The University of Hong Kong

COMP4801 Final Year Project

Coding Learning Platform

Yu Jian Hui (3035790719)

Abstract

A coding learning platform is designed to provide students with a superior learning experience in web development, compared to other platforms. Beyond the courses available on the platform, users also have the opportunity to create their own courses. A range of editing tools, akin to those found in Microsoft Word, are provided to facilitate the creation of course content.

The choice of libraries and framework for building the platform was influenced by their specific advantages and functionalities.

For the front end, the Qwik framework was chosen primarily for its support for server-side rendering, which allows for faster web page loading. Tailwind CSS is employed for styling the web pages.

On the back end, Vercel is used for hosting as it can run servers in multiple locations worldwide, thereby reducing latency. The Turso database is used as it also supports low latency operations by hosting the database globally. Drizzle ORM is used for better data control through object-based database management. Lucia Auth handles user authentication and authorization. A dedicated server, Elysia, is used for password hashing due to Vercel's limited computing power. Elysia, in conjunction with a message queue known as QStash, is responsible for sending verification emails to users.

Acknowledgements

We would like to express our deepest gratitude to our final-year project supervisor, Prof. T. W. Chim, for his invaluable insights and guidance throughout our project. We are also grateful to the Department of Computer Science, the University of Hong Kong, for providing us with this wonderful opportunity.

Table of Figures

Figure 1: W3Schools Home page 2

Figure 2: React documentation 3

Figure 3: An illustration of the working principle of Qwik drawn using Excalidraw. 7

Figure 4: Working principle of SSR 7

Figure 5: Styling using CSS class 1 9

Figure 6: Styling using CSS class 2 9

Figure 7: Styling using TailwindCSS 9

Figure 8: Environmental variables in Vercel 10

Figure 9: SQL SELECT statement 11

Figure 10: SELECT statement using ORM 11

Figure 11: Workflow of user authentication 12

Figure 12: Workflow of user authentication with hashed password 13

Figure 13: Verification Email 14

Figure 14: Workflow of sending verification email 15

Figure 15: First section of the home page 16

Figure 16: Tutorial section 17

Figure 17: Audio-guided course section 17

Figure 18: Audio-guided course section (highlighted) 18

Figure 19: Courses page 18

Figure 20: HTML course details page 19

Figure 21: HTML course chapter 1 19

Figure 22: Live code editor within a course 20

Figure 23: Audio player at the bottom of the course content section 20

Figure 24: Audio-highlighting effect 20

Figure 25: Fill-in-the-blank question in a checkpoint chapter..... 21

Figure 26: Multiple-choice question in a checkpoint chapter..... 21

Figure 27: Steps to sign up for an account..... 22

Figure 28: Dashboard page 23

Figure 29: Dropdown menu when hovering over the icon 23

Figure 30: Profile page..... 24

Figure 31: Creator page with expanded course..... 24

Figure 32: Requesting for the editing page of the chapter..... 25

Figure 33: Editing page of HTML course chapter 1 25

Figure 34: Toolbar on the chapter editing page 26

Figure 35: Button to create a new course..... 26

Figure 36: Steps to create a course 27

Figure 37: Responsive design of the course content page 27

Figure 38: An example of implementation of responsive design 28

Figure 39: Dark mode of the course centent page 28

Figure 40: An example of applying style on dark mode..... 28

Figure 41: Prototype of the interactive code playground..... 29

Figure 42: Terminal of the code playground..... 30

Figure 43: Display of the output 30

Figure 44: Codes that the output is based on 30

Figure 45: Result of inputting a different route 31

Abbreviations

VS Code	Visual Studio Code
SSR	Server-side rendering
ORM	Object-relational mapping

Table of Contents

Abstract II

Acknowledgements III

Table of Figures IV

Abbreviations VI

1 Introduction 1

1.1 Project Background 1

1.1.1 Web development 1

1.1.2 Barriers to learning web development 1

1.1.3 Coding Learning Platform 3

1.2 Project Objectives 4

1.3 Project Contribution 4

1.4 Report Outline 5

2 Project Methodology 6

2.1 Development tools 6

2.2 Project Architecture 7

2.2.1 Frontend implementation 7

2.2.2 Backend implementation 10

3 Project Results 15

3.1 Platform Design 16

3.1.1 Home page 16

3.1.2 Start Learning 18

3.1.3 Account management 22

3.1.4 Editing and creating own courses 24
3.1.5 Responsive Design 27
3.1.6 Dark Mode 28
3.2 Unfulfilled design 29
4 Conclusion and future works 32
4.1 Conclusion 32
4.2 Future works 33
Reference 34

1 Introduction

This section provides a concise overview of the project, including its context and aims. At the end of this section, there is a report outline that details the following report's content.

1.1 Project Background

1.1.1 Web development

As the Internet becomes increasingly prevalent, companies have launched their own web services to meet the needs of their customers, enhancing the quality of their offerings. This has led to a surge in demand for web expertise, elevating the number of web developers and positioning them among the highest earners. In Hong Kong, the average monthly income for entry-level web developers is around HK$25000 (JobsDB, 2024). This figure is nearly on par with the median monthly household income of HK$29,800 reported in 2023 (Census and Statistics Department, 2023).

As the population of novice learners grows, they may frequently face obstacles along their learning journey, which can lead to significant frustration.

1.1.2 Barriers to learning web development

Lack of clear guidance

To become proficient in web development, one must grasp both front-end and back-end knowledge. The front-end is responsible for the client's user interface, allowing users to interact and submit requests, while the backend acts as a server, processing and responding to those requests. Each area demands a unique skill set and offers a variety of frameworks to choose from. For instance, in front-end development, one should start with HTML, CSS, and JavaScript. Following that, selecting a frontend library such as React, Vue, or Svelte can enhance the implementation process. Conversely, in backend development, one should begin by selecting a

framework like Node.js or Spring Boot and then work on integrating the necessary components within that framework.

With the vast array of topics to cover, new learners might feel overwhelmed and uncertain about where to begin. W3Schools (see Figure 1), a well-known web development learning site, seems not to provide clear guidance on starting points and subsequent steps. For example, they list Python following the JavaScript course, which may not align with the practical demands of web development, especially in regions like Hong Kong where Python is not commonly used in this field.

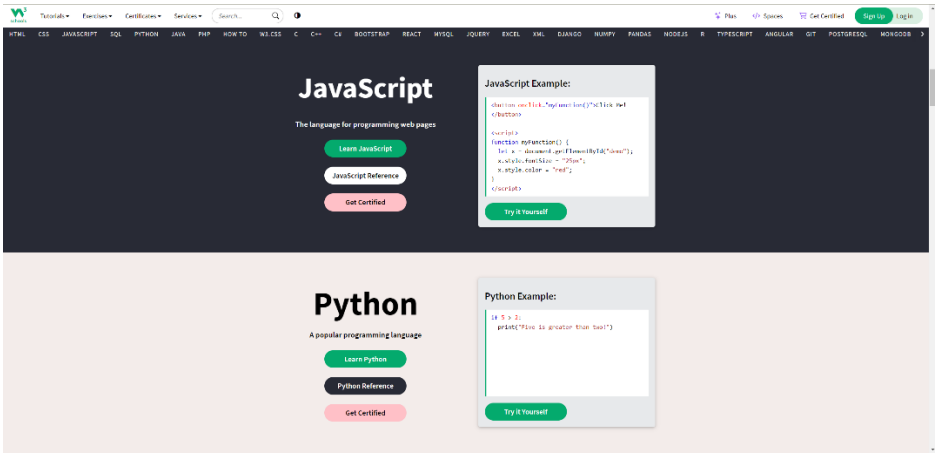

Figure 1: W3Schools Home page

<u>Redundant content</u>

Online learning resources do provide a comprehensive set of knowledge on various topics for beginners. The documentation for specific libraries also offers complete instructions on utilizing each feature.

However, these resources can be overwhelmingly detailed. For junior web developers, much of this information may be superfluous. This can lead to frustration among web developers who spend considerable time learning and memorizing infrequently used details yet struggle to apply them in real-world projects.

Taking the React documentation as an example (see Figure 2), the fundamental concepts of React may be too difficult for a beginner to understand. It is advisable to explore these principles after gaining some practical experience through some hands-on projects.

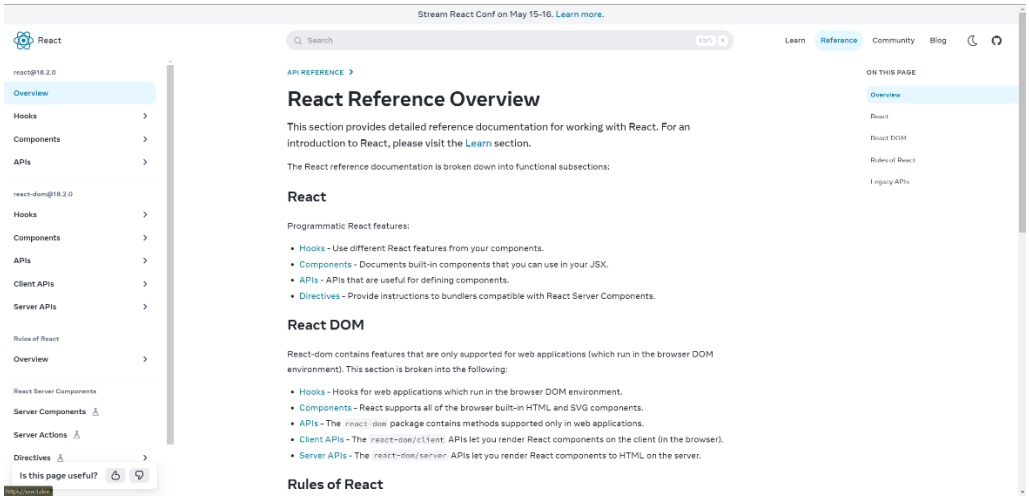

Figure 2: React documentation

1.1.3 Coding Learning Platform

To address the barriers outlined previously, a decision was made to create a coding learning platform tailored for junior web developers.

The platform offers clear instructions, with each vital web development skill transformed into a dedicated course. Learners receive straightforward guidance on which courses to take and the sequence to follow for mastering web development.

The content is designed to be precisely suitable for learners, with some courses developed in-house, ensuring that new learners avoid unnecessary material based on the group's past experiences. Highlighting the distinct features of the provided content: firstly, all materials are aesthetically crafted to enhance reading speed and retention, utilizing **bold effects, varied font sizes, headings, and highlighting effects**. Secondly, an interactive live code editor is integrated within the content, allowing learners to alter code and observe the immediate results. Lastly, the

courses feature audio guidance, with professional voiceovers reading the material, and as the audio plays, the relevant keywords in the content are highlighted.

In addition to the course content, the platform includes a content editor, which eliminates the need for hard-coding course content. This editor enables easy customization of course materials, offering an experience akin to editing a document in Word. Beyond expediting the development of course material by the team, the editor empowers other platform users to craft and publish their own courses. It is hoped that the users can actively participate in writing new course content.

1.2 Project Objectives

The project is designed to provide web development **students** with:

1. Web development courses of different topics in a planned sequence
2. Immersive learning experience featuring aesthetically pleasing course content and assistance tools such as live code editors and audio highlighting
3. Opportunities for reviewing the course content through checkpoint questions

Additionally, it is designed to offer **educators** with:

1. The ability to create and customize their own courses

1.3 Project Contribution

The contributions to the project work, listed in terms of tasks completed, are as follows:

1. Development of the home page
2. Creation of the dashboard page and profile page
3. Implementation of the audio player
4. Designing course materials

5. Designing the pages to be responsive
6. Initial design of the interactive coding playground (yet to be fully realized)

1.4 Report Outline

The subsequent sections, namely section 2 and 3, titled “Project Methodology” and “Project Result”, form the main body of this report.

- In the “Project Methodology” section (section 2), the method of building the entire platform is elaborated. This includes a discussion on the tools used for code editing and version control. In addition, it provides a detailed description of the platform’s architecture, including the libraries used for both the frontend and backend. Special emphasis is placed on the communication between these libraries and the rationale behind their selection.
- The “Project Result” section (section 3) provides a comprehensive guide through the platform. It delves into the practical implementation of the platform’s distinctive features such as the code editor, audio highlighting, and content editor.

The last section, titled “Conclusion and Future Work”, wraps up the project and discusses potential enhancements.

2 Project Methodology

This section explains how the entire platform was constructed. This involves discussing the tools used for code editing and cooperation. Additionally, a detailed overview of the platform's architecture is provided, including the libraries employed for both the frontend and backend

2.1 Development tools

Visual studio code

Visual Studio Code (VS Code) serves as the primary code editing tool for development. It offers extensive support for multiple programming languages and libraries. Notably, its wide range of extensions streamlines the development process, enhancing productivity and ensuring seamless workflows. Among its features are code IntelliSense, which aids in autocompletion, and AI code generation (powered by Copilot), which analyzes context to produce code efficiently. These capabilities collectively contribute to an efficient and error-reduced coding experience.

Git

Git helps synchronize local folder changes with the GitHub repository, keeping track of code modifications and enabling team collaboration. Additionally, the VS Code extension simplifies Git usage through a user-friendly interface.

Excalidraw

Excalidraw functions similarly to a whiteboard that can be written word, sketch graph on it. Its user-friendliness makes it an ideal tool for jotting down the fundamental concepts of various libraries (see Figure 3) and specific platform features. This aids in memory retention during the implementation process and provides a blueprint to adhere to.

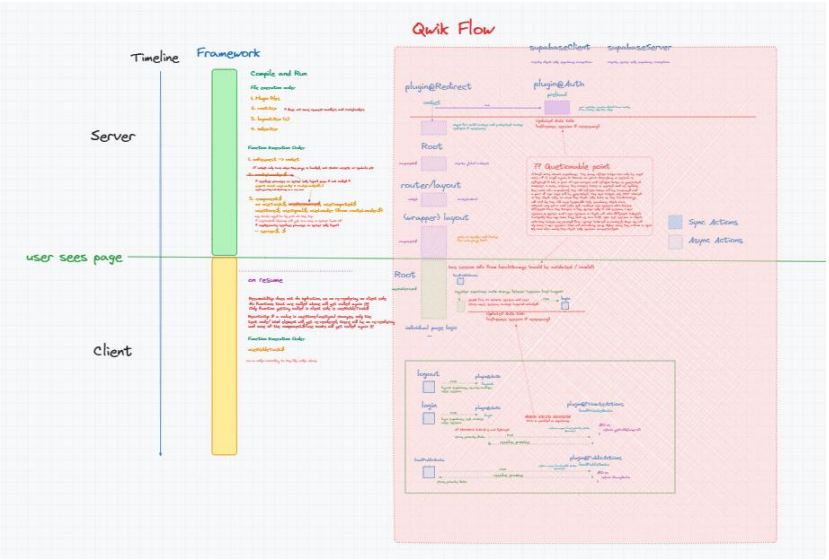

Figure 3: An illustration of the working principle of Qwik drawn using Excalidraw.

2.2 Project Architecture

2.2.1 Frontend implementation

<u>Web framework</u>

Before discussing the chosen web framework, it is essential to understand the concept of server-side rendering (SSR) (see Figure 4). When a user accesses a website, they request the HTML file that represents the webpage from the server. The server, after retrieving all necessary data from the database, makes the HTML file accordingly and sends it to the user for viewing. At this stage, the webpage is not interactive because the JavaScript code has not yet been delivered from the server. Once the user receives executes the code it, the webpage becomes interactive.

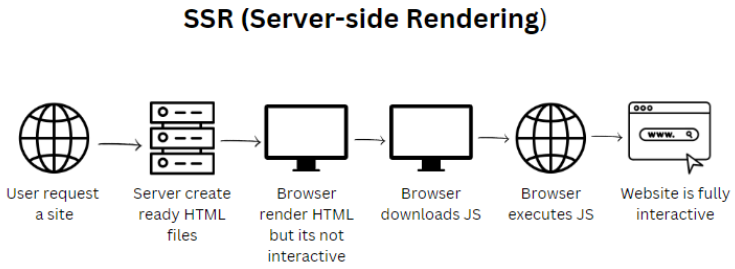

Figure 4: Working principle of SSR

In essence, SSR delivers only the HTML files to the client during the initial webpage load, ensuring that the client can view the page. The rest of the JavaScript code, which enables the webpage's interactivity and takes more time to download, is sent later. This approach of SSR provides a significant speed benefit when loading a page, leading to a substantial improvement in the user experience.

In light of the advantages of SSR, a decision has been made to utilize a frontend framework that incorporates SSR. For this platform, a less conventional framework, **Qwik** is used, as opposed to the more widely recognized NextJS.

Upon analysis, it has been discovered that Qwik outperforms other SSR frameworks by only sending JavaScript code to the client when the corresponding event listener is triggered. For instance, if a button on the webpage is clicked, the "onClick" event listener is activated and the relevant JavaScript is sent to the client for execution. This method reduces the volume of JavaScript code that needs to be loaded on the client side, thereby enabling even quicker webpage loading. Despite the fact that Qwik may lack extensive support from common internet libraries, the decision has been made to experiment with and explore this newly developed framework due to its superior performance.

Styling

TailwindCSS is utilized for the styling of web pages. With TailwindCSS, there is no requirement to create CSS classes in a separate file and then allocate those classes to the components (see Figure 5 and 6).

```
.nav {
  display: flex;
  gap: calc(var(--base) / 4) var(--base);
  align-items: center;
  flex-wrap: wrap;
  opacity: 1;
  transition: opacity 100ms linear;
  visibility: visible;

  > * {
    text-decoration: none;
  }
}
```

Figure 5: Styling using CSS class 1

```
<nav className={classes.nav}>{/* content */}</nav>
```

Figure 6: Styling using CSS class 2

Instead, CSS properties can be directly written into the class attribute of a component to attain the desired outcome (see Figure 7).

```
<p class="text-lg font-medium">
  “Tailwind CSS is the only framework that I've seen scale
  on large teams. It’s easy to customize, adapts to any design,
  and the build size is tiny.”
</p>
```

Figure 7: Styling using TailwindCSS

This approach of styling is similar to using inline styles. However, inline styles come with certain limitations, such as their inability to use media queries for responsive design and to target states like hover or focus.

2.2.2 Backend implementation

Website hosting

Vercel allows the platform to be hosted at no cost and provides servers that respond to user requests for webpages. A significant feature of Vercel is its globally distributed servers, deployed in thousands of locations worldwide (Vercel, n.d.). This implies that when a user requests a resource, the nearest server to the user's location from all the servers around the world will respond, minimizing the latency and hence improving user experience.

It is worth mentioning that Vercel also assists in managing the platform's environment variables, enabling developers to effortlessly manage variables in both the production and development environments during platform construction.

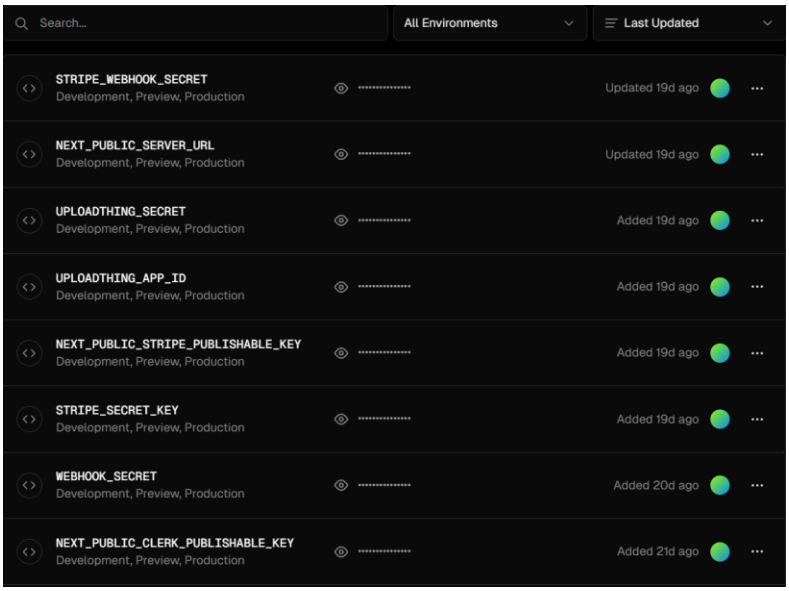

Figure 8: Environmental variables in Vercel

Database

When comparing relational databases such as SQL and PostgreSQL to non-relational databases such as MongoDB, several advantages stand out. One significant advantage is the assurance of full-type safety, as the data structure, including columns, primary keys, and relations, is

predefined. Another advantage is the guarantee of data safety, especially during transactions where modifications to data across multiple tables need to either fully succeed or fail.

In light of the above, the relational PostgreSQL database **Turso** has been selected as a solution for storing user sessions and course content. Turso databases are replicated in over 34 locations worldwide, enabling users to access the closest database with minimal network latency (Turso, n.d.).

Instead of writing raw SQL statement, an Object-Relational Mapping (ORM) library called **Drizzle ORM** is utilized to manage data in the database. This library converts data from database tables into objects in a chosen language, which enables the alteration of data in the database via these objects, thereby simplifying database operations (see Figure 9 and 10).

```sql
SELECT *
FROM countries
LEFT JOIN cities ON cities.countryId = countries.id
WHERE countries.id = 10;
```

Figure 9: SQL SELECT statement

```
await db
  .select()
  .from(countries)
  .leftJoin(cities, eq(cities.countryId, countries.id))
  .where(eq(countries.id, 10))
```

Figure 10: SELECT statement using ORM

User Authentication and Authorization

Lucia Auth is used to authenticate the identity of a user and assign certain authority based to the user's role such as an admin role or a user rule.

When a user enters their login credentials and presses the login button, Lucia Auth on the Vercel server retrieves their relevant data from Turso to see if the credentials correspond with the data. If there is a match, a user session is initiated and managed by Lucia Auth. Based on the user's session information, the login status can be maintained until the user logs out (which is also done by Lucia Auth), and certain permissions can be granted based on the user's role.

Figure 11: Workflow of user authentication

Password hashing

The significance of password hashing cannot be overstated, given that passwords are confidential data that should never be stored in the database without being hashed. If the database were to be accessed by individuals with malicious intent, they could potentially reveal the passwords, putting all user accounts at risk.

Nonetheless, the Vercel servers that host the website lack robust computational power, making themselves unsuitable for computationally demanding tasks such as password hashing. As a result, a dedicated server is required specifically for hashing passwords.

To meet this need, a backend server for password hashing has been implemented using **Elysia**. Elysia was chosen over the more commonly used Express because Elysia is supercharged by Bun runtime (something similar to NodeJS runtime). This allows Elysia to operate at a speed that is 21 times faster than Express (Elysia, n.d.).

Once Elysia is embedded, an additional step will be needed for user authentication (see Figure 12). the Vercel server will send the password to Elysia and obtain the hashed password. Subsequently, Lucia Auth will carry out its password-matching process.

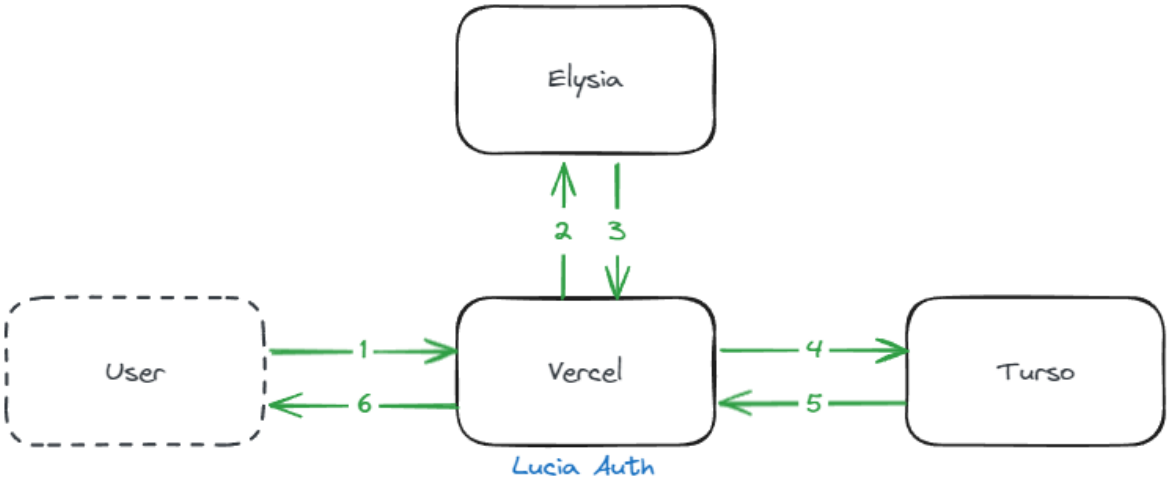

Figure 12: Workflow of user authentication with hashed password

Email Verification

The platform incorporates an email verification process. Upon signing up, new users are required to access their email and click the "Verify Email Address" button (see Figure 13. Only after completing this step can they create their own course on the platform. It is noted that users can still access existing courses without logging in, although their progress will not be saved in such cases.

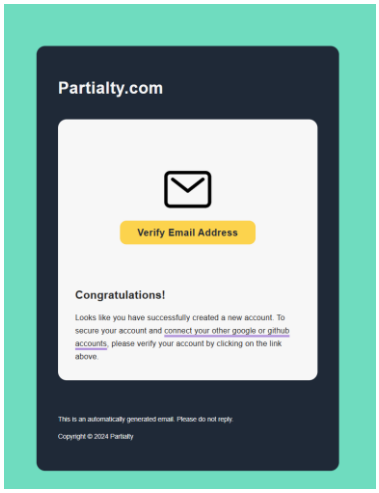

Figure 13: Verification Email

The task of sending emails is handled by the Elysia server. Nevertheless, it does not involve synchronous communication between the Vercel server and Elysia (where Vercel sends a request to Elysia, Elysia sends the email, and then sends a response back to Vercel). Synchronous communication would require waiting for a response from Elysia, tying up resources and putting a strain on the Vercel server.

To address this, asynchronous communication is employed via **QStash** for email services. Rather than sending a request directly to Elysia, the Vercel server sends a message to Qstash's message queue. Elysia, which is subscribed to this queue, will carry out its email service whenever it detects a message in the queue.

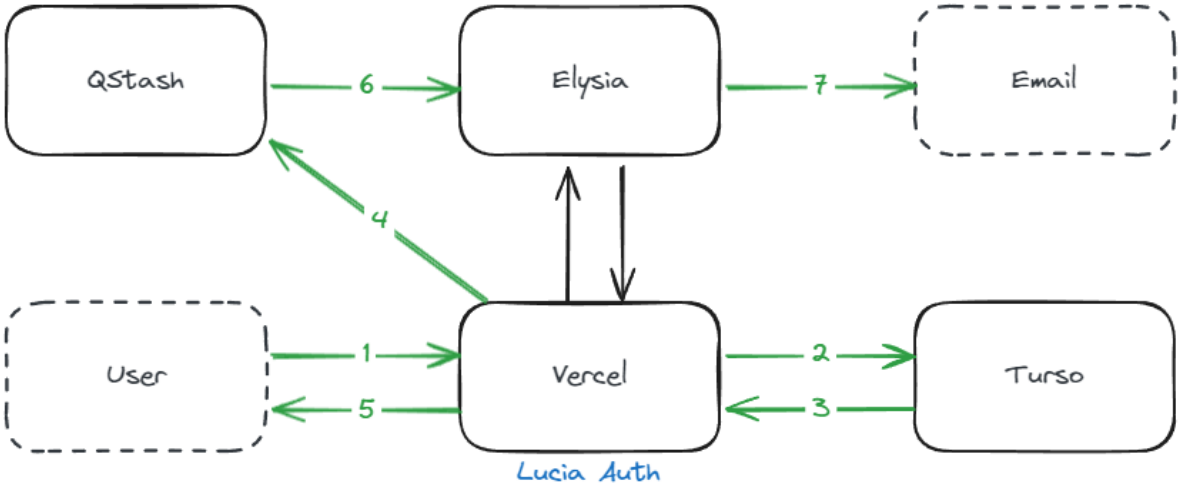

Figure 14: Workflow of sending verification email

Asynchronous communication implies that once Vercel sends the message to the queue, the email-sending process on the Vercel server is complete and the associated resources can be freed up. The remaining tasks are handled by Elysia. This approach also reduces the dependency between the two servers.

3 Project Results

This section serves as a thorough walkthrough of the platform. It explores the application of the unique features, including the code editor, audio highlighting, and content editor.

3.1 Platform Design

3.1.1 Home page

Introduction section

This is the first section that visitors encounter when they land on the website (see Figure 15). On the left side, there is an animated code editor that simulates an editing effect. The right side clearly displays the intended function of the platform.

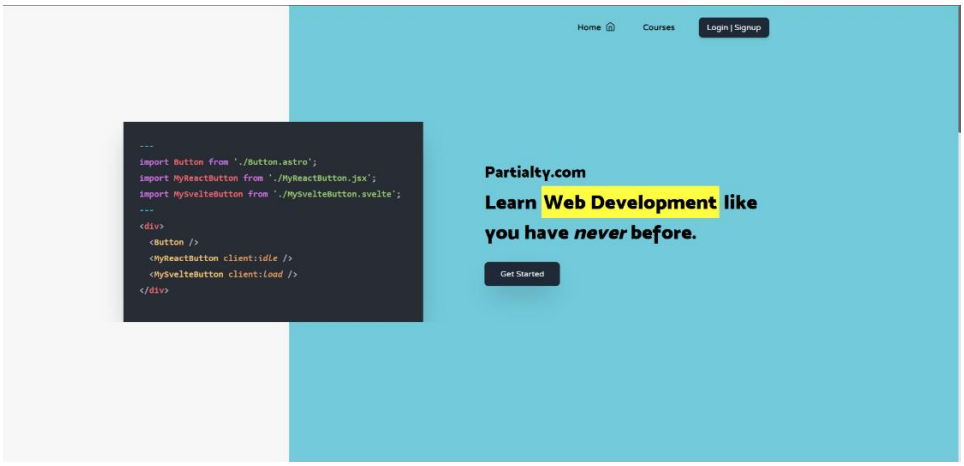

Figure 15: First section of the home page

Tutorial section

This is the second section that visitors will see as they scroll down the webpage (see Figure 16). This animated section alternates to display various tutorials available on the platform, each accompanied by a concise description of what the tutorial covers.

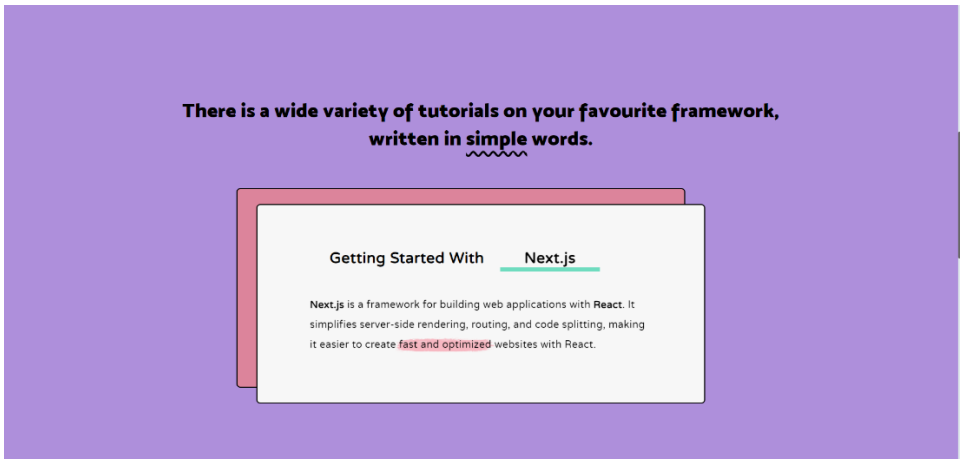

Figure 16: Tutorial section

Audio-guided course section

The final section showcases a unique feature of the platform, which is the inclusion of audio-guided courses. As the audio soundtrack progresses and keywords are spoken, the corresponding keywords on the screen are highlighted (see Figure 17 and 18).

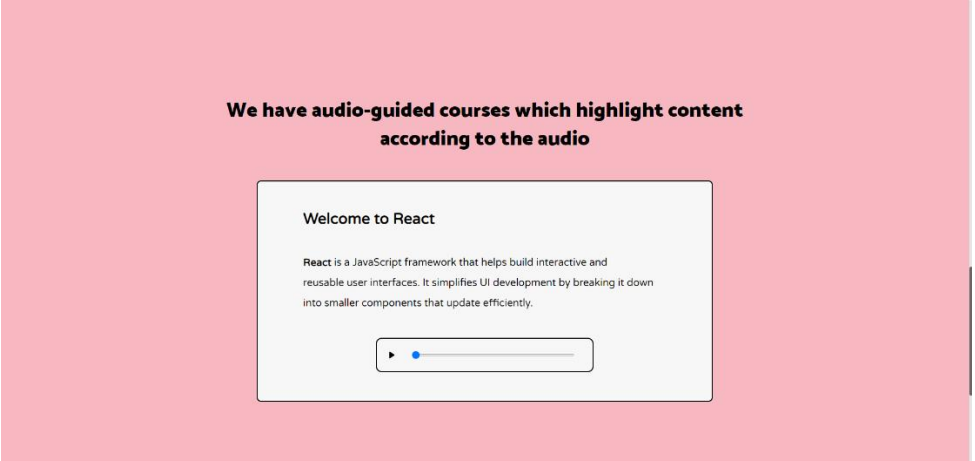

Figure 17: Audio-guided course section

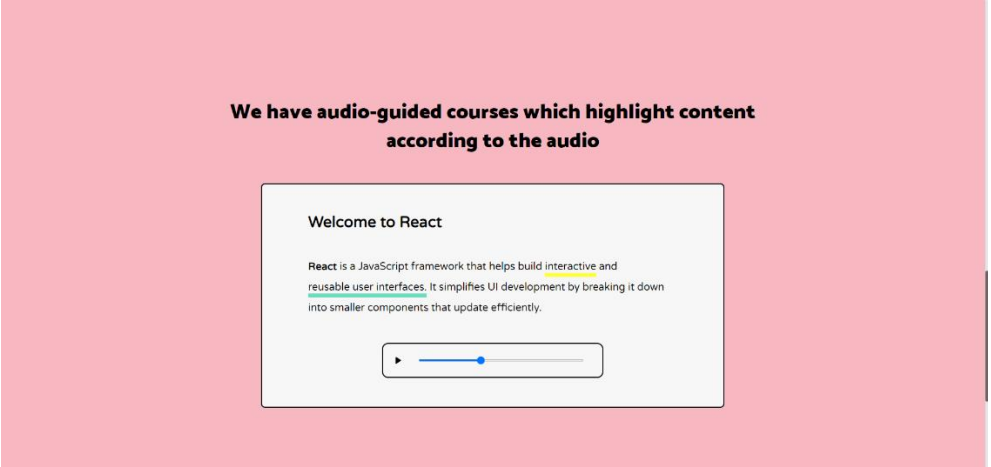

Figure 18: Audio-guided course section (highlighted)

3.1.2 Start Learning

Courses page

Without requiring a login, once the "Courses" button at the top of the home page is clicked, users will be navigated to the courses page displaying all available courses (see Figure 19). As of the deadline for submission, due to constraints on time, only five courses have been held. These courses cover basic web development skills, including HTML, CSS, JavaScript, and NodeJS.

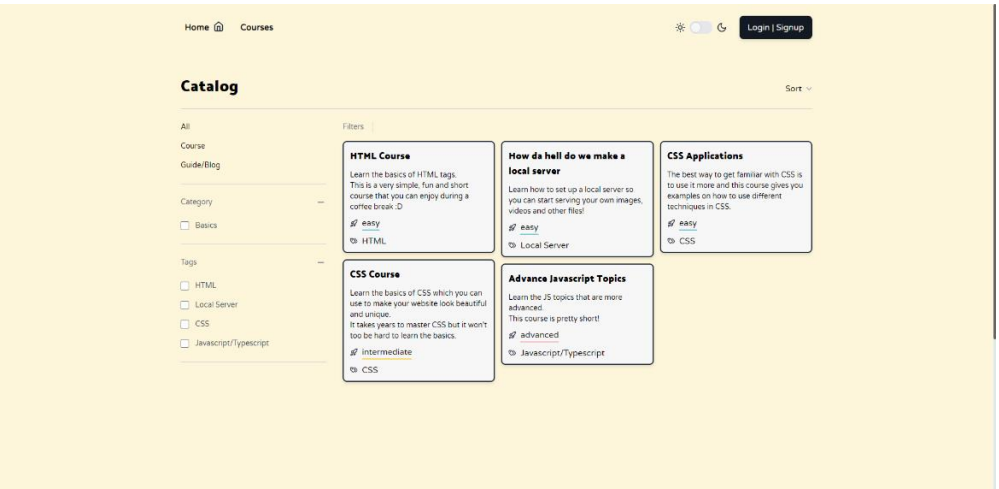

Figure 19: Courses page

Course details page

When a course is selected and clicked on, users are directed to the course details page (see Figure 20). This page provides comprehensive information about the course, including its content, difficulty level, number of chapters, the author of the course, and the language in which the course is taught.

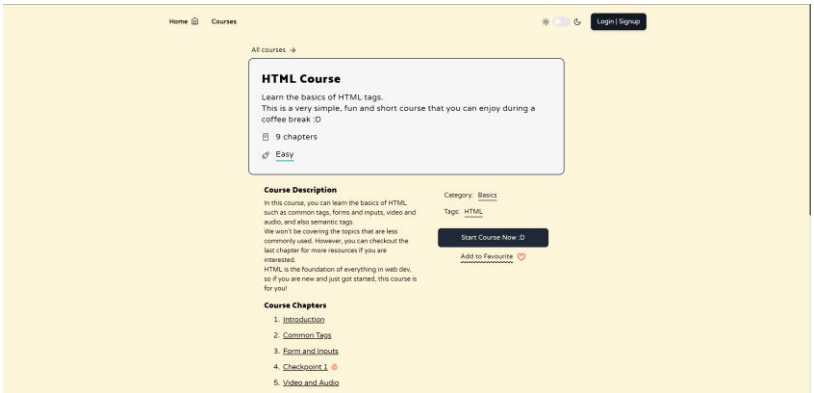

Figure 20: HTML course details page

Course content

Upon clicking the "Start Course" button on the course details page, users are directed to the course content page (see Figure 21). The left side of the page presents **a list of chapters** that users can select. It is expected that users will progress through these chapters sequentially. The right side of the page displays **the content of the chosen chapter.**

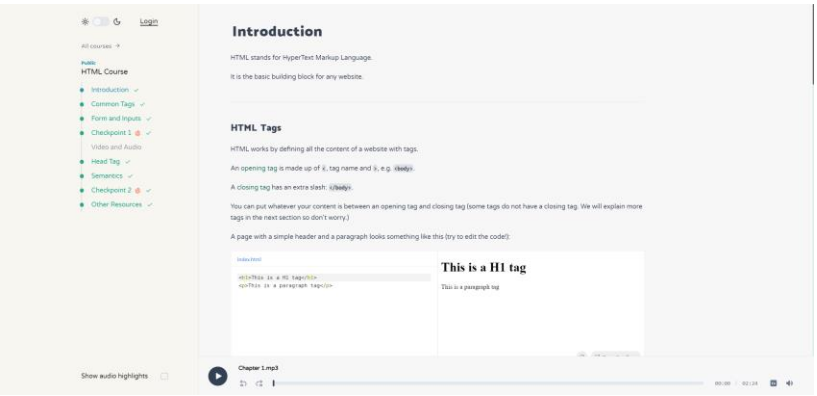

Figure 21: HTML course chapter 1

Interactive code editors are included within each course to demonstrate the output of the code (see Figure 22). Users can alter the code on the right side to observe the effect of the modified code on the left side.

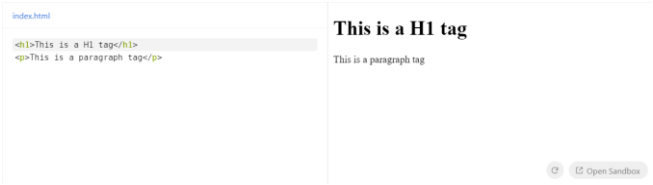

Figure 22: Live code editor within a course

An audio player is located at the bottom of the course content section (see Figure 23). If the play button is clicked, users can listen to a narration of the chapter content. As the audio progresses, the keywords that are spoken will be highlighted on the screen in sync with the audio (see Figure 24). Users also have the option to adjust the speed and volume of the audio for an optimized listening experience.

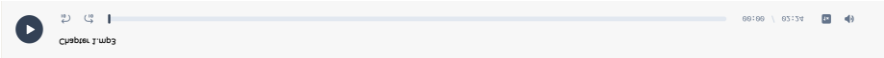

Figure 23: Audio player at the bottom of the course content section

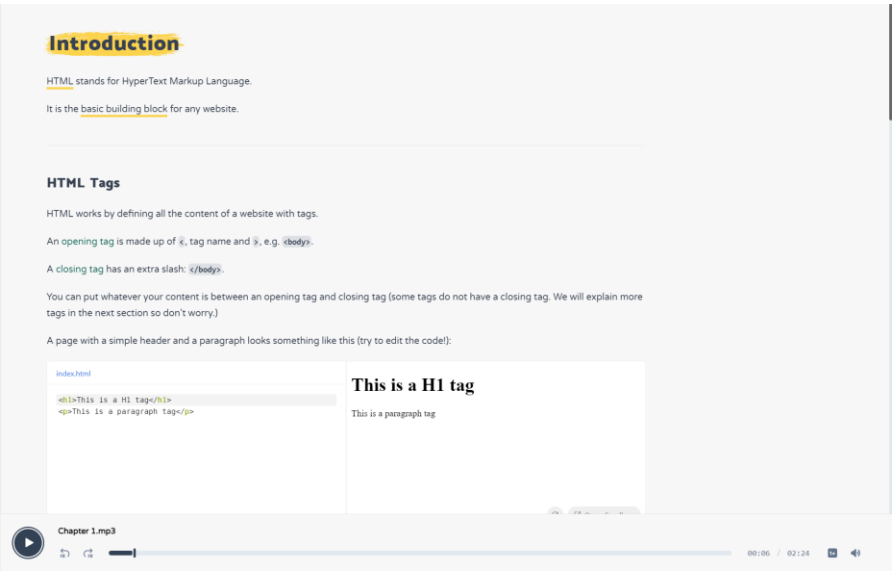

Figure 24: Audio-highlighting effect

After progressing through several chapters, users will encounter **a checkpoint chapter**. This provides an opportunity for users to review what they have learned before advancing to a higher level. The checkpoint includes two types of questions: fill-in-the-blank (see Figure 25) and multiple-choice (see Figure 26).

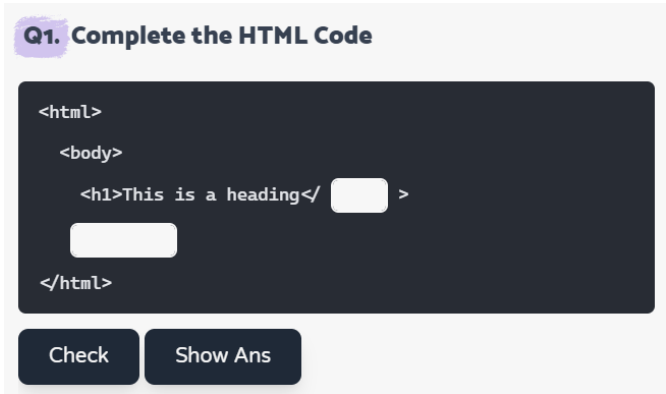

Figure 25: Fill-in-the-blank question in a checkpoint chapter

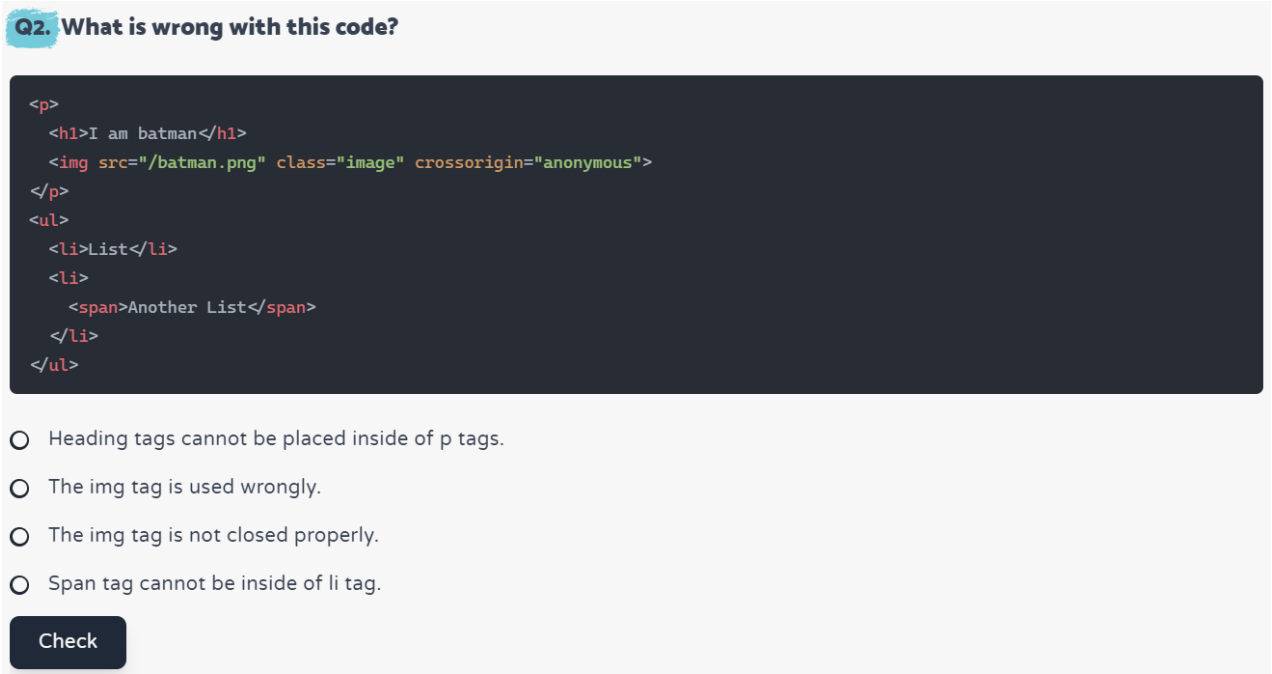

Figure 26: Multiple-choice question in a checkpoint chapter

3.1.3 Account management

Login and Sign up

To record learning progress and access the profile page, users must first log in to their account. A "Login" button will always be visible at the top of each screen for users who are not logged in. By clicking on it, they will be directed to the login page.

Users have the option to log in to their account using their own email or through OAuth of Google and Github. If they do not have an account yet, they can sign up. They must follow a few steps to complete the sign-up process (see in Figure 27).

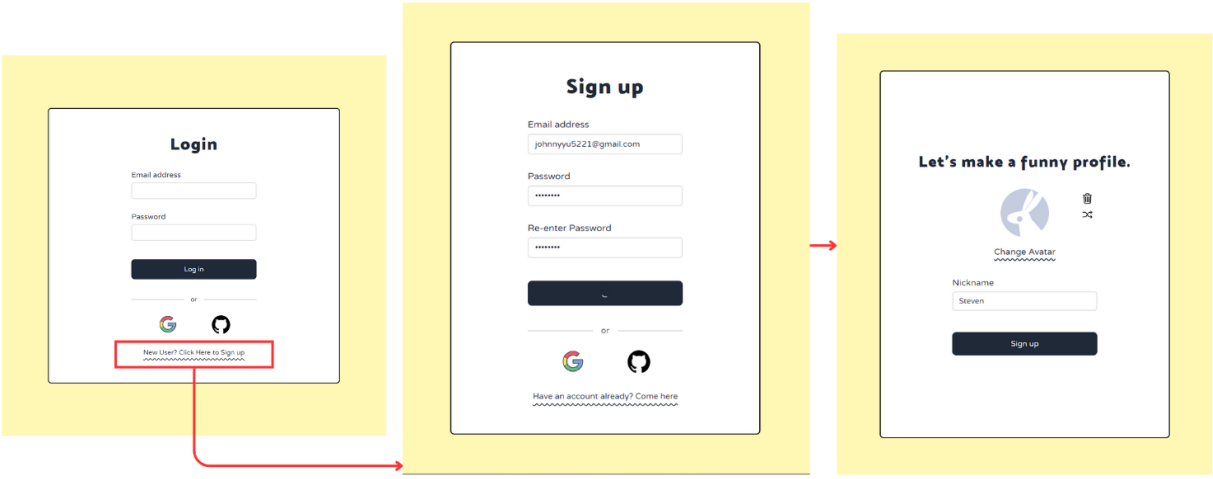

Figure 27: Steps to sign up for an account

Dashboard page

Once logged into their account, users are directed to the dashboard page (see Figure 28). The dashboard displays all the courses that the user has enrolled in and tracks their progress until completion. If users choose to expand a course, they can view detailed completion status for each chapter.

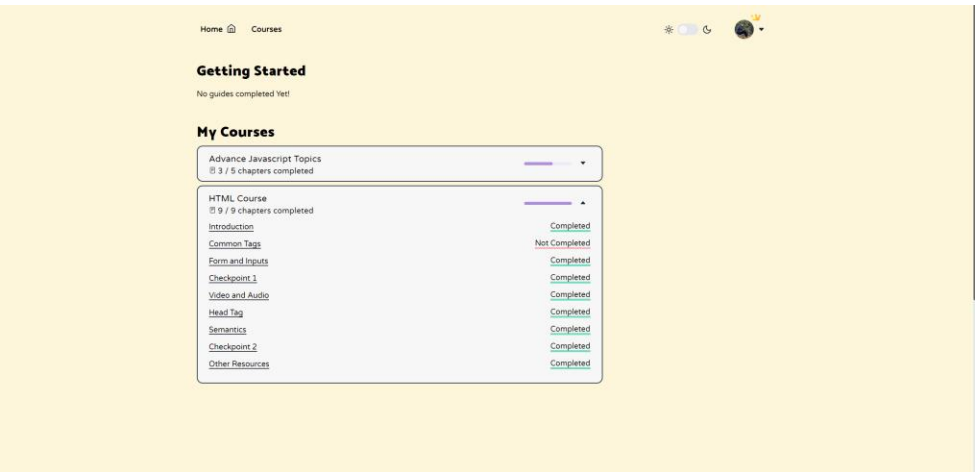

Figure 28: Dashboard page

Profile page

When the icon at the top right corner of the screen is hovered over after logging in, a dropdown menu is displayed (see Figure 29).

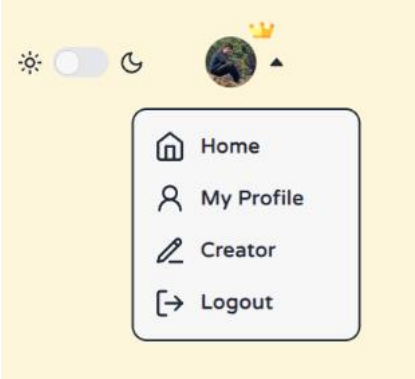

Figure 29: Dropdown menu when hovering over the icon

Among the various options, if users select “My Profile”, they will be directed to the Profile page (see Figure 30). On this page, users have the ability to modify their nickname, password, and avatar.

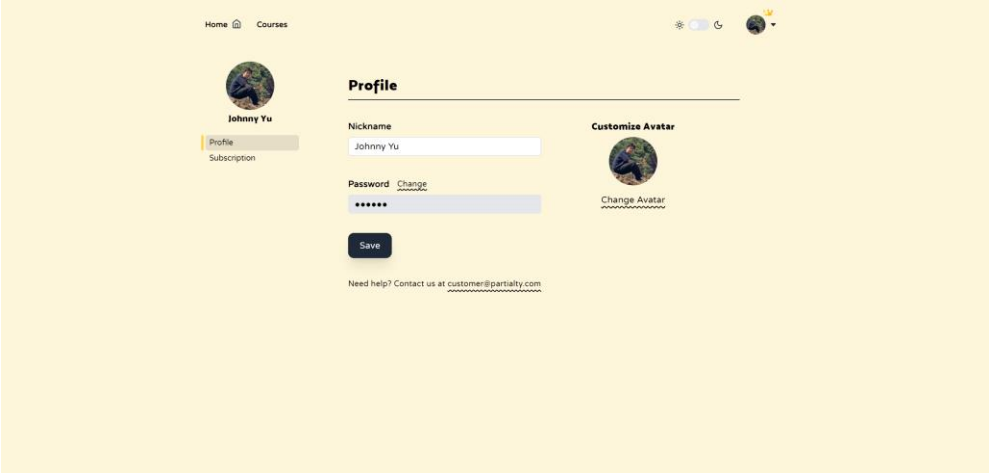

Figure 30: Profile page

3.1.4 Editing and creating own courses

Editing course

When the icon is hovered over and "Creator" is selected from the dropdown menu, users are directed to a page that lists all the courses they have the authority to modify. By clicking on a course, it expands to reveal more details (see Figure 31).

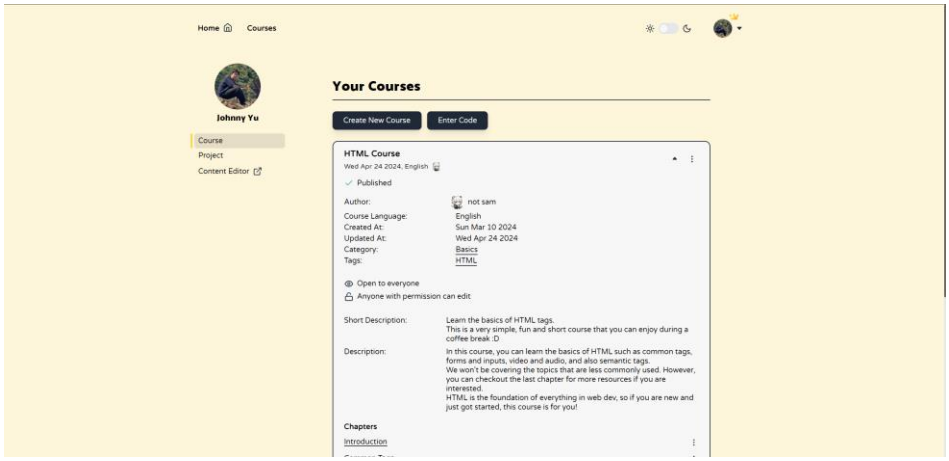

Figure 31: Creator page with expanded course

As users scroll down, they can see a list of chapters. By clicking on the three dots next to a chapter name and selecting “Edit Content” (see Figure 32), users are taken to the chapter’s editing page (see Figure 33).

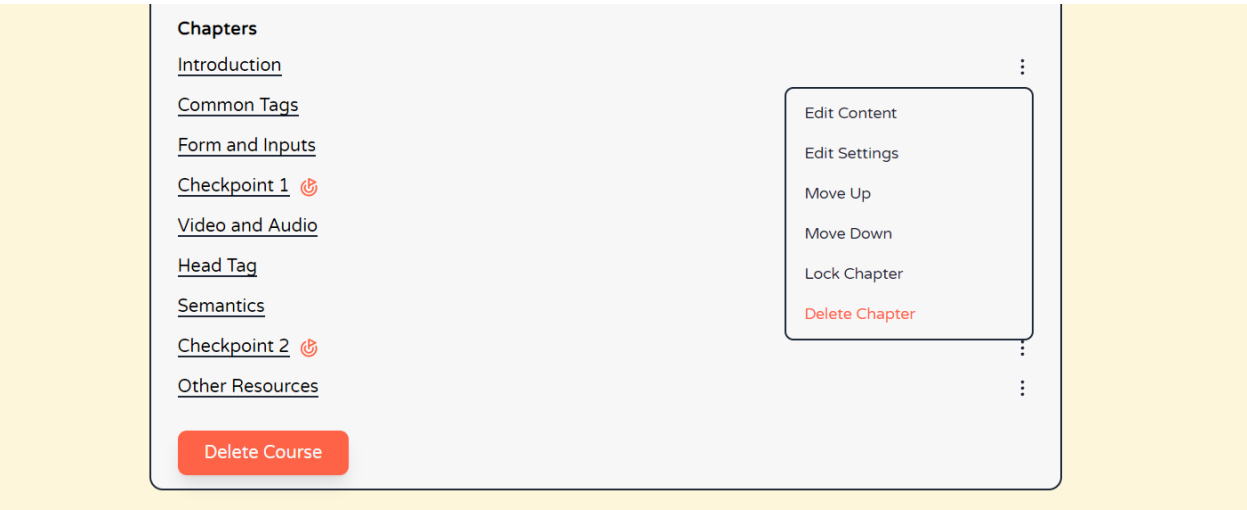

Figure 32: Requesting for the editing page of the chapter

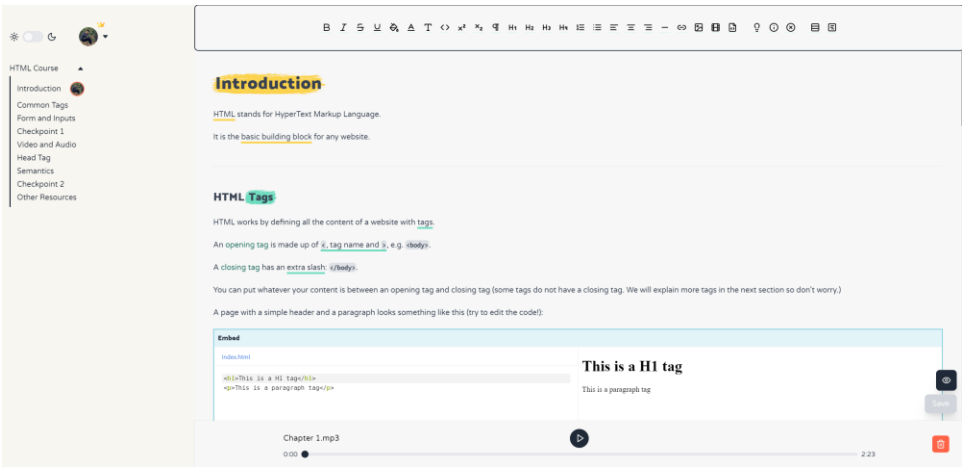

Figure 33: Editing page of HTML course chapter 1

Users now have the ability to modify the chapter content to their liking, similar to editing a Word document. A toolbar at the top of the page enables the creation of a customized course with various styles and components (see Figure 34).

Figure 34: Toolbar on the chapter editing page

The following is a list of all available styles and components that can be inserted:

1. Effects such as bold, italic, and strikethrough
2. Three distinct highlighting effects
3. Adjustments to font size, typeface, and spacing
4. Code snippets in various programming languages
5. Superscript and subscript text
6. Paraphrasing and headings ranging from level 1 to 4
7. Various types of lists
8. Text alignment options
9. Line breaks
10. Image insertion
11. Embedded code editor

Creating course

On the same page where chapter editing actions can be performed, users can click on "Create New Course" to initiate a series of steps to complete the course creation process (see Figure 35).

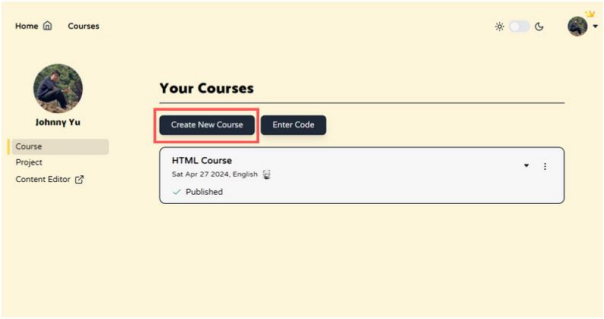

Figure 35: Button to create a new course

Users will then be prompted to provide a course name, description, category, tags, and language (see Figure 36).

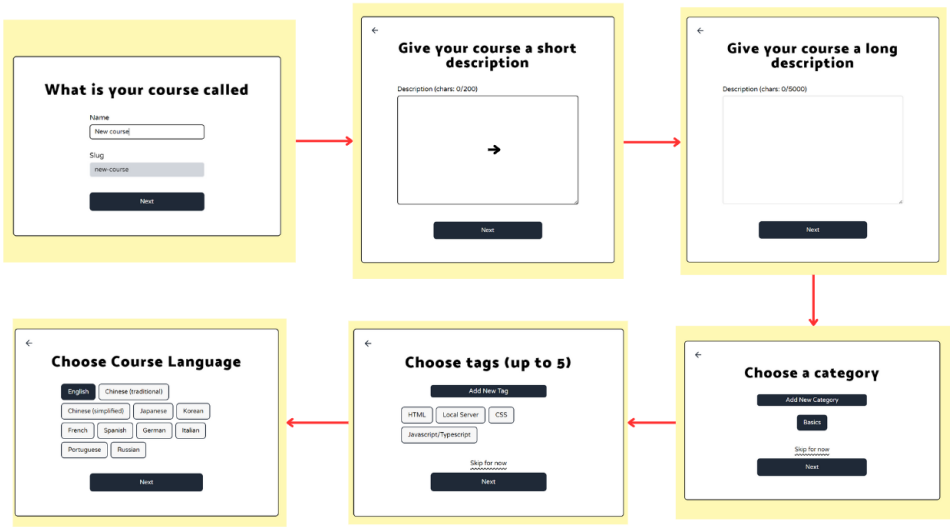

Figure 36: Steps to create a course

3.1.5 Responsive Design

Every page is constructed following the principles of responsive design. This approach allows the viewport to adapt based on the device being used by the users. As a result, even on devices with smaller screens like smartphones, users can easily engage with the learning material (see Figure 37).

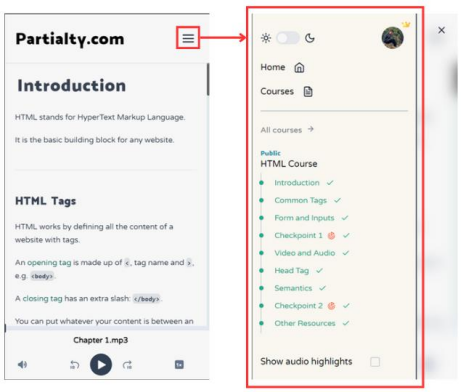

Figure 37: Responsive design of the course content page

For the implementation of responsive design (see Figure 38), CSS styling must be applied to every component on the pages such as containers, sections, paragraphs, texts, etc. Styling aspects such as width, height, and layout also need to be specified for each screen size to ensure optimal viewing on different devices.

```
<!-- Width of 16 by default, 32 on medium screens, and 48 on large screens -->
<img class="w-16 md:w-32 lg:w-48" src="...">
```

Figure 38: An example of implementation of responsive design

3.1.6 Dark Mode

A toggle button is always present on the screen, enabling users to switch between light and dark modes according to their preference.

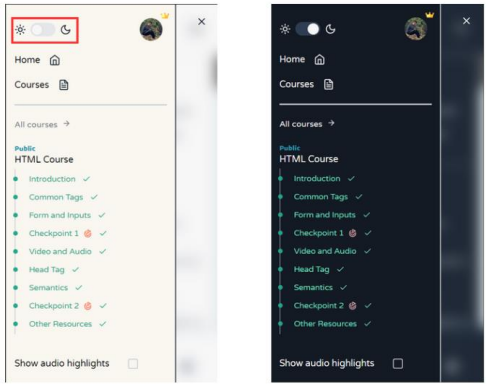

Figure 39: Dark mode of the course centent page

Implementing the dark mode required extra effort to define the color style for each component in both light and dark modes.

```
<div class="bg-white dark:bg-black">
  <!-- ... -->
</div>
```

Figure 40: An example of applying style on dark mode

3.2 Unfulfilled design

The initial design was for an interactive coding playground to be integrated within a platform. This playground could be used for testing code and even completing tasks set by the courses. A prototype of this playground has been developed (, featuring a folder tree, an editing section, a top bar for file management, a terminal for managing dependencies and files, and a screen for displaying code output, similar to what can be found in VS Code.

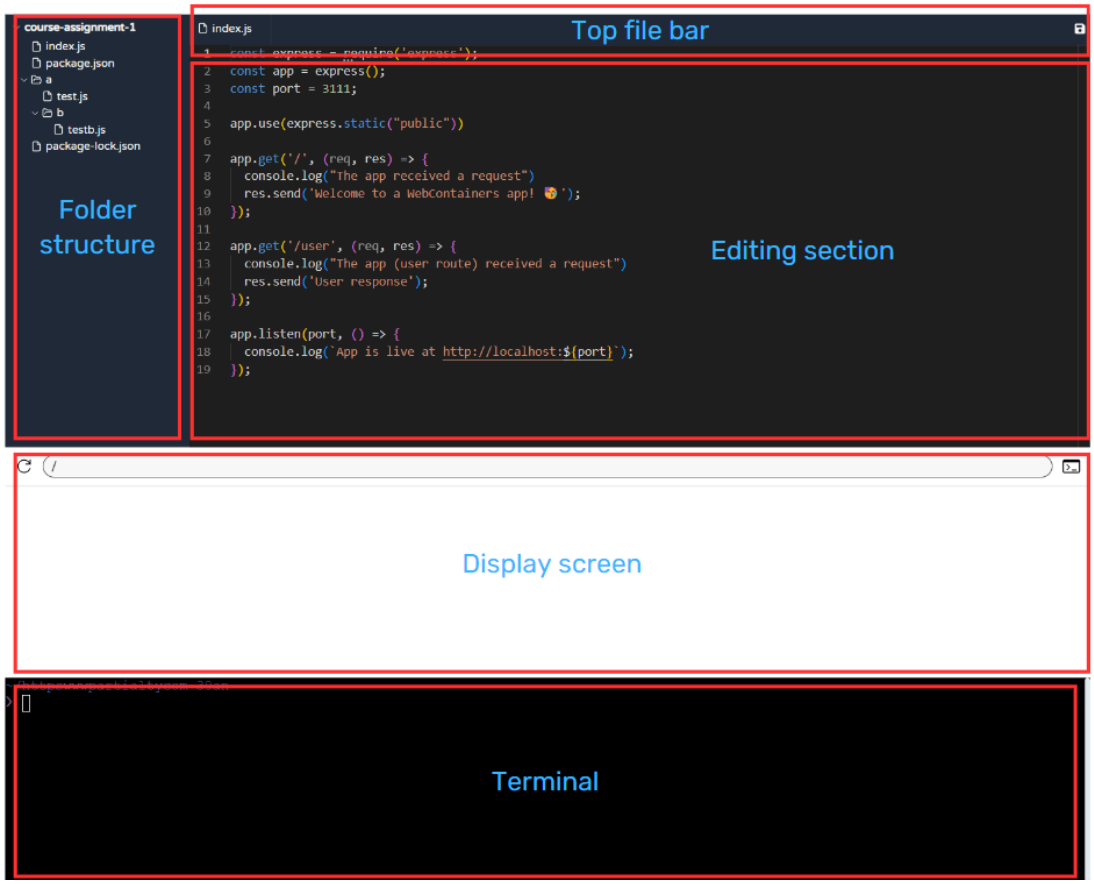

Figure 41: Prototype of the interactive code playground

However, there are still aspects that need to be completed, but time is running short. Considerations need to be made for embedding the code editor into the course, executing code in different programming languages (currently, only JavaScript code can be executed), and

managing user files. In short, there are many factors to consider for completion, but time is limited.

Despite this, a few steps on how to use the prototype will be demonstrated to provide a more tangible example of what the coding playground can achieve:

1. Enter the command `npm start` in the terminal (see Figure 42) and press the `Enter` key.

Figure 42: Terminal of the code playground

2. The screen will display an output (see Figure 43) that is based on the code (see Figure 44).

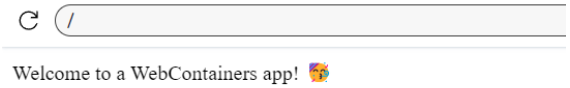

Figure 43: Display of the output

```
app.get('/', (req, res) => {
  console.log("The app received a request")
  res.send('Welcome to a WebContainers app! ');
});
```

Figure 44: Codes that the output is based on

3. Input a different route in the search bar to request a different type of response (see Figure 45).

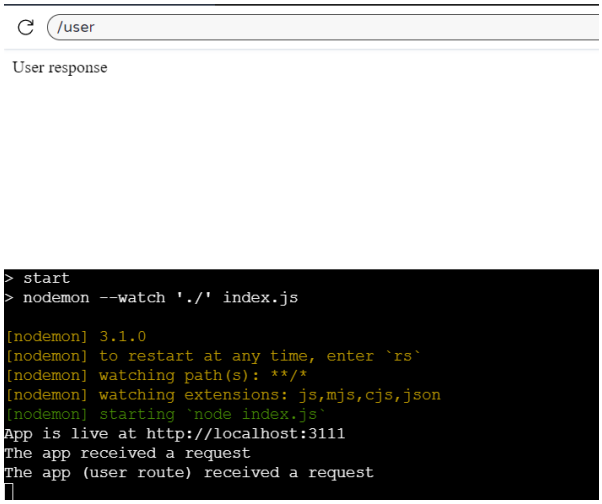

Figure 45: Result of inputting a different route

In simpler terms, the interactive coding playground is designed to function similarly to how users would operate VS Code. The key difference is that instead of executing code on a user's personal computer using VS Code, the playground executes the code on a remote server. This allows for smooth coding experience regardless of the computing power of the user's computer.

4 Conclusion and future works

4.1 Conclusion

To summarize, considering the current challenges of online web development learning, a unique coding learning platform has been developed. This platform aims to offer students an enhanced learning experience in web development compared to other platforms, with distinctive features such as live code editing and audio highlighting.

In addition to learning from the courses on the platform, learners are also given the opportunity to create their own courses. A suite of editing tools, similar to those found in a word document, is provided to enable the creation of aesthetically pleasing, customized course content.

The selection of libraries, frameworks, and tool techniques was based on their specific advantages and functionalities. For development tools, VS Code is utilized for code editing due to its support for extensions for various languages. Git is employed for file version control and team collaboration. Excalibraw is used to provide a comprehensive view of the entire project architecture, ensuring adherence to the architecture throughout the project.

For the front end, the Qwik framework is chosen mainly because it supports server-side rendering, which enables faster web page loading. Tailwind CSS is used for web page styling.

On the back end, Vercel is used as the server for hosting because it can run servers in various locations worldwide, reducing latency. The Turso database is used as it also supports low latency manipulation by hosting the database globally. Drizzle ORM is used to manage the database through an object for better data control. Lucia Auth is used for user authentication and authorization. A dedicated server, Elysia, is used for password hashing due to Vercel's low computing power. Elysia, in conjunction with a message queue known as QStash, is responsible for sending verification emails to the user.

Currently, the platform offers five courses covering HTML, CSS, JavaScript, and NodeJS server. Each course is beautifully designed and incorporates the most essential skills for mastery, without any unnecessary content.

4.2 Future works

The future plan is to continually update the platform with more course content. This includes courses on the latest trends in web development, such as React and Astro frontend web libraries, as displayed on the homepage. These are essential skills for mastering web development. Plans are also in place to add backend server courses, such as Spring Boot and .NET Core, in addition to the existing NodeJS course.

Another plan is to introduce a new course category called ‘Project’. In these courses, a combination of skills, libraries, and frameworks will be taught to create practical projects that require both frontend and backend knowledge. Examples of these projects include a profile page, a commerce page, a social media page, and more.

Lastly, there is a need to complete the interactive code playground. At present, the playground can only execute JavaScript code. Two potential solutions are being considered. The first is to set up a server using Amazon EC2 to execute code in other languages and return the results to the users. The second option is to use Amazon Lambda service to execute the code and return the results. Both options likely operate on similar mechanisms, but further research is needed.

Reference

Census and Statistics Department. (2023, May 30). *Quarterly Report on General Household Survey*. https://www.censtatd.gov.hk/en/wbr.html?ecode=B10500012023QQ01&scode=200

Elysia. (n.d.). *Elysia - Ergonomic Framework for human*. https://elysiajs.com/

JobsDB. (2024, April). *Web developer salary in HK*. https://hk.jobsdb.com/career-advice/role/web-developer/salary

Turso. (n.d.). *Turso SQLite for production*. https://turso.tech/

Vercel. (n.d.). *Vercel: Build and deploy the best web experiences with the frontend cloud*. https://vercel.com/home