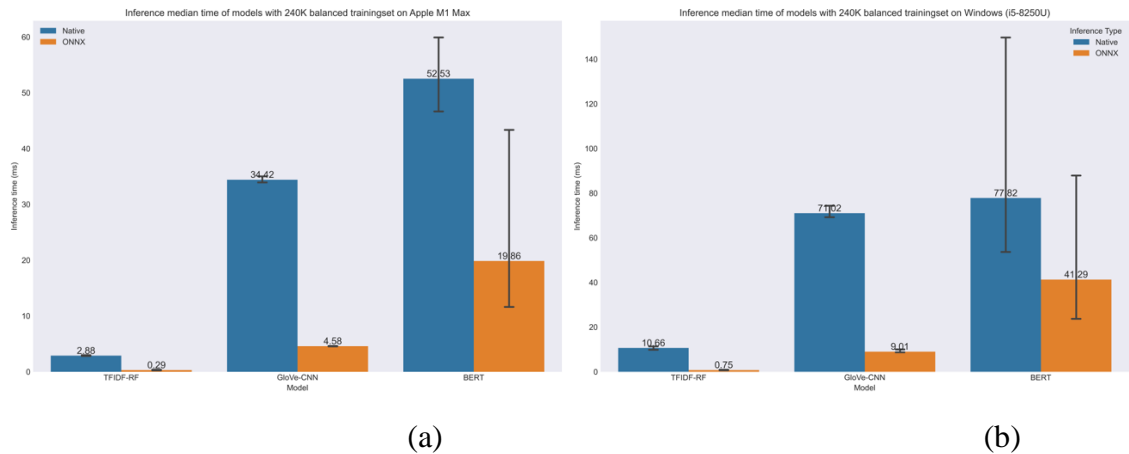


Figure 28: Median inference time of original and ONNX model on different machines. (a): Machine 1 (Windows i5-8250U). (b): Machine 2 (Apple M1 Max)

	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.

## 4.2 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 in this stage 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), search results page (Section 4.2.4), game page (Section 4.2.5) and review page (Section 4.2.6). The following sub-sections will discuss and explain the mentioned pages and components in detail. The current stage of the development has implemented the responsive web design for mobile viewport only for the toolbar and the search result page. The responsive design of other pages will be gradually implemented in the future.

### 4.2.1 Toolbar

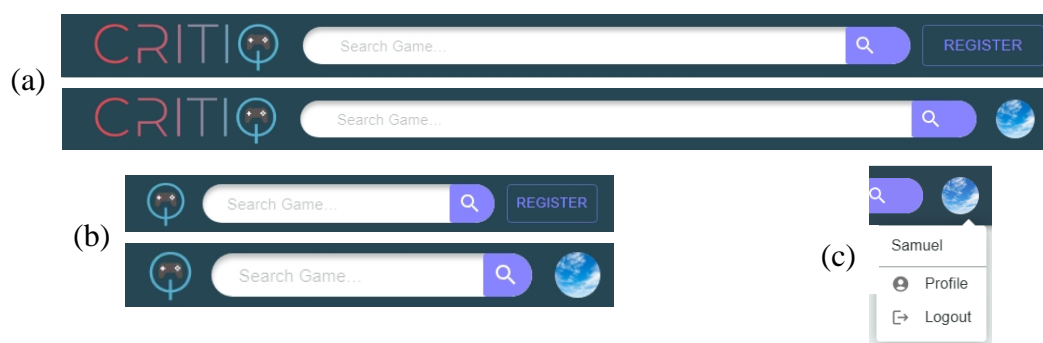


Figure 29 Web application's toolbar design. (a): Toolbar design for desktop viewport. (b): Toolbar design for mobile viewport. (c): Avatar icon button drop down menu.

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 29).

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.

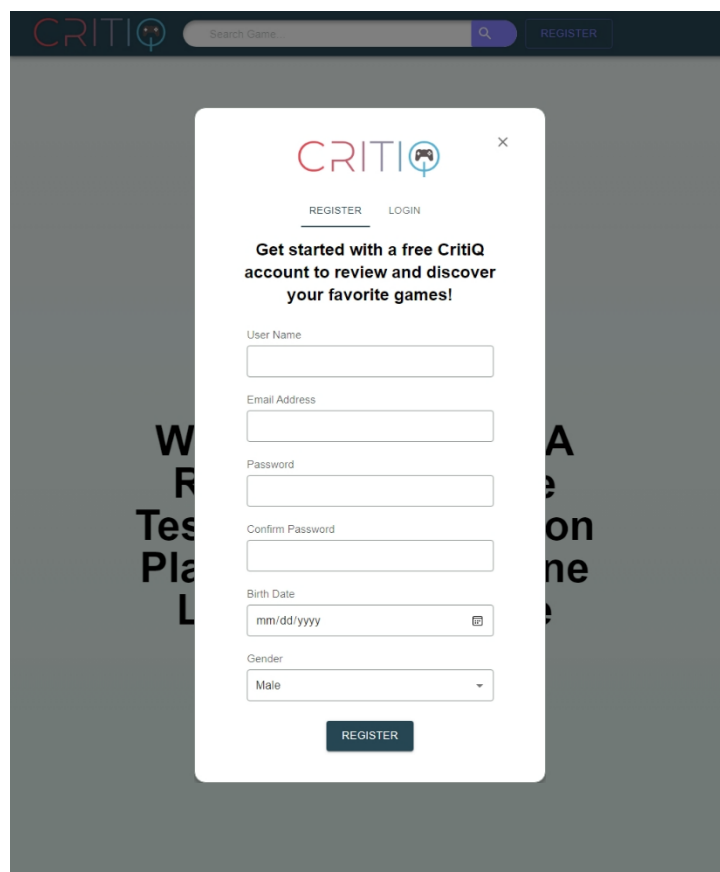
Finally, on the right, there is either a register button or a user avatar button, depending on the user's login status. The register button opens a popup modal that enables the user to create a new account or sign in to their existing one. The details of the popup modal's implementation, features, and design will be discussed in Section 4.2.2. The user avatar

button opens a menu (see Figure 29c) 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 29(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



The image shows a registration modal for the CritiQ web application. The modal is white with a dark border and a close button (X) in the top right corner. It features the CritiQ logo at the top, followed by 'REGISTER' and 'LOGIN' links. Below this is a promotional message: 'Get started with a free CritiQ account to review and discover your favorite games!'. The form includes the following fields: 'User Name' (text input), 'Email Address' (text input), 'Password' (text input), 'Confirm Password' (text input), 'Birth Date' (calendar icon and 'mm/dd/yyyy' text input), and 'Gender' (dropdown menu with 'Male' selected). A dark blue 'REGISTER' button is positioned at the bottom center of the form.

Figure 30 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 31 Register modal input validations

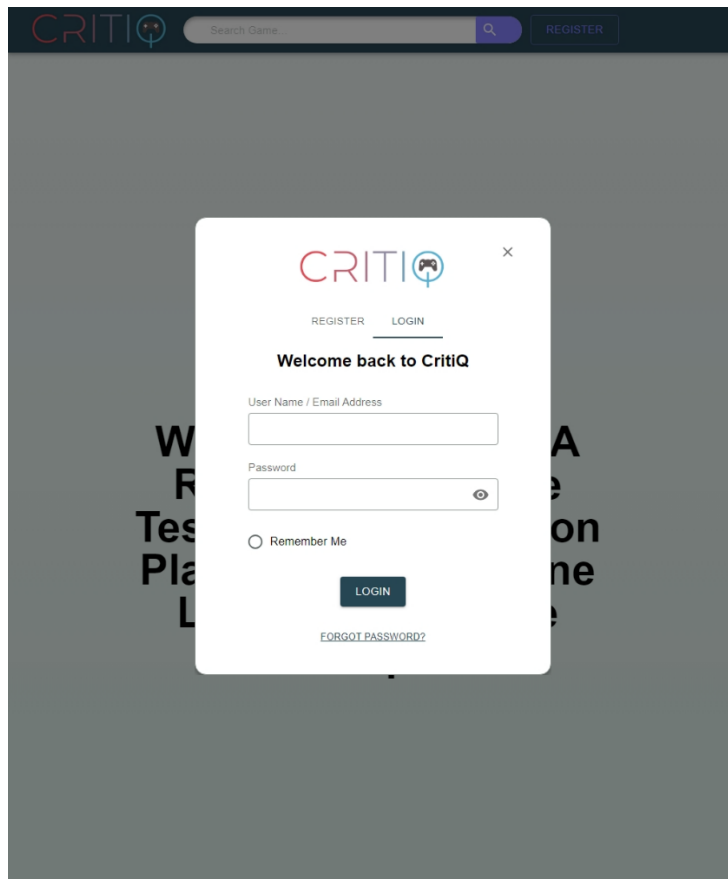


Figure 32 Web application's login modal popup

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.

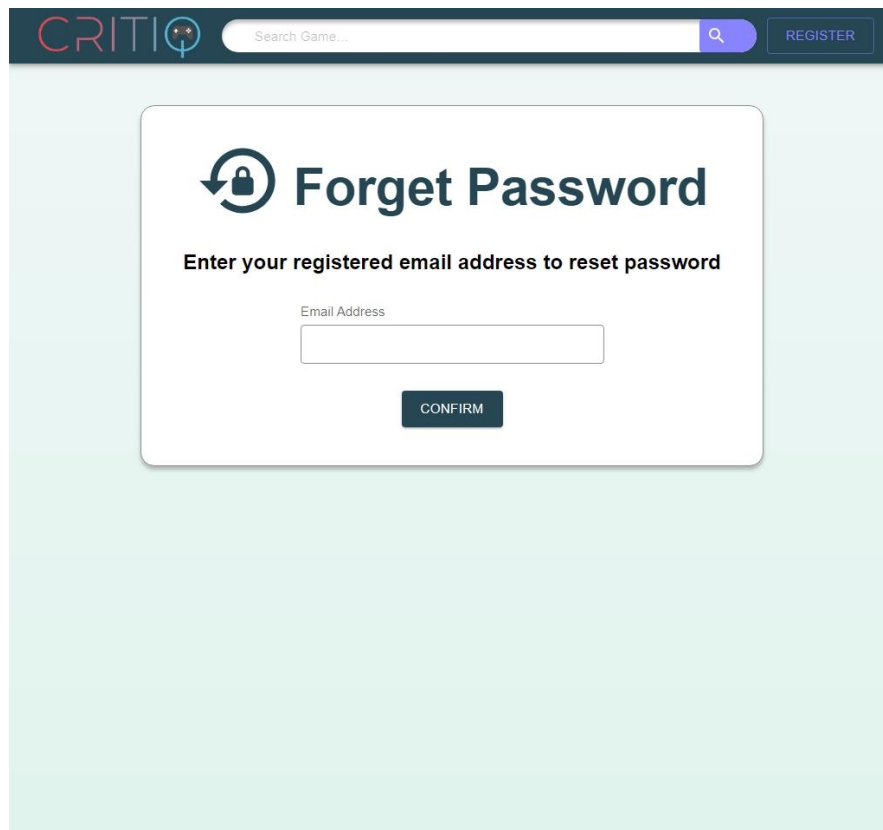
The register modal (See Figure 30) requires the users to enter all the fields to create a new account, which comprise username, email address, password, confirm password, birth date and gender. The front-end application performs validations on all the input fields using regular expressions (See Figure 31) 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 32) 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 blue navigation bar containing the 'CRITIQ' logo, a search bar with the placeholder text 'Search Game...', and a 'REGISTER' button. The main content is centered in a white rounded rectangle. It features a circular icon with a padlock and a refresh symbol, followed by the heading 'Forgot Password'. Below the heading, the instruction 'Enter your registered email address to reset password' is displayed. A text input field is provided for the email address, with the label 'Email Address' above it. A dark blue 'CONFIRM' button is positioned below the input field.

Figure 33 Web application's forgot password page design.

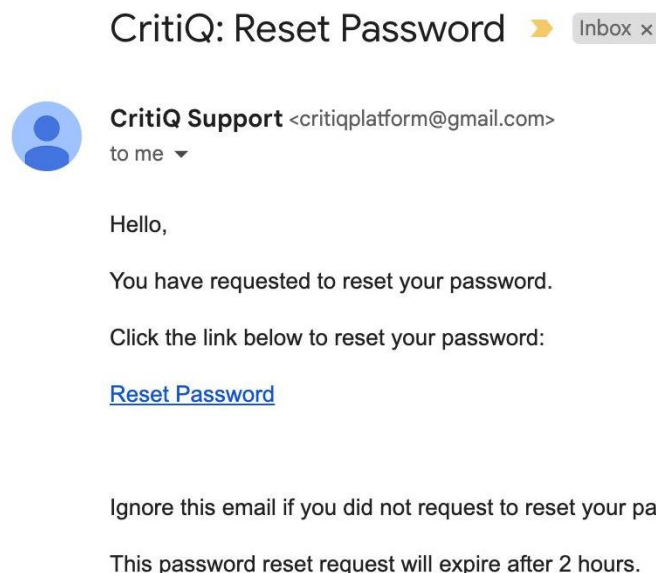
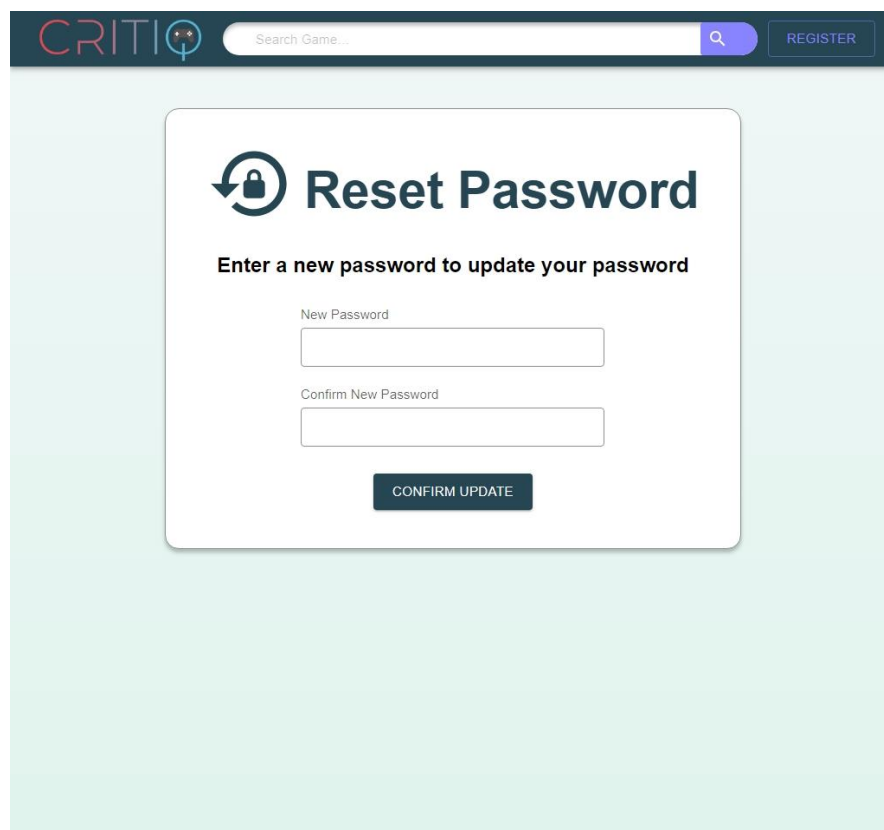


Figure 34 Reset password email

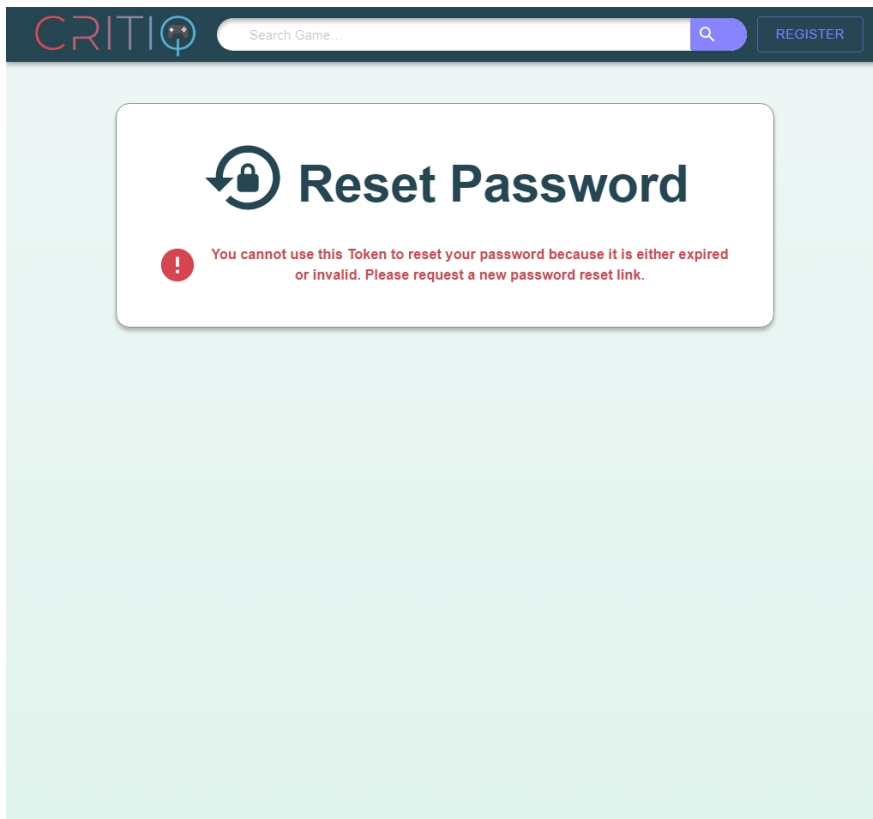
The user can access the forgot password page (See Figure 33) by clicking the forgot password button in the login page. On this page, the user can input the email address associated with their account. Validations will be performed by both the frontend application and the backend system. The frontend 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 34) 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 with the placeholder text 'Search Game...', 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 lock 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 35 Web application's reset password page design*



*Figure 36 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 35) 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 36). 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.

## 4.2.3 Search Result Page

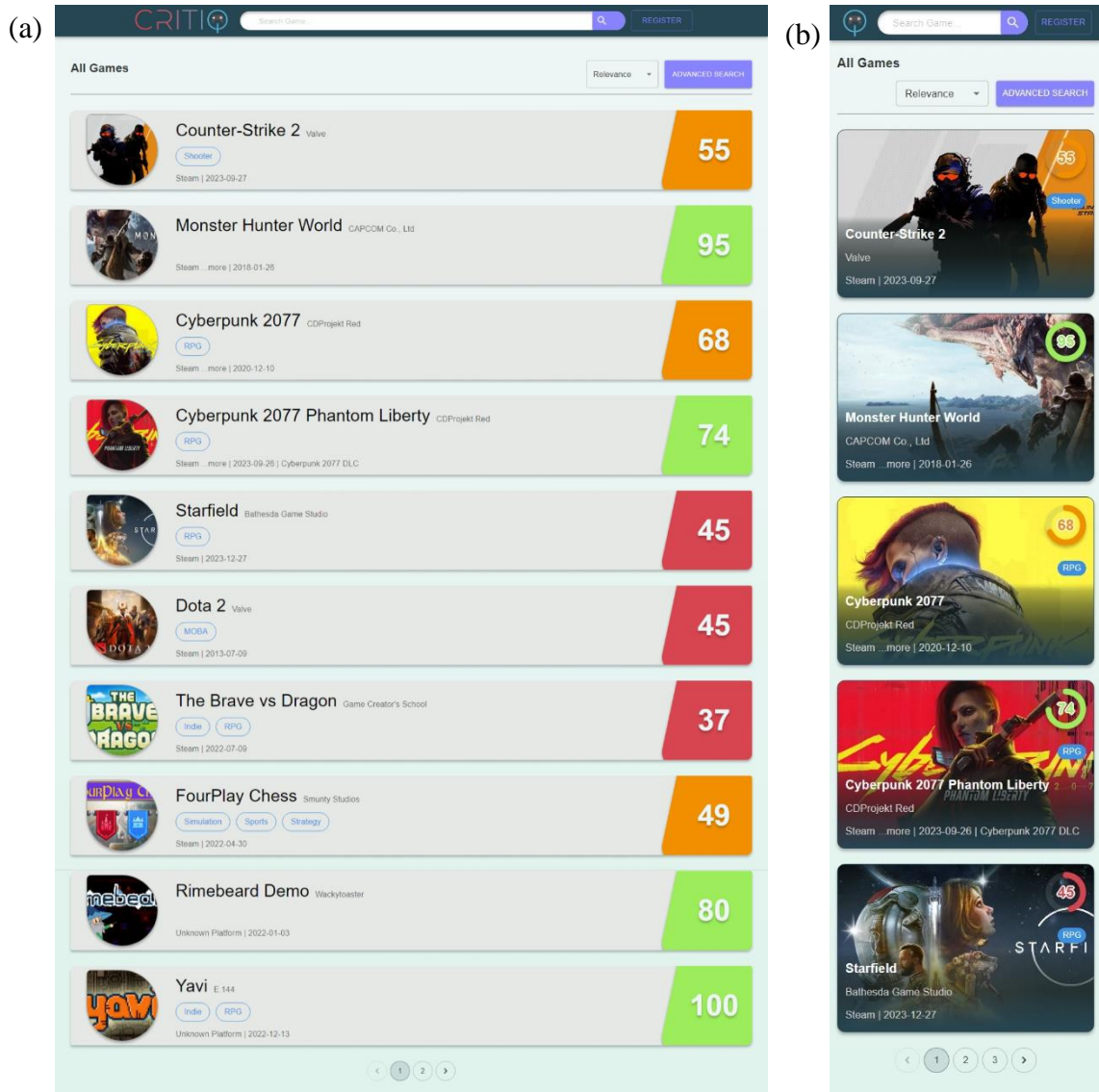


Figure 37 Web application's search result page design. (a): Search result page design for desktop viewport. (b): Search result page design for mobile viewport.

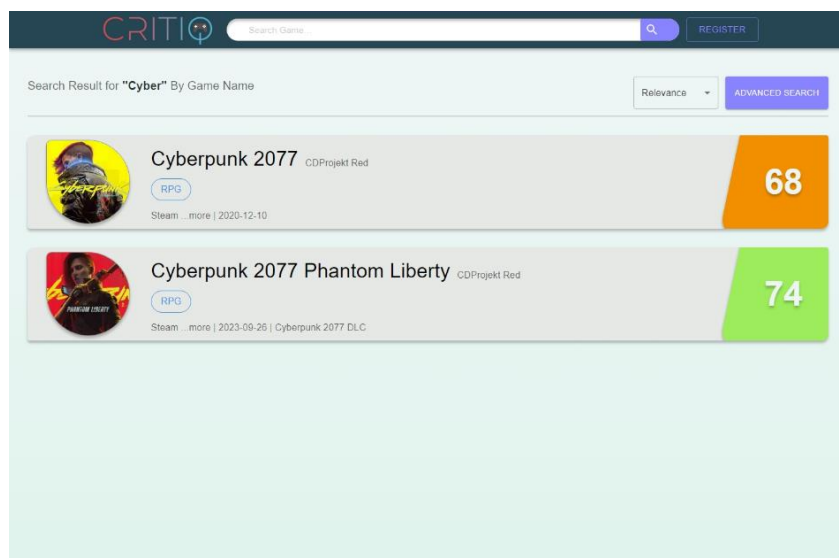


Figure 38 Search result page searching example

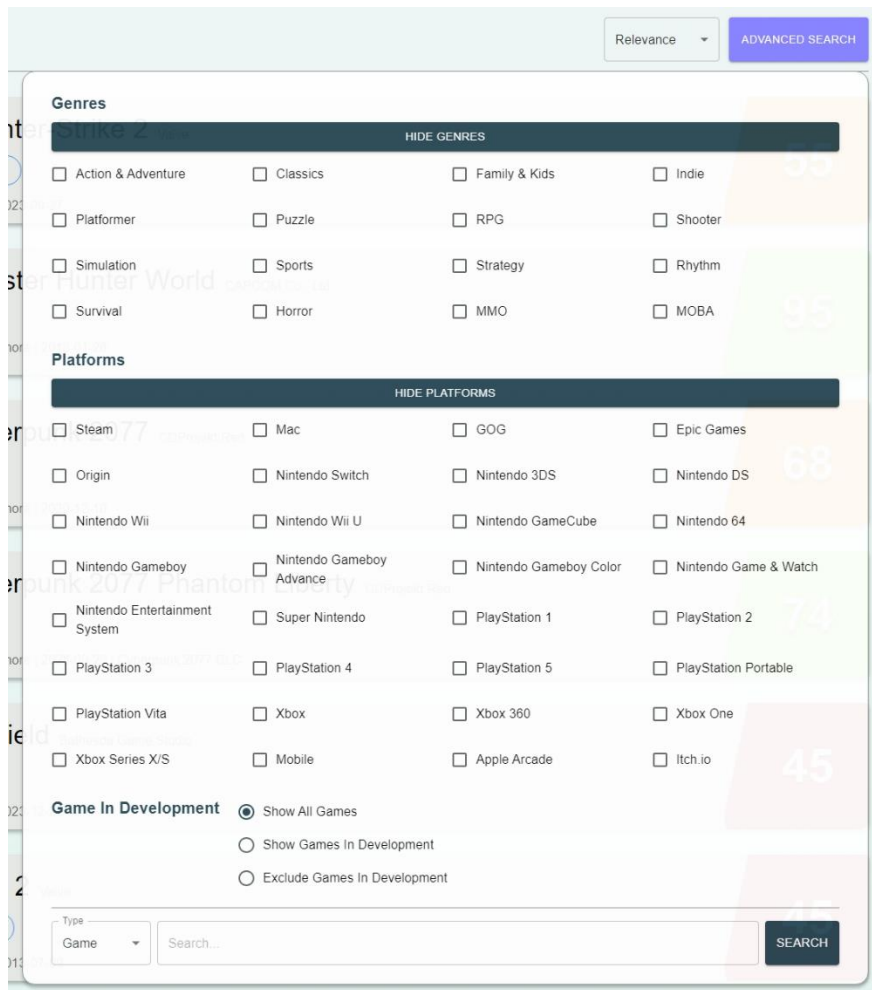


Figure 39 Advanced search modal for search result page

Section 4.2.1 describes how the user can access the search result page via the search bar and the search button in the toolbar. 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 37 (a)). 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 38). 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 39) 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.

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.

#### 4.2.4 Game Page

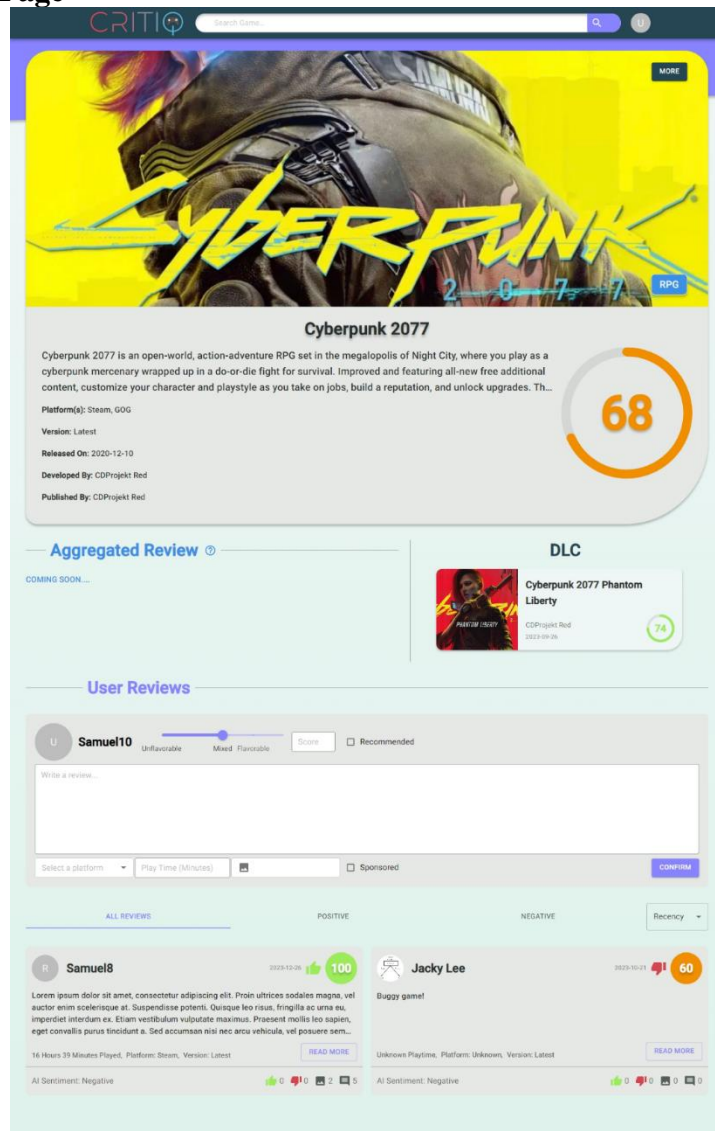
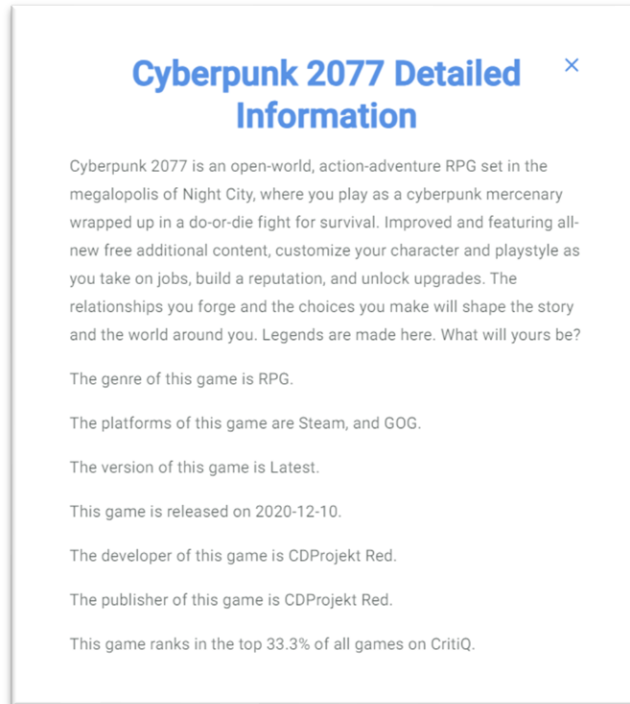


Figure 40 Web application's game page design





*Figure 41 Game detailed information popup modal*

As stated in the preceding section, the game page (See Figure 40) can be accessed by selecting the search result card on the search result page and include three main sections, Information, Add Review and All reviews.

The information section, located at the top of the game page, displays all the game information that was previously presented on the search result card. Moreover, 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, and the overflowed text will be concealed by ellipsis. To access the full description of the game, the user can click on the more button located on the top right of the information section, which will open the game detailed information modal (See Figure 41).

The information section is followed by the aggregated review section and the DLC section. The aggregated review section is currently under development. It will utilize natural language processing models to generate an aggregated review that summarizes all the reviews of the game for the user, and display the summary information on this section. The DLC section will only be visible for games that have DLC (See Figure 40), 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.

Following the aggregated review section and the DLC section is the user review section. This section consists of two sub-sections, which are the add review section and the game review section.

The add review section is only visible to authenticated users. Users can create new reviews by completing the input fields in the add review form. 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.

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 all, negative-only, or only positive-only reviews by selecting the respective tab in 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. The game review cards, which are components that present some essential information about the game review and the reviewers, are situated below the tab bar and the select button. 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. 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.

#### 4.2.5 Review Page

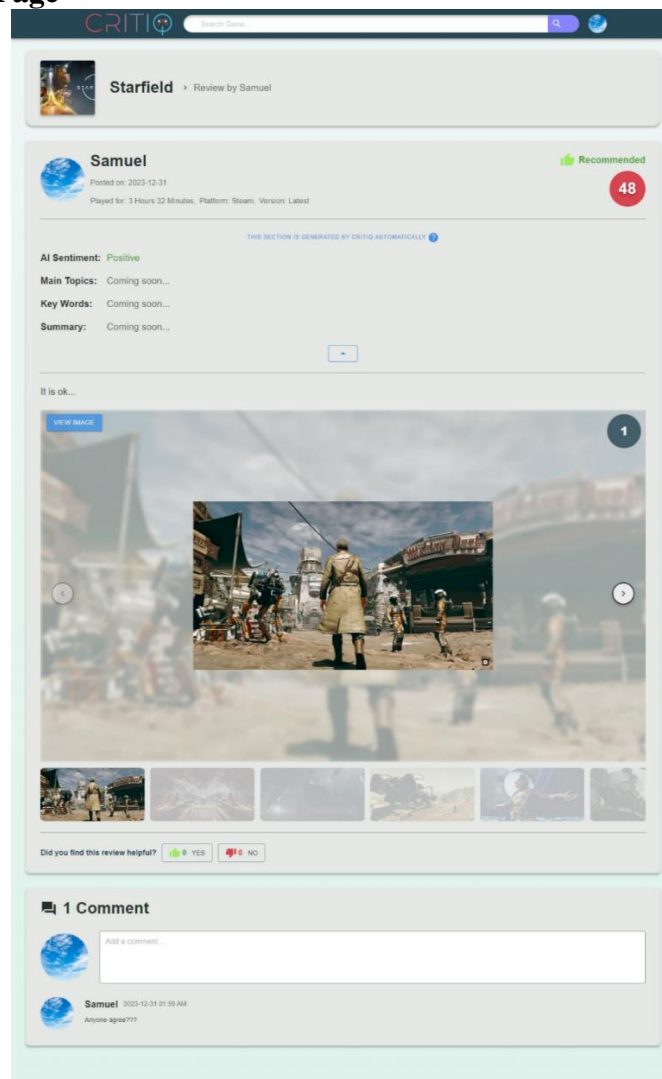


Figure 42 Web application's review page design

This section presents the review page, which was introduced in the previous section. 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 42).

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. This information within the section is generated automatically using the results of the NLP models, and the user can toggle this section by clicking on the up or down arrow button. At this stage of development, only the sentiment analysis model is implemented, and the sentiment result is displayed in this section. In the future, the results of the topic modelling and keyword extraction model will be added to this section as well. 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 offers advanced features such as responsive layout based on viewport and thumbnails navigation. 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 4.2.3).

### 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 43). These data were 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
Bethesda Game Studio	0	Starfield	Bethesda	[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
Paweł 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 43 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 44).

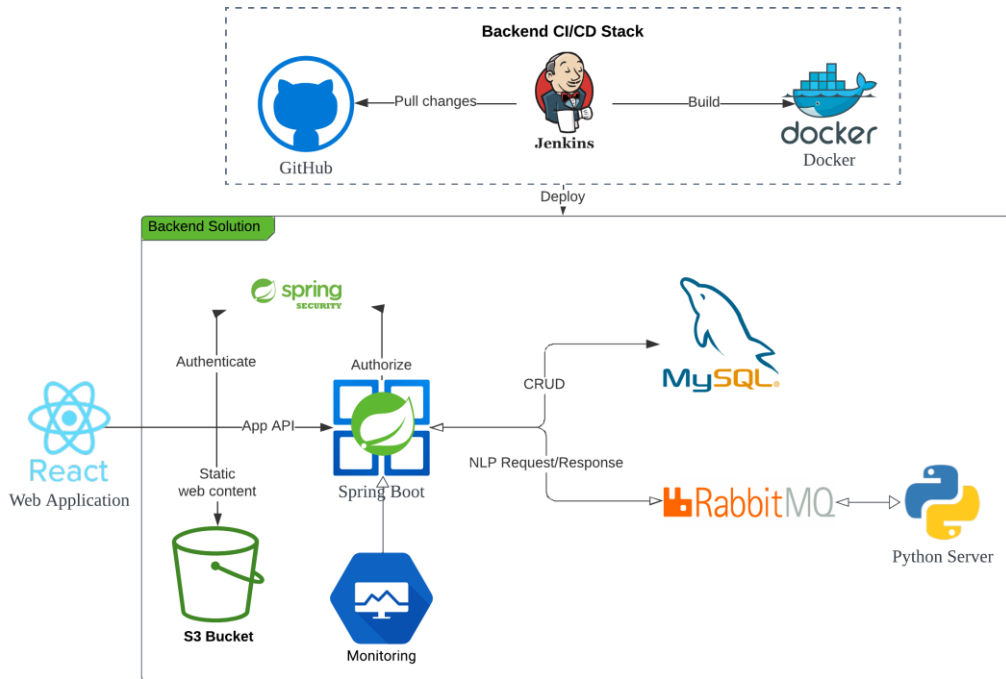


Figure 44 Backend Solution Architecture Graph

#### 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.

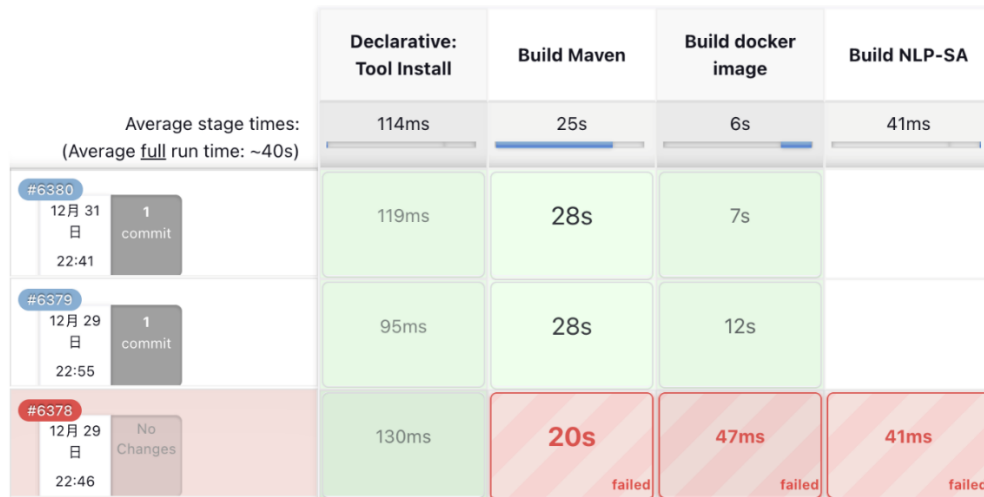


Figure 45 Jenkins deployment User Interface with different stages of deployments.

By using a Custom CI/CD pipeline (See Figure 45), 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 45).

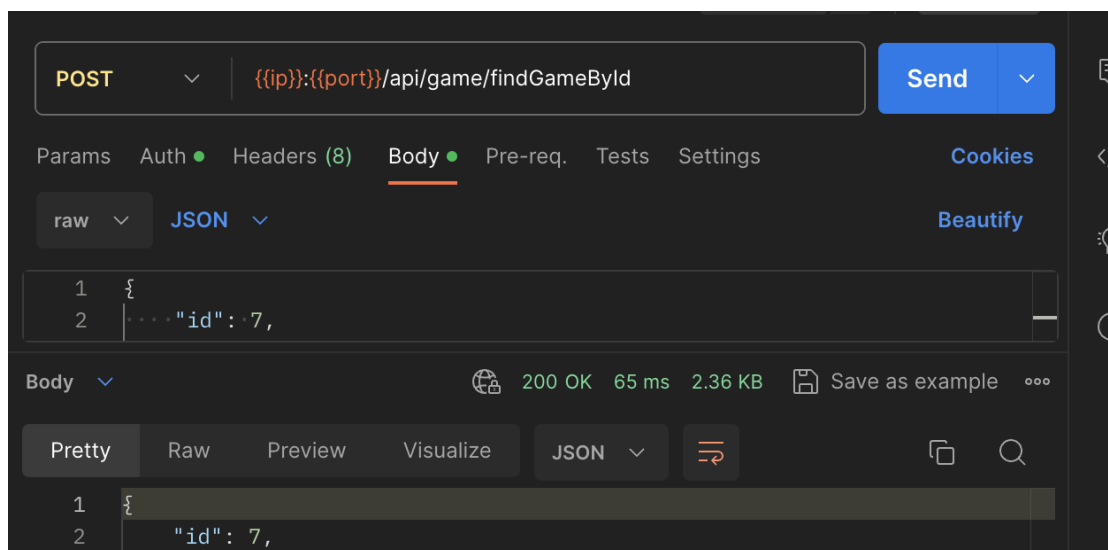


Figure 46 API call to `/findGameById` to fetch specific game information finishes in 65ms

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 47).

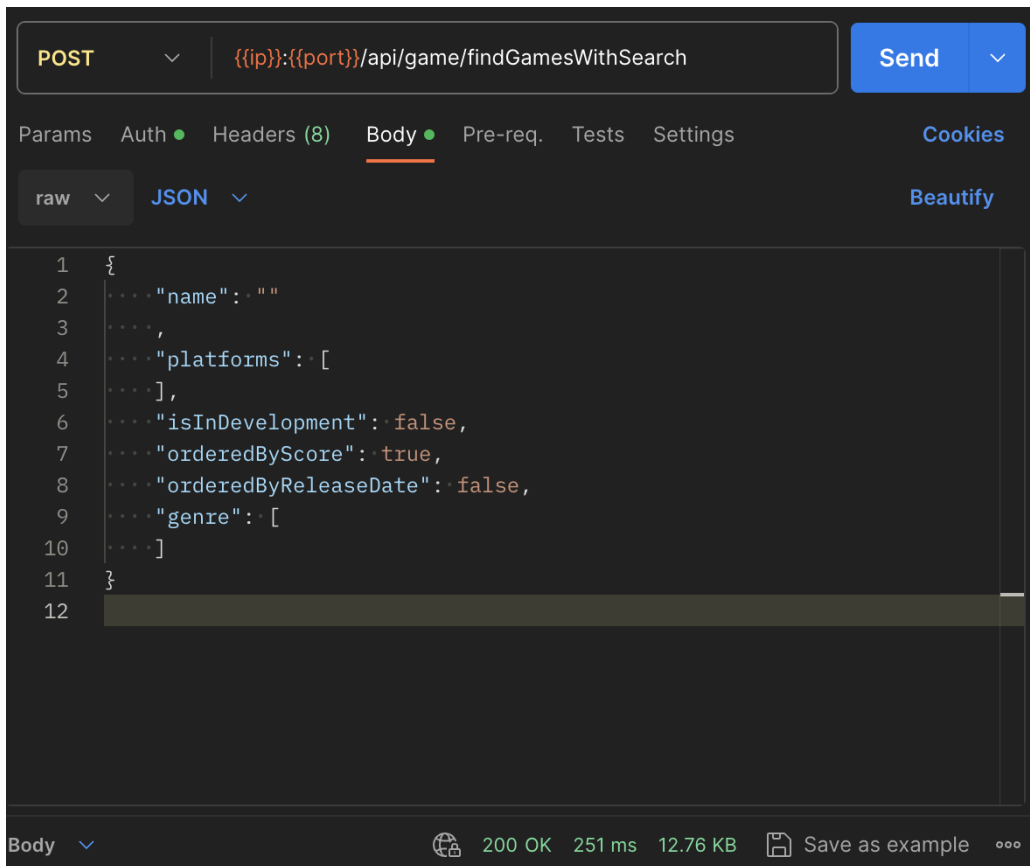


Figure 47 API call to /findGamesWithSearch to perform exhaustive game search finishes in 251ms

Our database plan offers a connection limit of 150 concurrent connections from any location (See Figure 48). To improve the performance of complex queries, including Advanced Game Search or Data Analysis, setting a larger connection size from Spring Boot had improve the query times by over 100%. Further testing have shown that 70-100 concurrent connection to the database in the production environment yielded the best result without utilizing too much connections (See Figure 49). 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 48 Current Database Plan with 2GB RAM offer a maximum of 150 concurrent connections

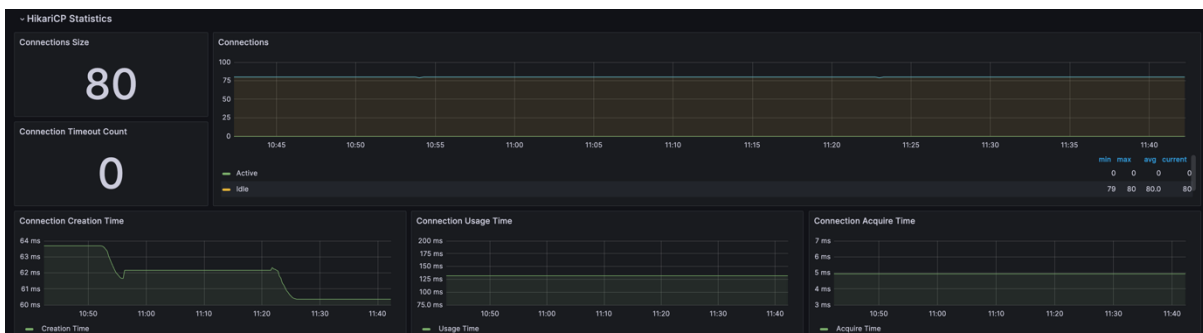


Figure 49 Spring Boot utilizes 80 concurrent connections to the database

### 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 50).

```
@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");
    }
}
```

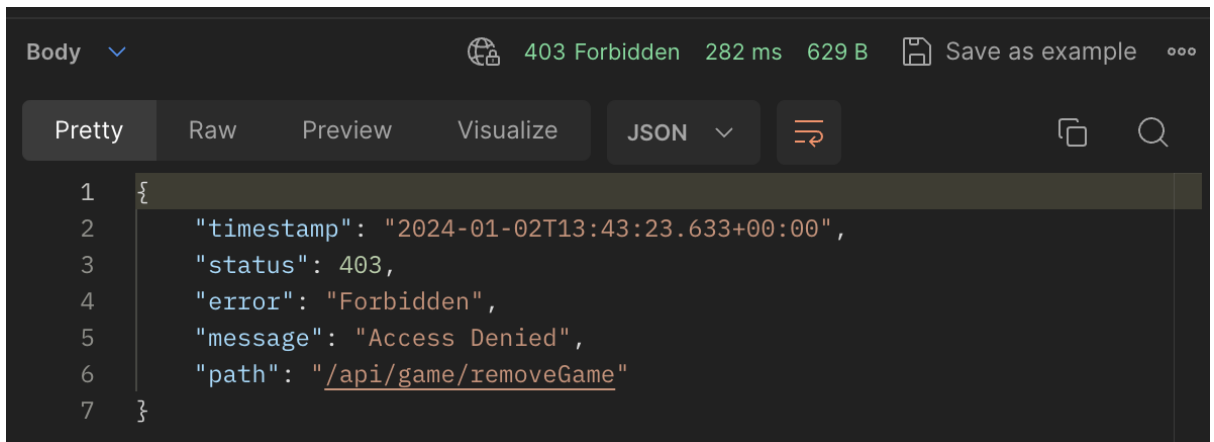
Figure 50 Access Control by verifying the user based on the JWT sent in HTTP requests using the `@AuthenticationPrincipal` annotation.

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 51).

```
@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 51 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 52)



```
Body ▾ 403 Forbidden 282 ms 629 B Save as example ⋮
Pretty Raw Preview Visualize JSON ▾
1 {
2   "timestamp": "2024-01-02T13:43:23.633+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "Access Denied",
6   "path": "/api/game/removeGame"
7 }
```

Figure 52 403 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 53).

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 53 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 purpose and tracking (See Figure 54) while the uploader for files uploaded by the Backend system will be marked as “System”.

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

Figure 54 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 55). 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.

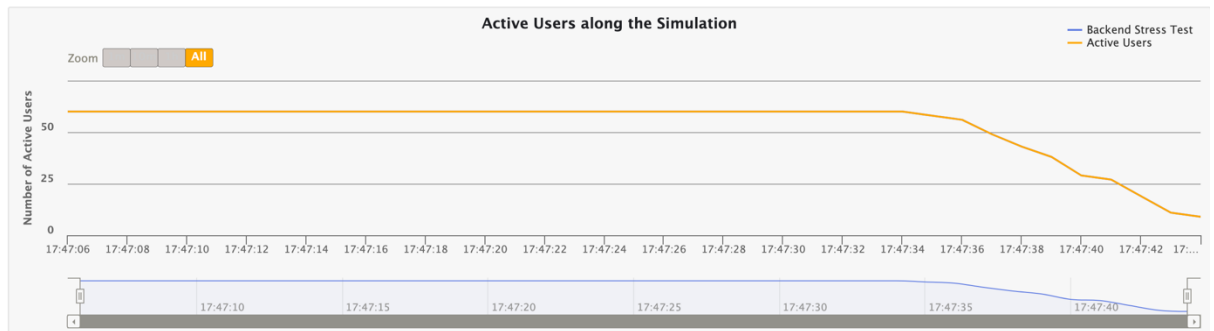


Figure 55 Backend Load-Testing using Gatling, showing the ability to sustain 60 active users performing complex queries.

## 5 Difficulties Encountered

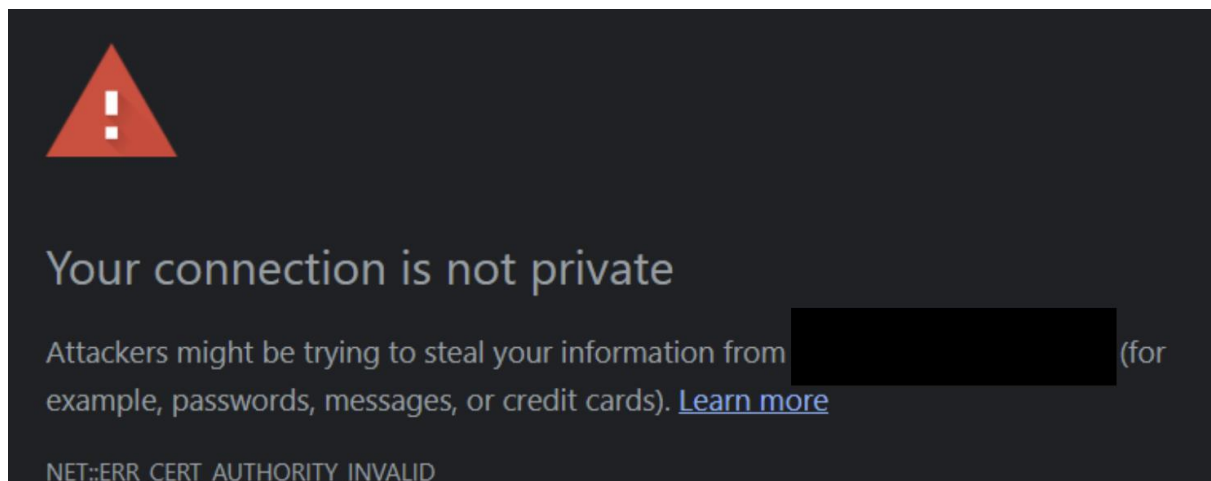
This section discusses the main difficulties encountered, including HTTPS and SSL certificates (Section 5.1) and Message Queue Disconnection (Section 5.2).

### 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 56).



*Figure 56 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 PKCS12 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 57).

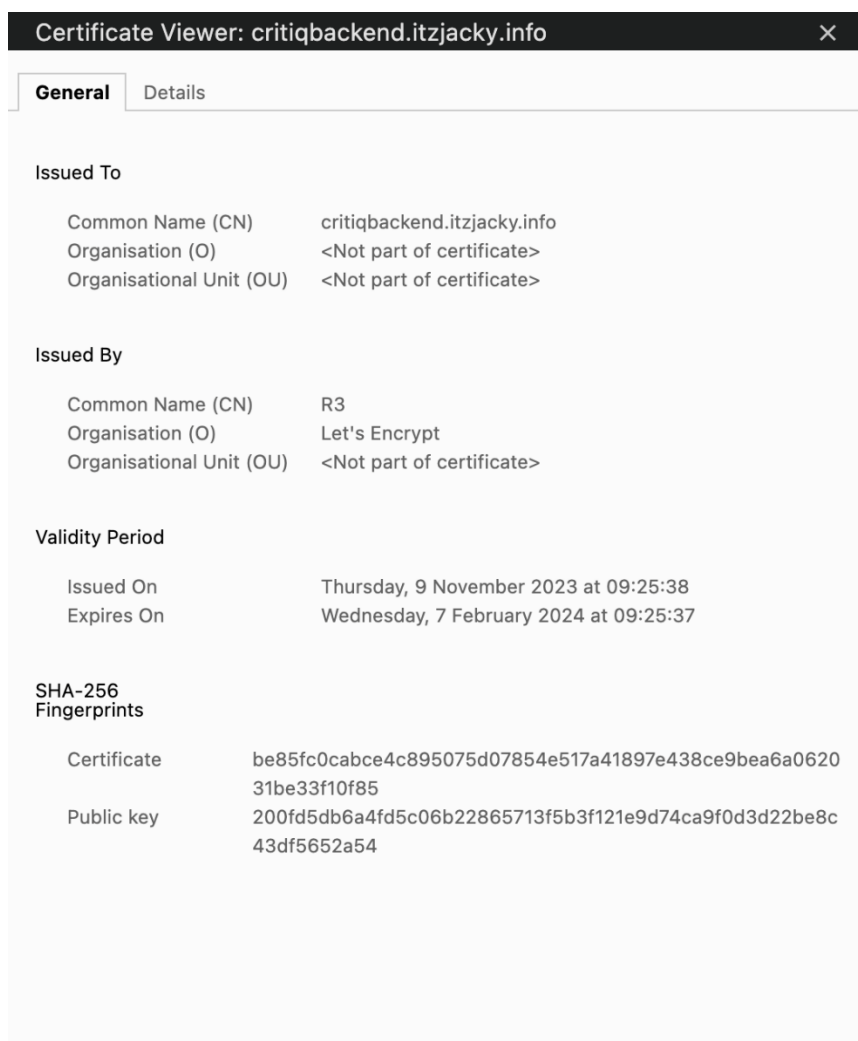


Figure 57 Certificate Viewer on Backend Domain Address using Google Chrome



## 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 both the Python client and 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 53). 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 58: Code snippet to initiate the RabbitMQ connect with heartbeat check disabled.

## 6 Proposed Schedule

The progress of the project is currently on schedule. The schedule below for the project is contingent upon the current rate of progress (See Table 6).

Period	Work to be done
Jan – Feb	<ul style="list-style-type: none"> <li>- Integrate the Sentiment Analysis model into our application.</li> <li>- Implement Landing Page and Game Dashboard Page</li> <li>- Implement Topic Modelling Model</li> <li>- Implement Keyword Extraction model</li> </ul>
Feb – Mar	<ul style="list-style-type: none"> <li>- Fine-tuning Topic Modelling Model</li> <li>- Fine-tuning Keyword Extraction model</li> </ul>
Mar – Apr	<ul style="list-style-type: none"> <li>- Fine-tuning of all models and their integration.</li> <li>- Debug and Refactor Code</li> <li>- Prepare for the Final Presentation</li> <li>- Prepare Demo and Demo Video</li> </ul>

*Table 6 Proposed Schedule for the project*

## 7 Work Distribution

The contribution of each member in the project is listed in Table 7.

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

*Table 7 Work Distribution Table of the project*

## **8 Conclusion**

Advancement in NLP is evolutionary, yet rarely applied in the field of gaming and game reviews. This project sets out to develop a web application that harnesses the power of NLP to enable and empower developers through automatic review analysis and aggregation. Early-stage results demonstrated the potential of adopting NLP into the gaming industry, particularly in review analysis.

The preliminary outcomes have resulted in a procedural iteration of the web application, which has been designed to incorporate an essential game review feature and automated sentiment analysis feedback. The backend solution is presently hosted on cloud platforms, employing a fully realized CI/CD pipeline to accurately reflect alterations made to the production environment, thereby facilitating expedited development and deployment processes. In terms of ML in NLP, current-stage empirical results have shown that NLP techniques such as Sentiment Analysis, with careful tuning, can be effectively applied while maintaining high accuracy and performance. This project has proven that, with a well-designed system architecture, NLP tools can be seamlessly integrated into existing systems with minimal cost and hardware requirements, enhancing game developers' and users' insights into the strengths and weaknesses of the game. Future work will include the implementation and evaluation of more NLP features such as topic modelling and keyword extraction.

## 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](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.