# Web Application Final Report: Empowering Users to Learn React.js

Anthony Gudiel, Jashan Gill, Clement Lau, Duc viet Nguyen

Group 16 Moons

CMPT 276 D300 - Fall 2023

GitHub Repository:
https://github.com/anthony-gudiel/reactpedia
Website:
https://reactpedia.vercel.app

# Table of Contents

# Project Overview

On our interactive learning platform, we offer a comprehensive and accessible approach to teaching React.js, aimed at beginners and amateur developers. Our integration of OpenAI-powered tools is the central to the platform's design, which provides a personalised and engaging learning experience. This is complemented by a hands-on coding environment within the browser, allowing learners to apply React.js concepts in real-time. An extensive library of video tutorials, created by React.js experts, covers a wide range of topics, enhancing the learning experience visually and audibly. The platform's unique features include integration with YouTube for additional educational content and the OpenAI API for advanced insights and problem-solving strategies. Additionally, the embedded in-browser development tools facilitate a seamless and practical coding experience, which is particularly beneficial for beginners. This platform represents not just a learning tool but a comprehensive gateway into the world of React.js, equipped with the resources and support to ensure a thorough and enjoyable learning journey.

# SDLC Model & What Worked

In our original project proposal, we stated that we intended to use something similar to a V-Shaped software development life cycle model, as we liked the idea of constant testing. Most of us are new to creating an application like this, thought that we'd likely be making many mistakes, hence why we thought the constant testing of V-Shaped sounded enticing. In the end, things actually turned out quite differently, as we found a different SDLC (software development lifecycle) that fit our style of development much better.

During development, we had multiple instances where we realised that a feature, API, or other important aspect of the application might need to be implemented differently than originally intended. This led to us having more of a flexible development style, consistently making changes on the fly, which was completely different from the strict sequential nature of V-Shaped that we had once planned on following. Consequently, we had naturally become more of an Agile environment. We had weekly calls to check in with everyone, discussing what task they're currently working on and what they plan to work on next, resembling the standups that are commonplace in Agile environments. Of course, we would have preferred to have daily standups/meetings rather than weekly ones, but too many outside factors such

as work obligations and workloads from other courses restricted us from doing so.

Alongside the weekly standups, we also utilised a Kanban board to help us manage our tasks. During our check-in with our assigned teaching assistant, he recommended that we use GitHub Projects to help us keep track of who's doing what, what we need to do, and when it should be done, instead of just communicating it verbally like we'd been doing up until that point. Upon recommendation, we linked our repository to a GitHub Project, keeping track of our issues and their priority. Additionally, some of us used Trello (a separate Kanban board) to help keep track of our personal tasks from not only this course but from other courses as well, helping us to make sure that we get our project work done and our other coursework done, which we feel is relevant enough to mention for our particular situation. Admittedly, there were a few times where, instead of first putting our issues on the Github Project board, we skipped that step and just worked on the feature. Of course, we didn't do this without communicating with our team first via message, but we're aware that's not the best practice. We think this was due to the time constraints and the pressure we were feeling, to the point where we just wanted to focus on getting the actual development done, so there were times where we weren't as focused on what our actual development process looked like.

In the end, we found the SDLC that worked best for our flexible development environment was an Agile approach that incorporated elements of both Kanban and Scrum. Some challenges that came with choosing this approach included settling for weekly calls instead of the daily ones that you would typically see, as well as remembering to put our tasks onto the Kanban board and properly keep track of them.

# Overview of Implemented Features

ReactPedia, your go-to destination for mastering React, the cutting-edge JavaScript library for building user interfaces. Whether you're a beginner looking to dive into the world of React or an experienced developer aiming to deliver a well-organised start for beginners, ReactPedia is designed to cater to your learning needs.

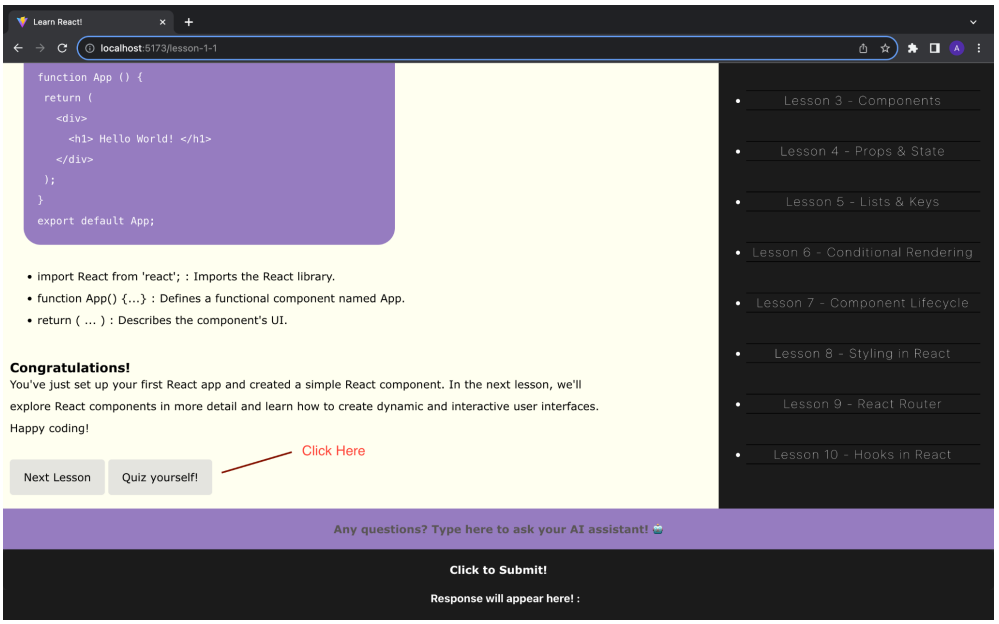
- **Linear-based progress lessons.**
    - **Lessons with active navigation state**: Clear navigation with an active header bar and lesson directory helps learners easily track their progress within a lesson.

    - **Lessons designed for linear progression**: Linear progression ensures a structured learning path, allowing learners to grasp React concepts in a logical order.
    - **Quizzes after lessons**: Immediate feedback on quiz performance helps learners identify areas for improvement and solidify their understanding of the material.
- **AI assistant**
    - **Q/As (Questions and Answers)**: A repository of questions and answers facilitates additional learning.
    - **Related questions**: A list of 5 related questions based on the AI's output are generated and available for the user to select for further interactions.
    - **Generate relevant information and questions**: Learners can explore commonly asked questions, share their insights.
- **Videos watching space**
    - **In-browser video environment**: An in-browser video environment allows seamless integration of video content directly within the learning platform.
    - **Search video - up to 25 videos**: The ability to search and access up to 25 relevant videos directly from the platform that offers a vast resource pool - Youtube.
    - **Add a video to playlist**: Enables users to add videos to their YouTube playlist directly from the learning platform, which provides a convenient way to organise and revisit content for future reference.
    - **Subscribe to a youtube channel**: Integration with YouTube allows learners to subscribe to channels directly from the platform, keeping them updated on new content and industry trends.
- **In-Browser Compiler Environment**: An in-browser compiler environment lets learners practise coding without switching between different tools. This feature enhances hands-on learning, allowing users to apply theoretical knowledge in a practical coding environment seamlessly.

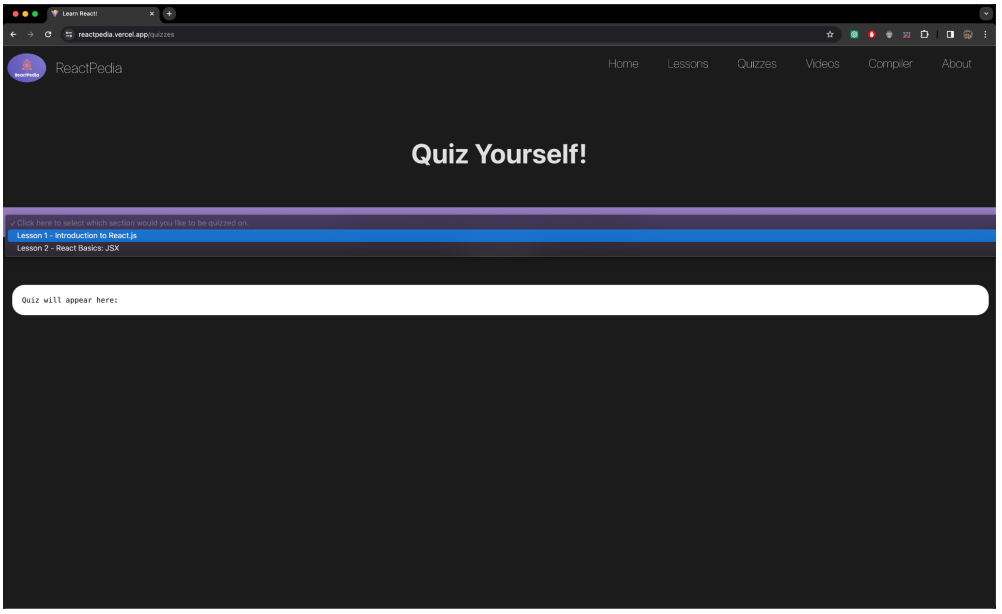
# APIs and Their Features

**OpenAI APIs:**

- **Quizzes after lesson - Feature 1**

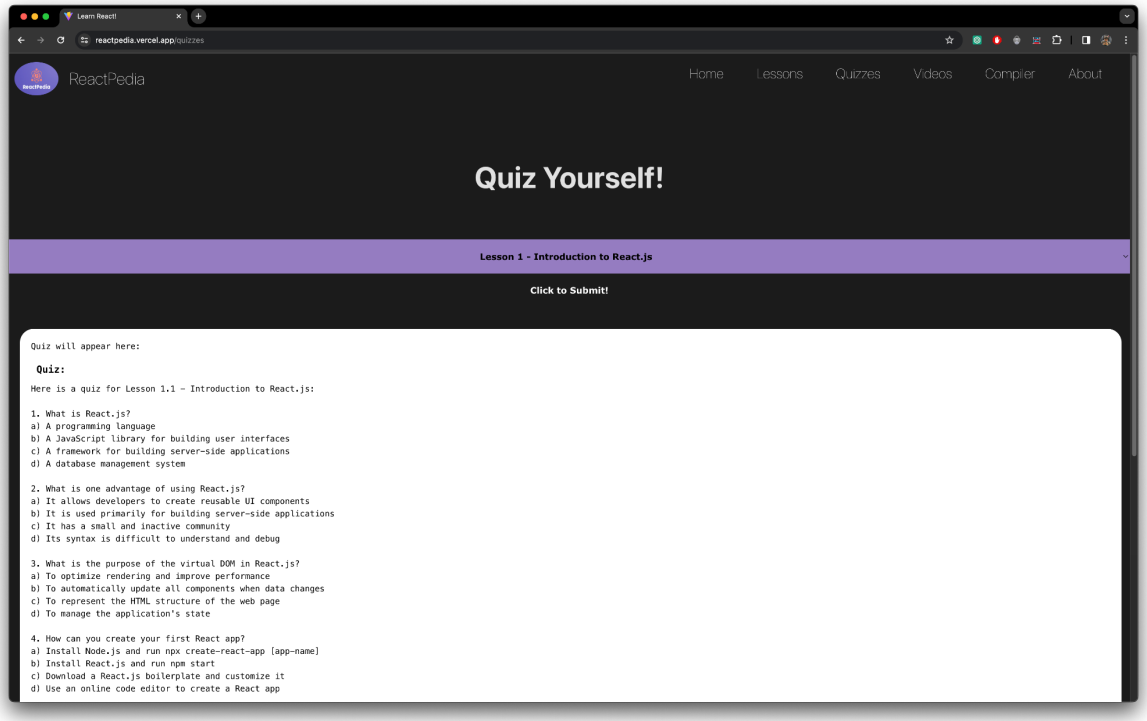



*From the any lesson page, click "Quiz yourself" at the bottom of the page and it will navigate to Quizzes.*



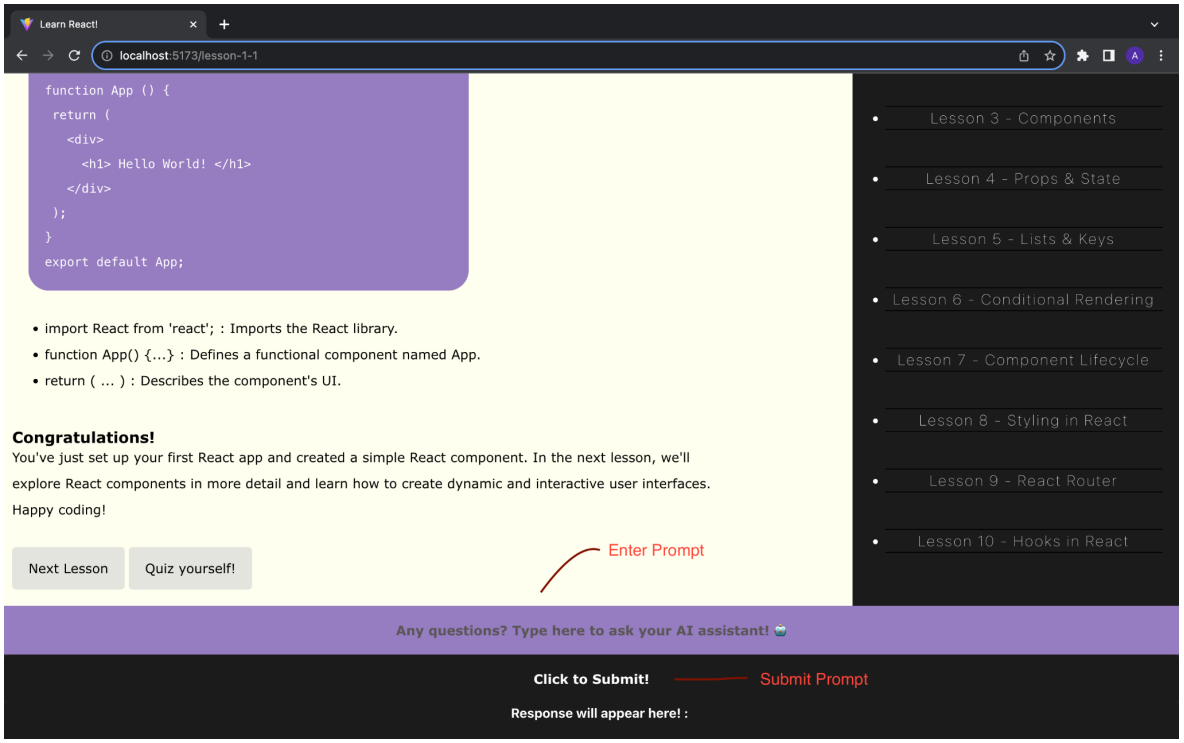

*Choose the lesson you want to have quizzes and submit.*

*(Lessons 1 - 10 are now available, though this image was captured before that was the case)*

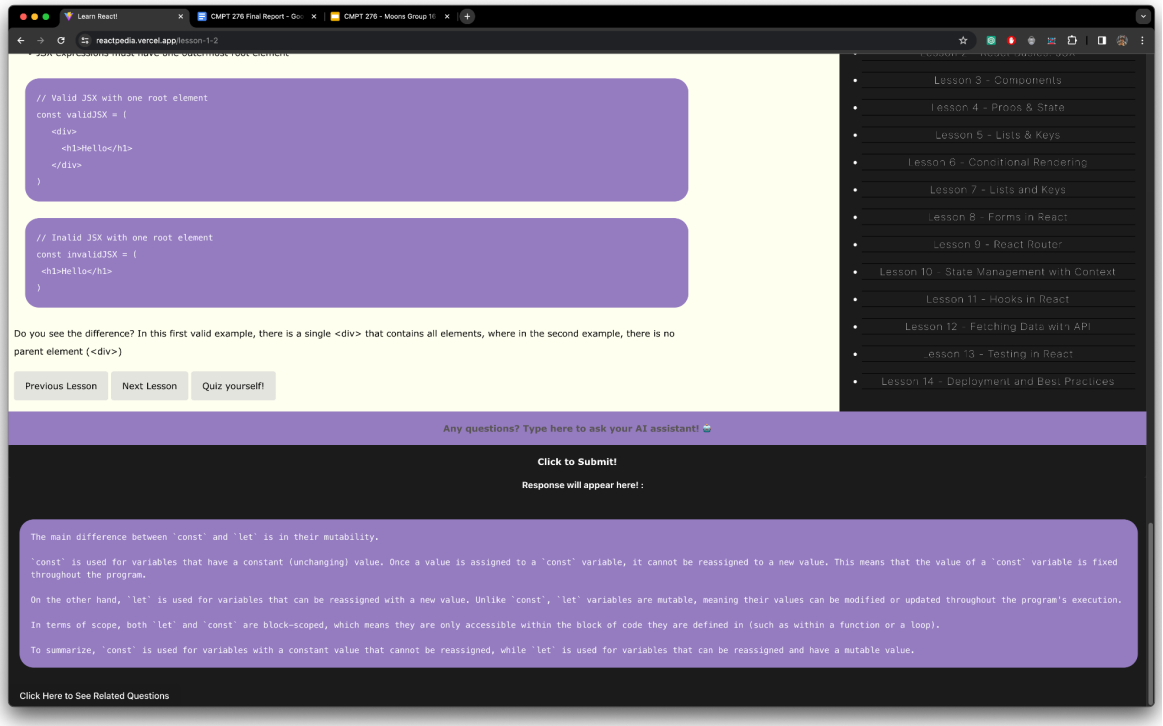



*The result should be as below.*

- **Q/As (Questions and Answers) - Feature 2**

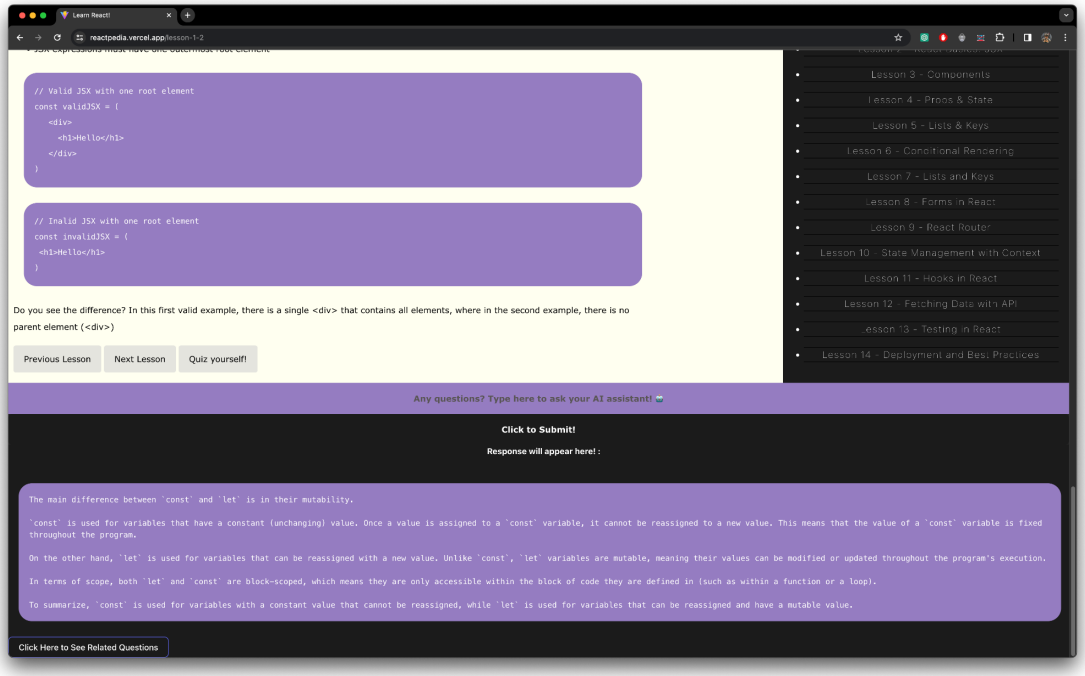



*Enter a question to the prompt and click "Click to Submit".*

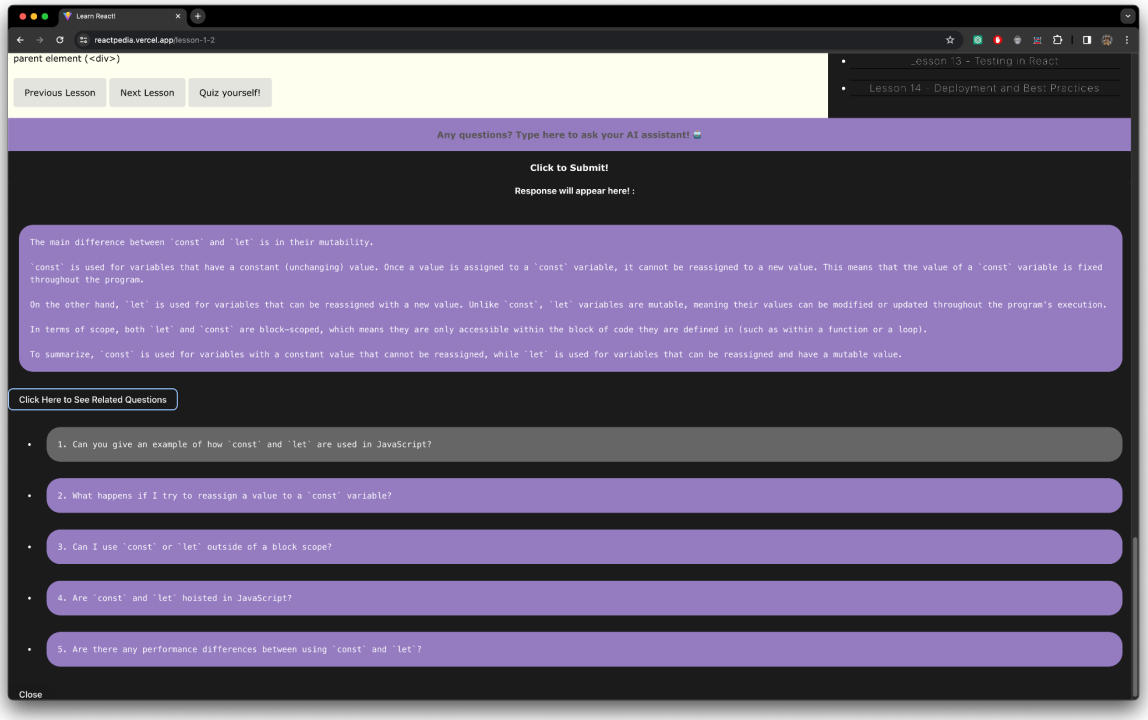



*The result should be as below.*

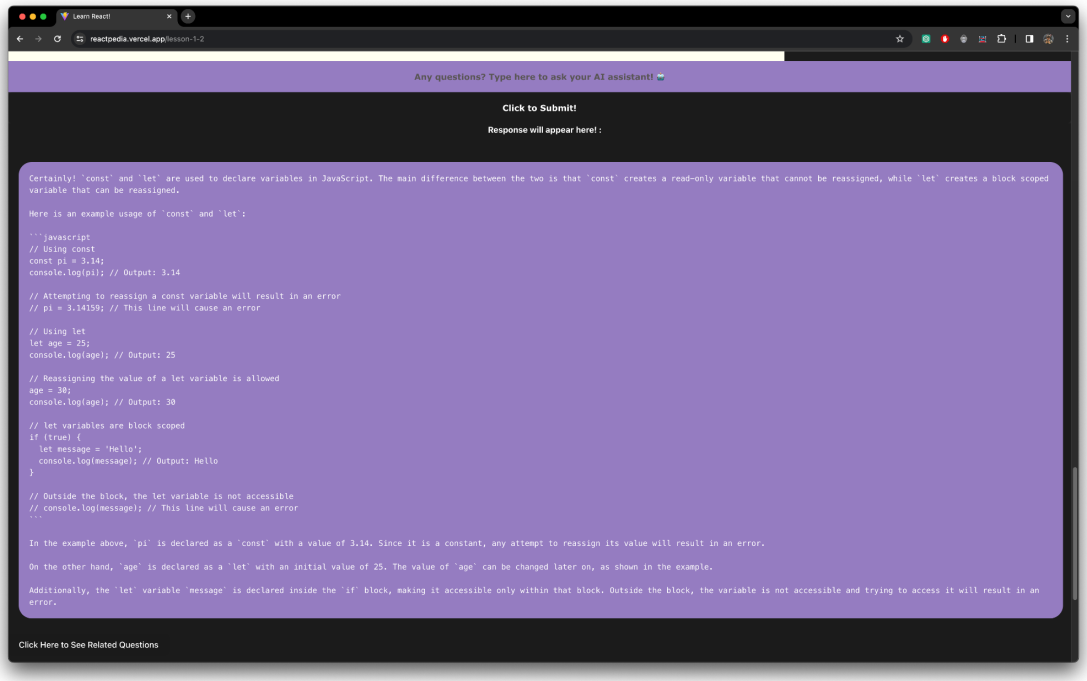
- **Generate relevant information and questions - Feature 3**




*Click at the bottom-left button to generate relevant questions.*



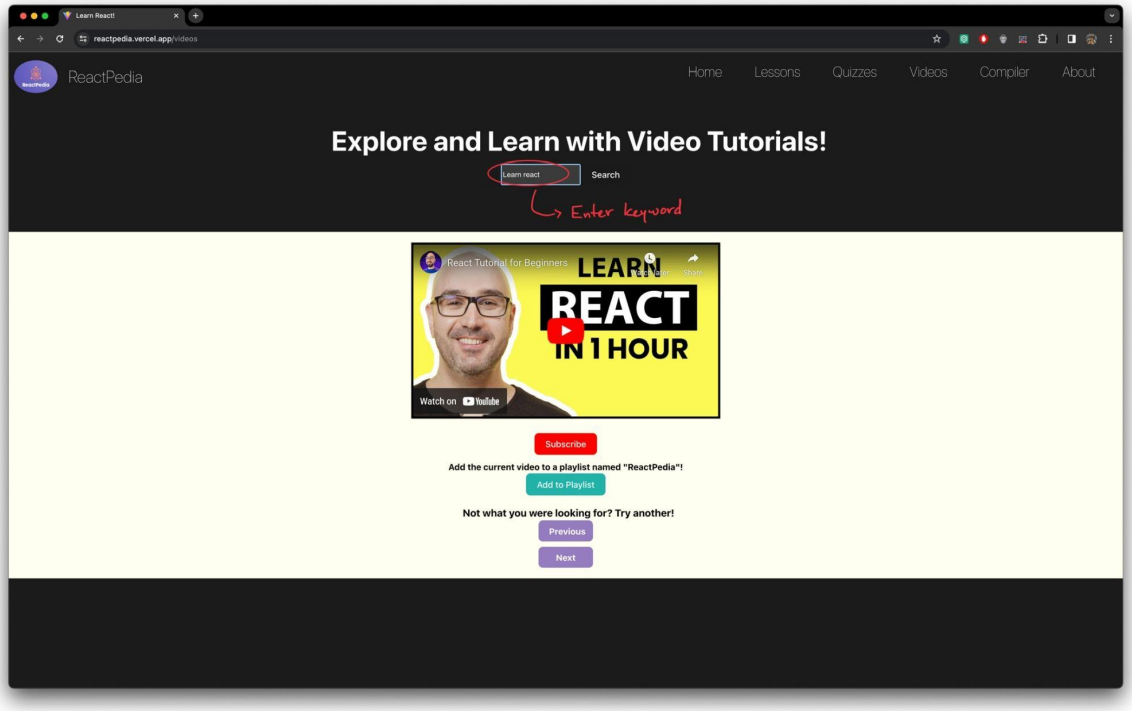

*Choose the suggested questions.*



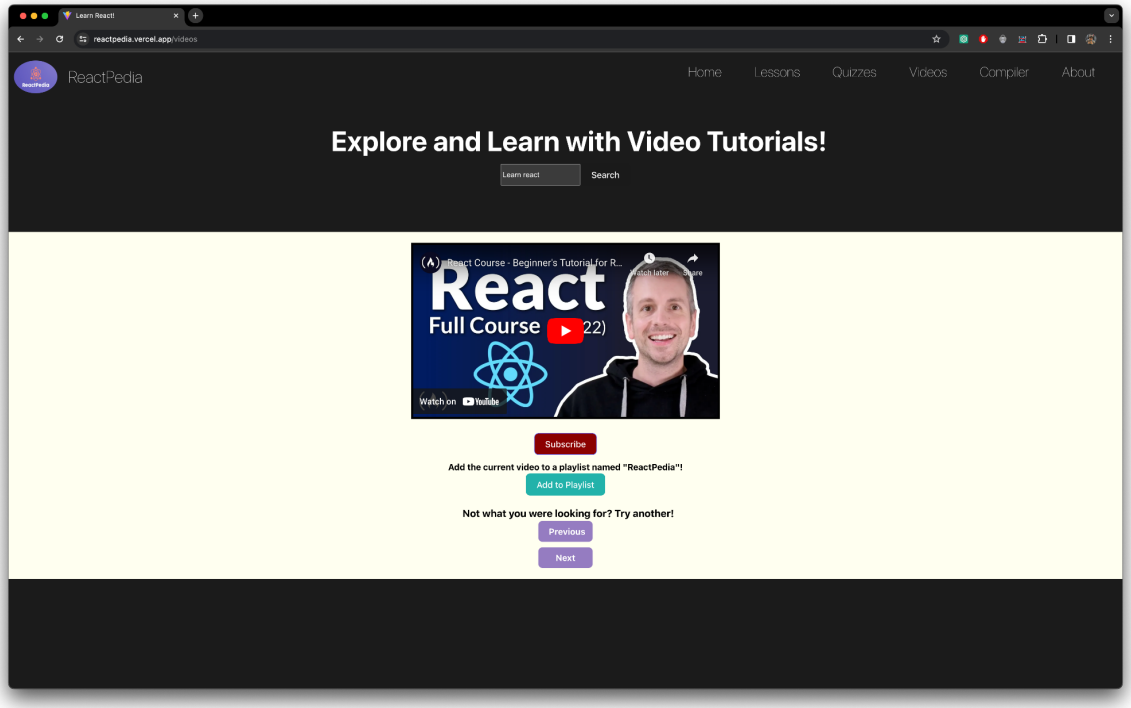

*The result should be as below.*

**Youtube Data v3 APIs:**

- **Search video - up to 25 videos - feature 1**

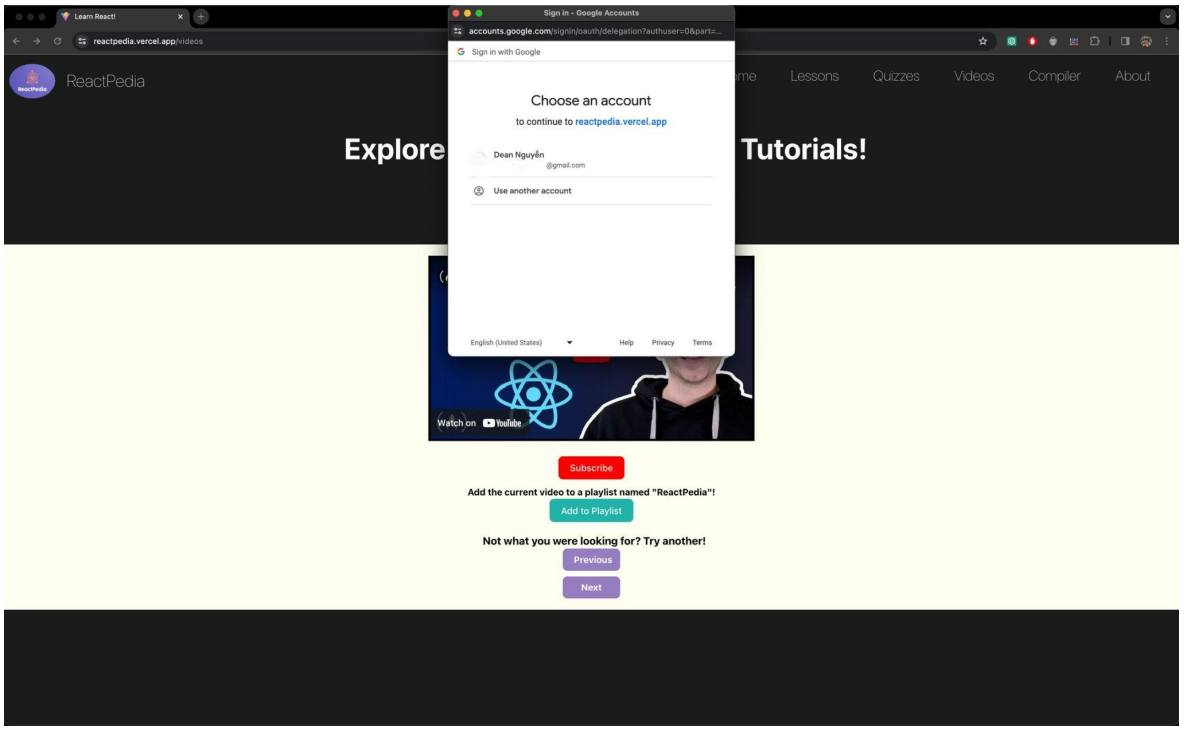



*On the Videos page, enter the keyword to search video and click "Search" after finishing.*

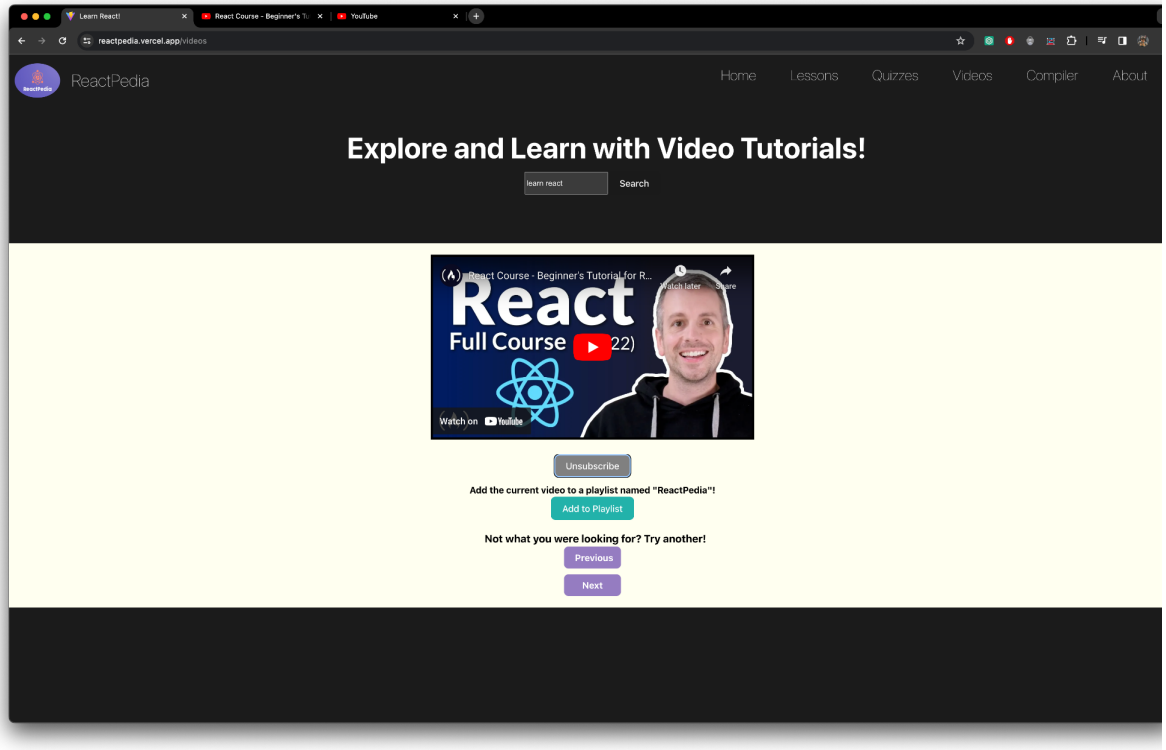



*The result should be as below.*

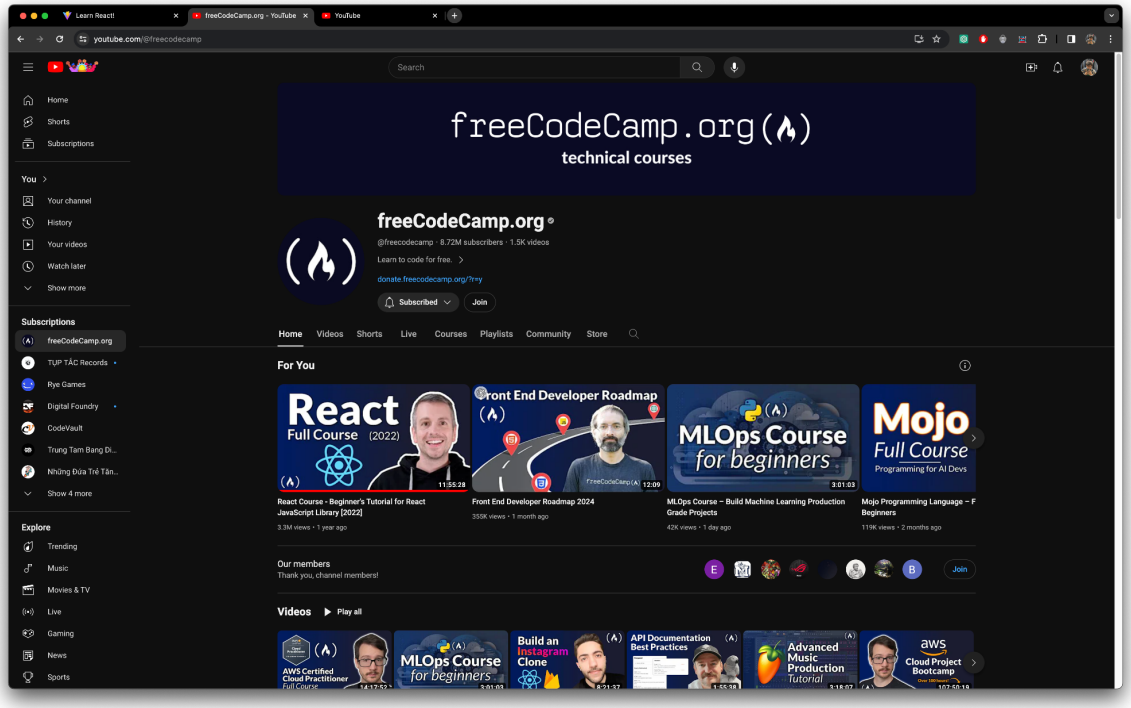
- **Subscribe a youtube channel - feature 2**




*A window will pop up to require access to login to your account.*

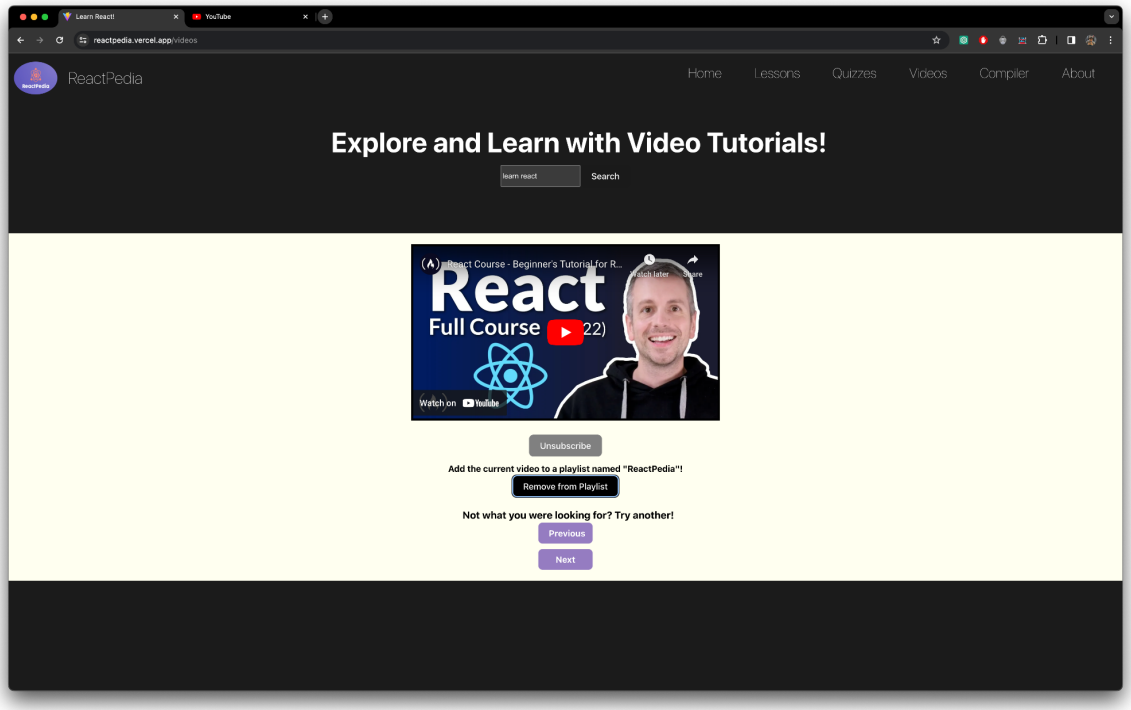



*After login, click "Subscribe" again. The result should be as below.*

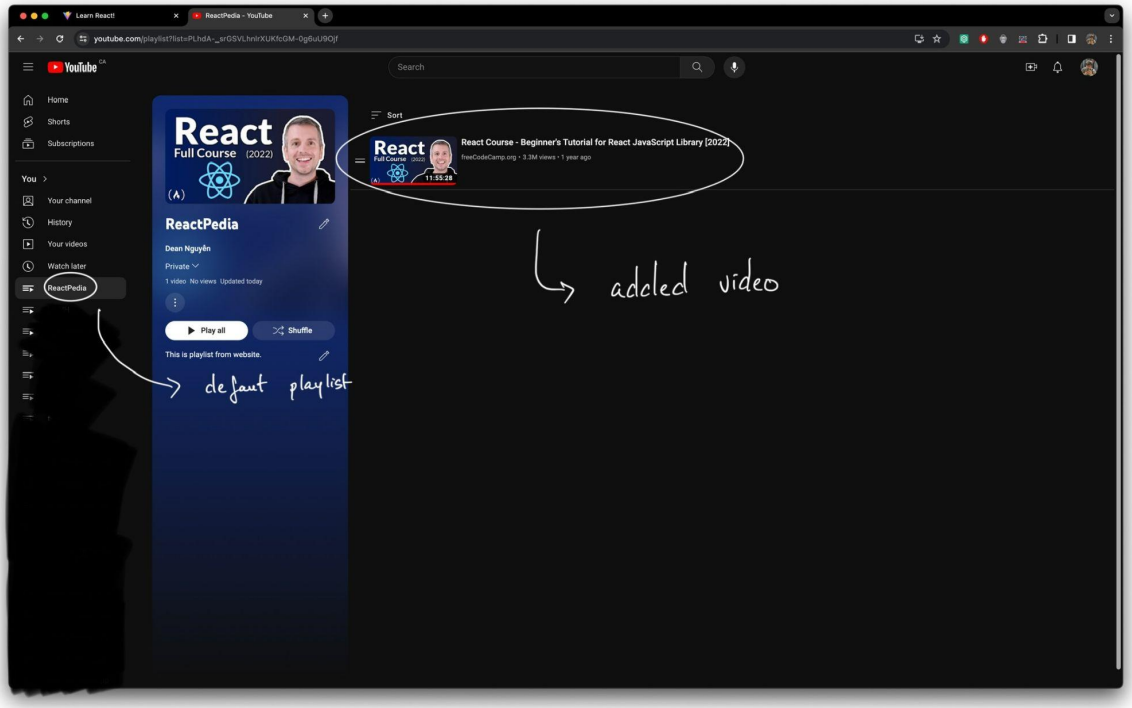



*Check again the channel, your account should have subscribed to the channel at this point.*

- **Add to playlist on youtube account**:




*As login or you can login as the same as the login process for subscribe, you can add video to a playlist on your Youtube account. They would appear on the page.*




*On the youtube page, a playlist named "ReactPedia" will appear. The videos will always be added to this playlist.*

# Testing

We chose Vitest and React Testing Library as the frameworks for our test suite. Vitest is a relatively new testing framework that is based on Vite (The build tool we chose over Create React App) and has a similar syntax and functionality compared to Jest. React Testing Library provides functions like Render and FireEvent to test our components. All of our tests use mocks to simulate API responses, since our APIs have quota limits and the outputs are unpredictable. Our test suite consists of 8 unit tests and 6 integration tests, and they can be found categorized in the `**test**` folder inside `**app**` and can be executed by running `**npm run test**` inside the `**app**` directory in our repository.

## Unit Tests

### onSearch.test.jsx

This test file tests the onSearch function for our YouTube search feature. It verifies that the correct GET request is sent.

**handleSwitch.test.jsx**

This test file tests the next and previous functions of the video navigation feature to ensure that the correct video is displayed.

**oauth.test.jsx**

This test ensures that the client ID environment variable exists and the OAuth client is initiated correctly.

**subscription.test.jsx**

This file has 4 test cases for different possible scenarios to test the correctness of the subscription feature.

**playlist.test.jsx**

Similar to subscription.test.jsx, this file also has 4 test cases for testing possible outcomes of the playlist feature. This assumes that the user's playlist ID already exists (test for this is covered below).

**findPlaylist.test.jsx**

This test tests the find playlist function that finds a playlist named "ReactPedia" in the user's account. There are 3 test cases for different scenarios.

**createPlaylist.test.jsx**

Closely related to findPlaylist, createPlaylist is a function that is called if findPlaylist fails to find a playlist with the name "ReactPedia". This is a simple test file that tests the function, with success and failure cases.

**chatbot.test.jsx**

The first OpenAI unit test, this test renders a lesson component and mocks an OpenAI query for the AI assistant/chatbot feature.

## Integration Tests

**search.test.jsx**

This integration test combines the onSearch and handleSwitch unit tests, where both features are tested together in a single workflow.

**subscription.test.jsx**

This test simulates a user's input when they first load the webpage, where an OAuth token is first obtained, then POST and DELETE requests are sent for subscriptions.

**playlist.test.jsx**

Similar to the subscription integration test, an OAuth token is obtained initially, and the video is added and deleted from a mocked playlist. In addition, the playlist is created through mocked responses from findPlaylist and createPlaylist functions.

**chatbot.test.jsx**

This test simulates the user's actions by sending multiple chat prompts and interacts with different components and features.

**suggestedQuestion.test.jsx**

This integration test is also an extension of the chatbot test that tests the related question feature. A simple test environment based on the chatbot test is set up initially, then suggested questions based on the initially generated response are generated and verified.

**quiz.test.jsx**

This test tests the functionality of the quiz feature, as well as ensuring that it works properly with module components such as the output display, dropdown box and more.

# CI/CD Infrastructure

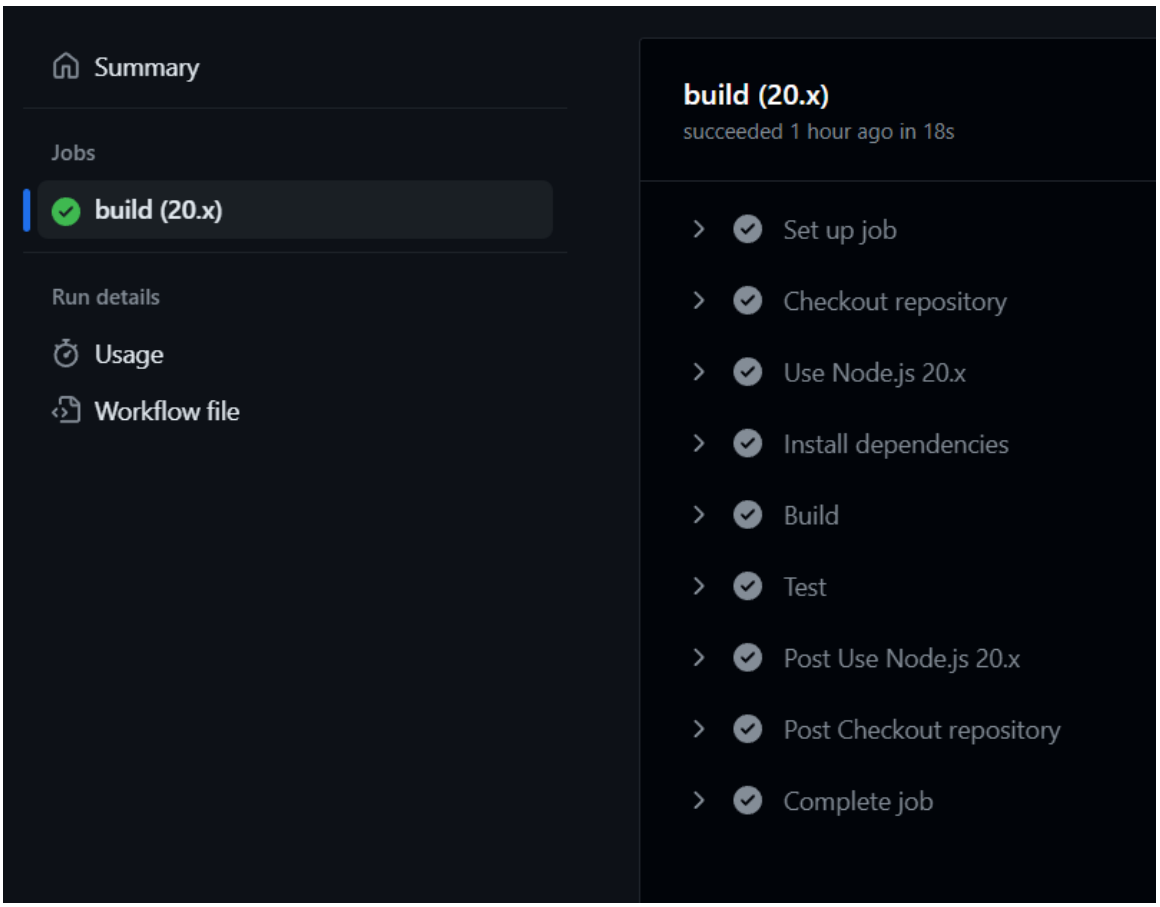
For our CI/CD infrastructure, we elected to use Git as our source control, GitHub Actions as our CI/CD application, npm and Node.js for building, Vitest and React Testing Library for testing, and Vercel for deployment. Our pipeline is as follows:

**Pull request created:**

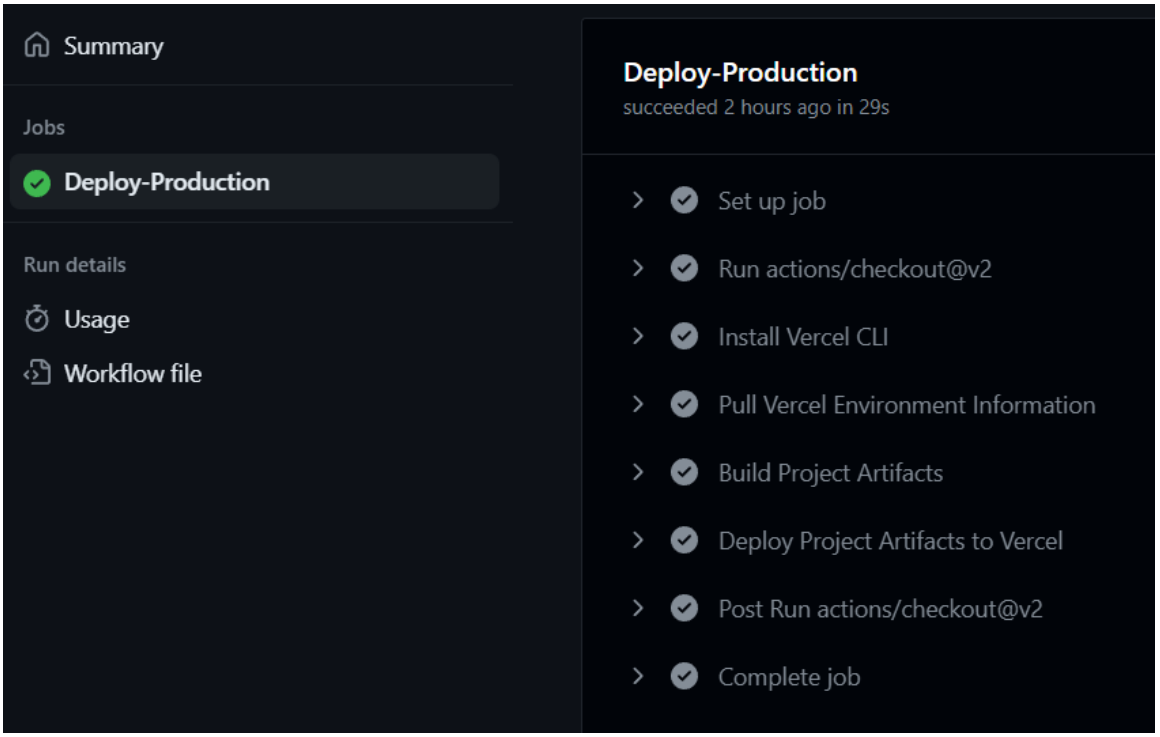
The React CI workflow starts. It installs all necessary dependencies, builds using Node.js 20.x, then runs the unit and integration tests via Vitest. If the build and tests succeed, then the checks pass, and the commit receives a checkmark.

**Pull request merged:**

In addition to the React CI workflow, the Vercel Deployment workflow runs as well. This workflow deploys our web application to Vercel if the React CI workflow completes without errors. This ensures that our Vercel application is up-to-date with our main branch.

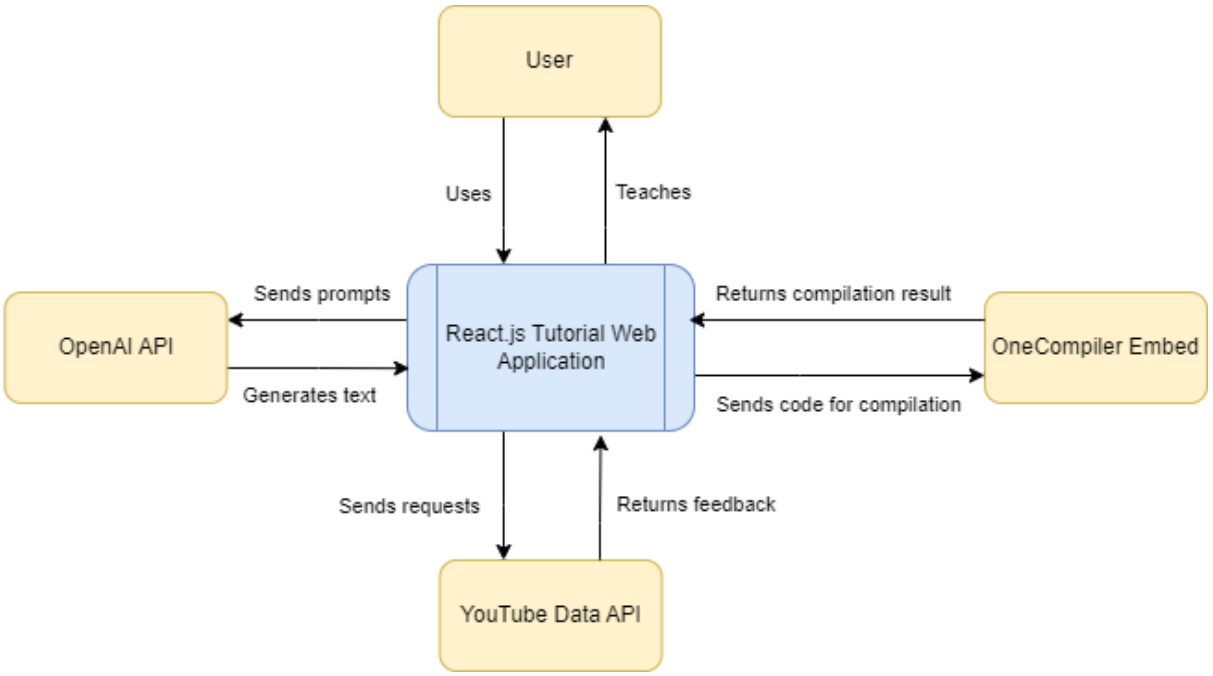



*React CI Workflow Process*




*Vercel Deployment Workflow Process*

# Data Flow Diagram




Our final high level data flow closely resembles our original diagram in project report 1. The only change we made is we chose to use OneCompiler instead of CodeSandbox as our in-browser IDE. Because of this, it is now a website embed instead of an API.

# Project Lessons Learnt and Challenges

Over the course of development, we certainly had many challenges, though for the most part we were able to learn from those challenges and become more knowledgeable developers because of them.

First and foremost, our biggest challenge was communication. There were instances where it wasn't clear who was working on what. This problem culminated with the integration of the YouTube API's playlist feature, where two of us ended up working on the feature at once, which was not the original plan. This and many other experiences emphasised the importance of being transparent with one another, and making sure that each member is frequently checking with the messages sent in the communication channels when they can.

Another challenge that we faced were the tight deadlines, which in turn taught us a lot about time management. For a team that had never created something like this before, creating an

entire web application in three weeks was a daunting task and a stressful experience. There were even times where we thought we might have to cut out some of the features that we wanted to include. To combat these tight deadlines, each member meticulously planned out their days, leaving designated time slots for working on development of the application. As a result of this process, we were not only able to meet the tight deadlines, but we also as a collective learned a lot about how to properly distribute our time amongst our different classes and this class, which certainly helped enhance our time management skills.

The integration of the YouTube API and OpenAI API proved to be quite the challenge as well. Prior to this class, none of us had much experience with integrating APIs into our projects, so as expected, it took us a while to figure out where to even start. Once we had obtained the respective API keys after what took us longer than we had originally thought, deciphering the API documentation presented another steep learning curve, and took even longer. Going through the documentation and understanding how to utilise the API's features required much patience and further research. Eventually, we were able to incorporate the features that we wanted from each API, but it certainly didn't come easily.

For the most part we've focused on challenges thus far, but one thing that we can definitively say is that we were able to take away from this experience was the use of Git and GitHub. By constantly using git push, git commit, git checkout, git fetch, git pull, and merging pull requests, we all deepened our understanding of version control systems immensely. Through constantly applying these different commands, we not only learned how to fully utilise them, but also how to resolve conflicts when they presented themselves. These hands-on experiences with Git and Github have allowed us to expand our developer toolkit, adding a skill that we will likely use far into the future.

# Work Division

| Group Members | Report #1 | Check in |
|---|---|---|
| Jashan | Made the schedule and wrote some API features. | Tech Stack and API presentation |
| Anthony | Work breakdown structure, prototypes, wireframes. | Home page, a couple of content modules. |
| Clement | Data flow diagrams, user stories, proofreading. | CI/CD infrastructure |
| Dean | Check-in video, API overview, APIs and features. | Research APIs for infrastructure of the project. |

| Group Members | Final Presentation | Report #2 |
|---|---|---|
| Jashan | Introduction , API high-level overview, and personal takeaways. CI/CD diagram. | Project overview, work division, and what worked |
| Anthony | Takeaways and challenges, demo video, application overview. | Takeaways and challenges, SDLC and what worked. |
| Clement | CI/CD infrastructure, personal challenges and takeaways, Q&A | CI/CD infrastructure, testing, data flow diagram |
| Dean | Application feature, personal challenges and takeaways, APIs features. | High-level implemented feature, APIs applications. |

| Group Members | Web Application |
|---|---|
| Jashan | Embedded compiler features and webpage, about page and Vercel deployment |
| Anthony | OpenAI features (lesson learning assistant, quizzes), general CSS/user interface for website, content modules. |
| Clement | CI/CD, YouTube features (search, subscribe, playlist), OpenAI feature (related questions), testing suites |
| Dean | YouTube feature (OAuth2.0, search, subscribe, playlist). |