COMP4801 Final Year Project 2023-24
Detailed Project Plan

Topic: Coding Learning Platform

Yu Jian Hui (3035790719)
Yung Chin Pang (3035778682)

Project Background

While many existing online learning platforms provide coding materials, they often lack guidance in building actual projects or applications. Users are left to navigate lengthy documentation or follow pre-designed tutorials on platforms like YouTube, which can be tedious and limit their ability to think independently.

To overcome these challenges, we plan to develop a server-side-rendered coding learning platform with comprehensive back-end and front-end functionalities. Our platform will provide guides, learning resources, and step-by-step project walkthroughs for different coding languages. What sets us apart is the incorporation of interactive coding playgrounds within the project instructions. These playgrounds will display the user's input code and the corresponding results side-by-side, allowing users to gain hands-on experience in building practical projects that can be hosted on the modern internet.

By offering all the essential details within the platform, users will no longer need to search for required components independently. Additionally, the inclusion of interactive coding playgrounds encourages users to think critically and understand the outputs of their code. This addresses the challenges faced by junior developers and promotes an in-depth understanding of the concepts.

Furthermore, our platform will include other features commonly found in other e-learning platforms, such as quizzes, comment sections, and progress tracking per user account. These features will ensure a comprehensive learning experience and enable users to track their progress throughout their learning.

Project Objective

The primary objective of this project is to build a web app that allows junior web developers to construct their own websites from scratch. These websites will encompass the key components typically found in standard web pages. The web app will guide users through interactive tutorials, which involve the following steps:

1. Provide essential and fundamental knowledge about various web-building techniques.
2. Instruct users to independently modify the code and instantly observe the corresponding changes in the webpage.
3. Evaluate the user's inputted answers before allowing them to progress to the next step.

The project is also aimed at providing users with diverse learning opportunities and an opportunity to review the topics they have mastered through multiple-choice questions, instructional videos, quizzes,
code challenges, and more.

Project Methodology

Development Tools

Our primary development tool of choice for editing is Visual Studio Code. Its extensive selection of supported extensions greatly improves the development process, allowing for a smooth workflow.

UI Framework and Libraries

We have decided to adopt a less common framework, Qwik, instead of the more popular Next.js for our platform this time. Through our analysis, we have found that Qwik offers better performance results by sending JavaScript code to the client only when the corresponding eventListener is triggered. This approach reduces the amount of code that needs to be loaded on the client side, hence allowing for faster loading of web pages. Although Qwik may not have extensive support from common libraries on the internet, we are willing to give it a try and explore this newly established framework because of its performance superiority.

Another important point about Qwik is that it removes the boundaries of front-end and back-end. Traditionally, a standard website will have a clearly separated front-end and back-end written in languages of choice, and an api server which is separated from the website that handles requests. WIth Qwik (and some other frameworks), we can write both front and back-end codes in the same file, essentially eliminating the need to write separate codes that handle the traffic between front and back-end. It also means that the whole website will be written in typescript which can significantly reduce the time spent on debugging compared to traditional JavaScript by providing type safety.

For styling our web pages, we have chosen to use Tailwind CSS, which is widely considered one of the most popular CSS frameworks available. With Tailwind CSS, we no longer need to define classes in a separate CSS file and assign them to the components. Instead, we can directly write CSS properties in the class attribute of a component to achieve the desired effect. This approach is similar to using inline styles. However, inline styles have limitations, such as the inability to use media queries for responsive design and the inability to target states like

hover or focus. Tailwind CSS solves these limitations with its state variants, making it easy to style different states using utility classes. While some individuals may initially perceive this approach as less organized compared to the traditional approach of defining classes, the web development community widely recognizes the benefits and advantages of this approach.

Back-end Structure

As with any websites, we need a server to host them in order for users to be able to access it and the design of the server structure is directly related to the needs of our app. Also, as an additional design preference, we will prefer serverless solutions over needing to configure our own server to reduce the initial coding cost.

For starters, we need a basic server to host our html, css and other static files for the website. We will do some server-side rendering (SSR) to render different pages for authenticated and unauthenticated users so we need a server capable of doing some computations and handling requests. We have used vercel for deploying the website since it runs on the “edge” which means the same code is deployed to vercel’s servers all around the world and only the closest server to the user will respond with the files, lowering the latency. We have also proxied the website through cloudflare’s servers so caching can be done more effectively

Secondly, we need a database to store all our data. One distinct advantage of relational databases such as SQL over non-relational databases such as MongoDB is that we have full type safety since every column and primary key and relations are predefined and we have full knowledge about the structure of the data we can retrieve. We have opted for Supabase for the serverless postgres solution since it acts as a Solution as a Service (SAAS) by including not only the database, but also the ability to perform authentication, storing static content, running edge functions and many more. However, we are actively looking for alternatives since the free tier only includes 500MB of data stored in the database and will likely not meet the demand

Thirdly, we would need a caching mechanism to store user’s session info for fast retrieval due to high latency incurred by accessing the database every time. We have used the Upstash solution of redis to store the users’ session data. Redis is an ultrafast KV-storage which is a database that stores data in the ram instead of in hard disks like traditional databases. Upstash provides an attractive free-tier with 10k requests per hour but the latency is a bit inconsistent after several testings

Fourthly, since we allow users to upload their own codes and files, we need a non-volatile and cheap storage solution with a generous amount of free-tier usage, preferably charged based on number of requests, but not bandwidth used since some files such as videos might have a large file size. To store static files, we went for cloudflare R2 which is a clear winner over AWS S3 due to its simplicity of use and an enormous free-tier of a million requests per month.

Last but not least, we are looking into the possibilities of hosting our own code-evaluation servers. As mentioned above, we cannot run python or java apps on the client side, and

providing intellisense would require a language server. Therefore, in the near future we might look into aws services such as renting EC2 instances and setting up load balancers to programmatically spin-up MIcroVM instances to run docker containers which can then run codes of any kind.

Code Playground

At this point, it should come as no surprise that we need a code editor library that can well-integrate with the features that we will need, namely code highlighting, intellisense, auto-complete and linting. Our initial take is to use codemirror due to its lightweight package size and being widely popular. Yet, the implementation is notoriously difficult and time-consuming since it has almost no built-in features except highlighting a few more others. Hence, we opt for monaco instead, which is the exact library that VSCode is built upon. It provides a clean, easy-to-use and yet feature-full editor interface out of the box. It also provides typescript intellisense and auto-complete natively through web-workers which is crucial. However, we will need dedicated language servers for other languages as mentioned above.

To actually run the codes, webContainers by Stackblitz is used for NodeJS apps. It is an API library that exposes and allows connections to be established to Stackblitz servers via web sockets where the code will be run and the results sent back. If the NodeJS app starts a server and opens a port, the API will return an URL where the user can navigate to and see the resulting page. The ease of use and setup comes at a tradeoff that several request headers need to be configured in order to use the API, and the implication is that embedding external resources, such as images from another site, might not work if the response does not have the same headers.

Learning through the platform

Answer Evaluation

To evaluate the correctness of input codes, we utilize an AST (Abstract Syntax Tree) parser specific to the corresponding programming language. The parser takes the input code and converts it into a tree structure. By analyzing the structure of the tree and checking for the presence of all the required components of the desired answer, we can determine if the user has successfully completed the task and assess their understanding of the relevant concepts.

Project Schedule and Milestones

Date	Milestones
September 2022	Assess the feasibility of the planned features

October 2022	UI design (wireframing & prototyping)
November 2022	Implement the pages designed
December 2022	Write tutorials content
January 2023	Write tutorials content
February 2023	Add in practical projects guides
March 2023	Set up dedicated servers to run codes
April 2023	Implement other functionalities