

Code Mate

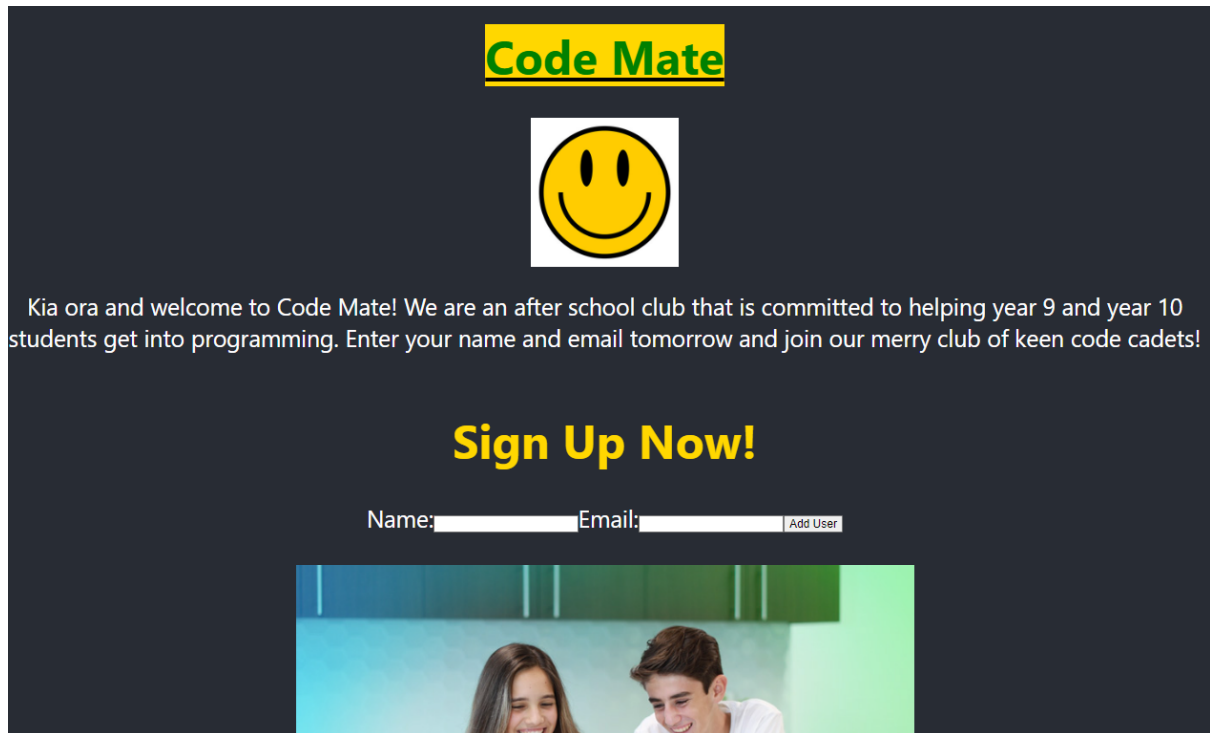


Figure 1 Code Mate landing page

I opted for Scenario B, an authentic project involving the development of the Code Mate app. This application is a comprehensive system for event management, information dissemination, and membership administration.

The project originated as a proposal and gained approval from my tutor, Sarmad. Here is a brief overview of the project.

Story Brief

The Code Mate Club Management System is a software solution meticulously crafted to streamline and enrich the administration of coding club activities. This platform empowers administrators to present upcoming events, allowing students to access event details and information about those organising and participating.

The application extends its functionalities to students, enabling them to explore coding courses, learn about their instructors, and connect with fellow students. Additionally, the system facilitates prospective members in joining the club by submitting their details through an online form seamlessly integrated with a Postgres database.

The SQLITE3 database captures and stores new members' details and retrieves data on existing club members.

Part 1: Planning and Web Application

Code Mate Club Management & Member Engagement App

Epic

The Code Mate Club Management & Member Engagement epic aims to establish a cohesive system for the Code Mate Club, optimising club activities and elevating member engagement. This epic encompasses several key features:

1. **Event Management System:** This component empowers administrators to schedule and oversee club events while students gain access to event details and participant information.
2. **News Hub:** Students can explore coding course instructor profiles and connect with fellow members through this centralised hub.
3. **Membership Portal:** Prospective members can seamlessly join the club via an online form, and their information is securely stored in a SQLITE3 database.
4. **Member Directory:** A comprehensive directory containing member names and email addresses is implemented to foster community building.
5. **Tutor Hub:** Contact details for tutors and instructors are stored, facilitating easy communication within the club.

Benefits:

- Augmented engagement and community building.
- Streamlined administration, leading to efficient event management.
- Enhanced student networking and improved communication with tutors.
- Informed decision-making based on data for club activities.

User Stories: Students, Admin, and Staff

The application will function as an event management, information hub, and membership management system.

Administrator

As an administrator, I want to create and schedule events so staff and students can view event names, dates, times, and locations.

As an administrator, I want to receive and manage membership applications submitted online so that I can effectively oversee all club members.

As an administrator, I want to provide tutor contact details so that students can find information about instructors involved in the club.

As an administrator, I want an app that shows information about club members so that students can find information about peers they can contact.

Staff

As a staff member, I want to see the number of students involved in the programme so that I can plan my classes to cater for different class sizes.

Student

As a student, I want to view upcoming events so that I can access event details and information about the individuals involved.

As a student, I want to explore instructors' profiles so that I can learn more about their expertise and teaching style.

As a student, I want to connect with the club through a simple sign-up form so that I can engage with other students immediately.

As a student, I want to access a member directory through an SQLITE3 database, including current club members' names and email addresses, so that I can view other members and get their contact details.

As a student, I want to find the contact details of my course instructors, including their names and email addresses, so that I can easily contact them.

Project Delivery

The Code Mate club, management system project was completed through a meticulously crafted plan. The project underwent distinct stages, including requirements analysis, design, development, testing, and deployment. Critical roles were assigned in the following manner:

Role and Plan

Week 1-9

Project Manager: Overseeing project coordination, scheduling, and milestone tracking was a critical and demanding role.

Week 3

Business Analyst: Gathering user requirements, crafting user stories, and aligning user needs with system features were imperative for building the desired app—a valuable learning experience.

Week 4

UI/UX Designer: Creating an intuitive and accessible user interface design presented a challenging yet creatively rewarding experience.

Week 5-6

Front-end Developer: Translating the UI design into a responsive and accessible interface did not pose too many hurdles.

Week 7-9

Back-end Developer: Developing core functionalities, including membership, event scheduling, and resource management, presented the greatest challenge in the building stage.

Week 7-9

Database Administrator: Designing and managing the SQLITE3 database for storing relevant data was crucial.

Week 8-9

QA Tester: Conducting rigorous testing across functionalities to ensure functionality, performance, security, and compatibility was often overlooked but played a vital role in the project. I believe my efforts were crucial in thoroughly testing the app.

Project Execution and Deviations

The project adhered closely to the planned timeline but experienced minor deviations:

Technical Challenges: Integrating user authentication with the existing website unexpectedly presented complexities, resulting in a slight development delay.

Data Privacy Concerns: Privacy concerns emerged during testing, prompting enhanced security measures such as data encryption and access controls.

Mitigation Strategies: The team effectively addressed these deviations through the following measures:

Technical Expert Consultation: Seeking expert guidance from online resources was crucial in assisting the project delivery.

Privacy Compliance: Rigorous tests were conducted to ensure the website met stringent privacy requirements.

Project Outcome

Notwithstanding minor deviations, the "Code Mate Club Management System" project reached completion, meeting both functional and non-functional requirements. The result is an efficient and simple platform for club management. Through meticulous planning, necessary adjustments, and effective mitigation strategies, the project holds value for administrators, staff, and students.

Part 2: Build Web Application

Short Summary of the Code Push to GitHub

I: The successful push of the app to GitHub was executed using the GitHub Desktop tool. While I had previously practised utilising the Visual Studio Code IDE for file management on my GitHub repository, I was eager to learn the Desktop tool for this task. I am pleased to have developed proficiency in both GUI and CLI tools. The upload transpired without encountering any errors.

II: No actual errors were encountered during the process. I researched Stack Overflow and Google to find solutions. Additionally, I discovered the convenience of making code adjustments directly within GitHub. Throughout the app testing phase, I utilised debugging tools and stop and breakpoint features to incrementally test my code.

III: To elevate the user experience of the Code Mate Club Management System, considerations for implementation include intuitive navigation, personalised dashboards, event reminders, course recommendations, interactive instructor profiles, discussion forums, membership application tracking, robust search and filters, mobile responsiveness, feedback mechanisms, onboarding tutorials, and stringent data security. These enhancements aim to provide an engaging, user-friendly, and personalised environment for administrators, students, and prospective members, fostering a strong sense of community and facilitating seamless access to club activities, courses, and member interactions.

Part 3: Testing

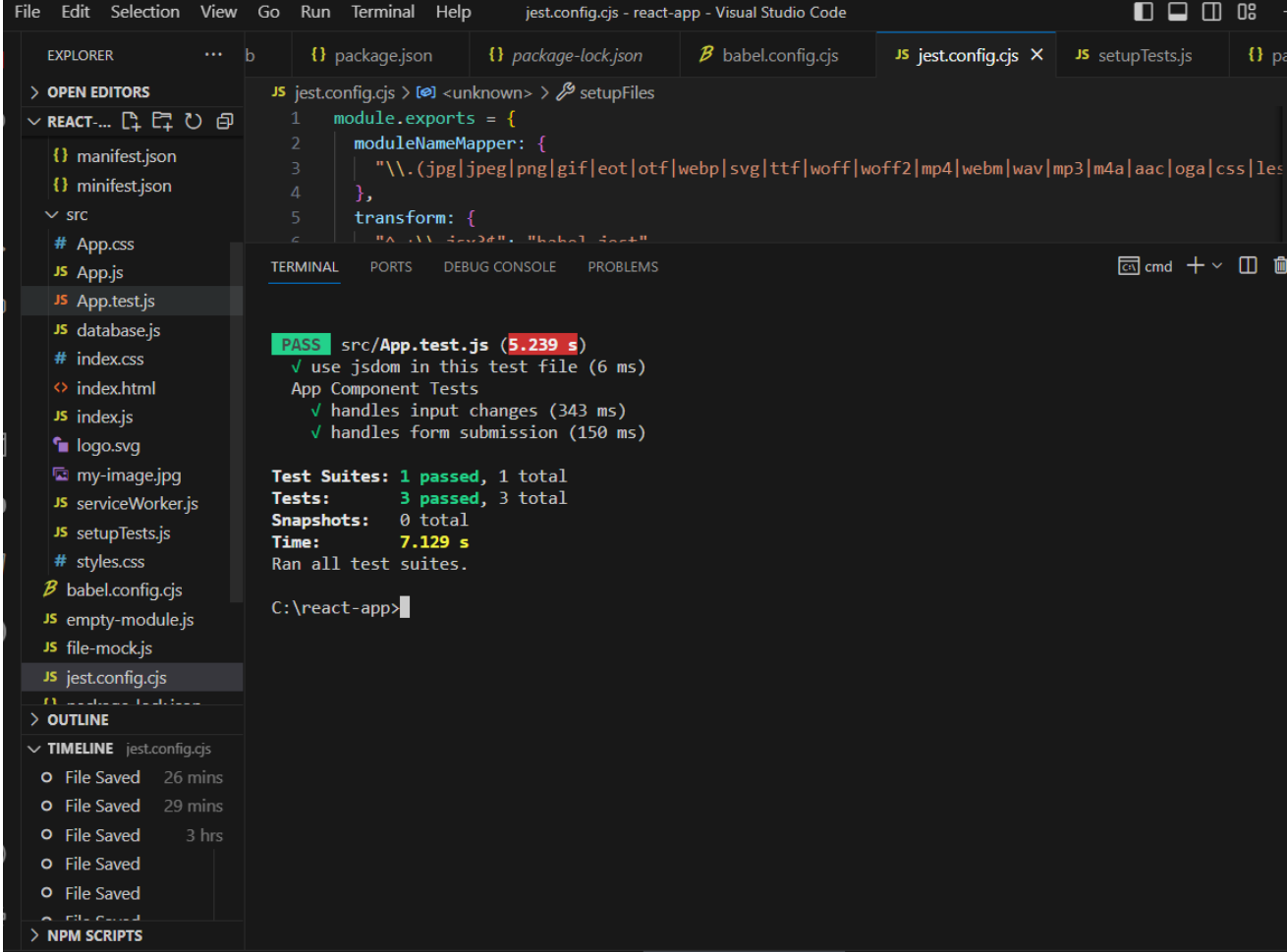
Testing the React app was a crucial aspect of ensuring its functionality and reliability, with a particular focus on the form, which played a vital role in collecting user input for subsequent storage in the database. The test suite leveraged Jest to execute various tests, validating that the application and its integration with the database met the specified requirements. The key tests aimed to capture the app's rendering, the crucial handling of input changes, and the submission of user details. The results were affirmative and confirmed the functionality of the app.

Conducted within the Visual Studio Code (VSC) Integrated Development Environment (IDE) through the terminal, the command "**npx jest --runInBand**" initiated the testing process. The positive outcomes of these tests provided a sense of assurance to the client administrator seeking a robust product that can benefit both themselves, the teaching staff, and, ultimately, the students.

Given the app's core data capture, storage, and representation objectives, the form's proper functionality was paramount. All tests, affirming the app's seamless interaction with the database, contributed to the client's confidence in the reliability of the service, especially in meeting the educational needs of their students. It is arguable that while further testing can always be done for the future resilience of the app, it is ready and fit for production.

Testing

Testing was the hardest part of the app. It required time for familiarisation with the testing tools and the testing environment. There was much to consider when testing the app, such as what the most appropriate tool for testing the app would be. There are many options, and it took time to balance the pros and cons between them. Ultimately, I selected Jest as it is a respected tool among developers and builders of apps using JavaScript frameworks such as React.



Tests of the app were applied using the Jest testing suite. These were applied to the web application's primary component, and all results were successfully met. The JSDOM testing library was used to test the application components.

```
File Edit Selection View Go Run Terminal Help
App.test.js - react-app - Visual Studio Code

EXPLORER
> OPEN EDITORS
  REACT-APP
  > my-express-app
  > node_modules
  > node-mysql-backend
  > public
    <> index.html
    {} manifest.json
    {} minifest.json
  > src
    # App.css
    JS App.js
    JS App.test.js
    JS database.js
    # index.css
    <> index.html
    JS index.js
    logo.svg
    my-image.jpg
    JS serviceWorker.js
    JS setupTests.js
    # styles.css
    babel.config.cjs
    JS empty-module.js
    JS file-mock.js
    JS jest.config.cjs
    {} package-lock.json
    {} package.json
    README.md
    JS server.js
    whitecliffes.db

  > OUTLINE
  > TIMELINE App.test.js
    File Saved now
    File Saved 2 mins
    File Saved 3 mins
    Undo / Redo
    Undo / Redo
    File Saved 4 mins
    File Saved 27 mins

JS App.test.js X JS empty-module.js JS file-mock.js my-image.jpg # App.css JS server.js whitecliffes.db

src > JS App.test.js > describe('App Component Tests') callback > test('handles form submission') callback
1
2 import { JSDOM } from 'jsdom';
3 const dom = new JSDOM();
4
5 global.document = dom.window.document;
6 global.window = dom.window;
7
8 test('use jsdom in this test file', () => {
9   const element = document.createElement('div');
10  expect(element).not.toBeNull();
11 });
12
13 jest.mock('./my-image.jpg', () => 'path/to/mock-image.jpg');
14 import React from 'react';
15 import { render, screen, fireEvent, waitFor } from '@testing-library/react';
16 import App from './App';
17
18 describe('App Component Tests', () => {
19
20   test('handles input changes', () => {
21     render(<App />);
22
23     // Simulate input changes
24     fireEvent.change(screen.getByLabelText(/name/i), { target: { value: 'John' } });
25     fireEvent.change(screen.getByLabelText(/email/i), { target: { value: 'john@example.com' } });
26
27   });
28
29   test('handles form submission', async () => {
30     render(<App />);
31
32     // Mock fetch to resolve successfully
33     global.fetch = jest.fn().mockResolvedValueOnce({ ok: true });
34
35     // Simulate input changes
36     fireEvent.change(screen.getByLabelText(/name/i), { target: { value: 'John' } });
37     fireEvent.change(screen.getByLabelText(/email/i), { target: { value: 'john@example.com' } });
38
39     // Simulate form submission
40     fireEvent.click(screen.getByText(/add user/i));
41
42   });
43
44 });
```

Using Jest, a test file comprising several primary test areas was created. This included tests of the input functionality of the app along with tests of the forms. The test results highlighted that the app's key features were functioning satisfactorily.


```
1 module.exports = {
2   moduleNameMapper: {
3     "\\.(jpg|jpeg|png|gif|eot|otf|webp|svg|ttf|woff|woff2|mp4|webm|wav|mp3|m4a|aac|oga|css|less)$": "identity-obj-proxy",
4   },
5   transform: {
6     "\\..\\.\\.\\.\\.jsx?$": "babel-jest"
7   },
8   testEnvironment: 'jsdom',
9   setupFiles: [
10    '<rootDir>/src/setupTests.js', // Add this line to include the setupTests.js file
11  ],
12  // ... other configurations
13 };
```

A Jest **jest.config.cjs** file was developed to clarify and outline how the tests would be executed. This file determined how the tests would be run, the settings that would be applied to these tests, and how the code could be transformed for testing and evaluative purposes.

```
{
  "name": "create-react-app",
  "version": "0.1.0",
  "private": true,
  "installConfig": {
    "pnpm": true
  },
  "dependencies": {
    "@toolz/is-a-regular-object": "^1.0.1",
    "aws-sdk": "^2.1454.0",
    "axios": "^1.5.0",
    "body-parser": "^1.20.2",
    "concurrently": "^8.2.1",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "mongodb": "^6.0.0",
    "mysql": "^2.18.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "sqlite3": "^5.1.6"
  },
  "scripts": {
    "client": "react-scripts start",
    "server": "node server.js",
    "dev": "concurrently \"npm run server\" \"npm run client\"",
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "jest --config jest.config.cjs",
    "eject": "react-scripts eject"
  },
  "browserslist": [
    ">0.2%",
    "not dead",
    "not ie <= 11",
    "not op_mini all"
  ]
}
```

The **package.json** file contains a variety of dependencies and dev-dependencies, which are crucial for the app to work. The **package.json** file also contains scripts which allow the app to be executed through the terminal.

Part 4: Reflection

Throughout the development of the Code Mate React app, my journey was marked by both successes and challenges, contributing significantly to my learning experience. One notable success was the smooth integration of user sign-up functionality, simplifying the process of joining the code club. This accomplishment resulted from meticulous planning and a commitment to best practices, ultimately enhancing the overall user experience.

On the flip side, a noteworthy challenge arose while handling data migrations. The complexity of adapting the database schema to accommodate evolving model changes exceeded initial expectations. I turned to Reacts comprehensive documentation to overcome this obstacle, engaged with the developer community, and adopted a step-by-step approach to database migrations. This experience underscored the importance of thorough version control and disciplined database management.

In hindsight, the project underscored the critical role of planning, adherence to best practices, and the support derived from a collaborative developer community when confronted with unforeseen challenges. The Code Mate project has proven to be rewarding, highlighting the significance of successful achievements and the valuable lessons gleaned from navigating and overcoming obstacles.

Appendix

Software Requirements Document: Code Mate

1. Introduction

The Code Mate Club Management System app allows Year 9 and Year 10 students to join an afternoon coding club. It will enable administrators to present upcoming events and for students to see them.

The application allows students to enter their name and email address and see the terms and emails of fellow students they can connect with at the club.

The web application also brings up the names of the instructors and provides information such as the time of the weekly coding club activity.

The names of the users and the instructors are stored and fetched from an SQLITE3 database.

This document outlines the functional and non-functional requirements for the system.






2. Purpose

This system aims to facilitate efficient organisation, communication, and coordination of activities, participants, and resources within the after-school code club.

The epic of the app is to **attract** members to contact/**join** the club through the completion and submission of the form. This form takes in the data and stores it in an SQLite DB. Data pulled from an SQLite DB shows students and tutors involved in the coding club.

3. Scope

The system will cover the following aspects:

-  Membership CTA, including contact form
-  Activity scheduling and events displayed
-  Course and instructor information
-  Club contact details
-  Private tutor details pulled from an SQLite DB

4. Functional Requirements

4.1. User requirements

-  Allow users to join/contact the club.

- 📖 Allows users to see upcoming events
- 📖 Allows users to see course and instructor information
- 📖 Allows the user to see club contact information

4.2. Activity Management

- 📖 Enable staff to create, schedule, and manage club activities
- 📖 Include activity details such as date, time, location, and description

4.3. Attendance Tracking

- 📖 Allow staff to promote the club to existing and future members

4.4. Communication

- 📖 Provide a system for members and staff to communicate and see their classes and timetables

4.5. Resource Management

- 📖 Allow staff to allocate and manage club resources (equipment, materials, rooms)

5. Non-Functional Requirements

5.1. Usability

- 📖 The user interface should be intuitive and easy to navigate
- 📖 The user contact form should be simple to find and easy to use
- 📖 The app should be accessible for users with accessibility issues
- 📖 The application of Te Reo Māori would be an advantage for cultural inclusivity

5.2. Performance

- 📖 The system should handle concurrent users without significant performance degradation
- 📖 Response time for user interactions should be within 2 seconds

5.3. Security

- 📖 User data should be securely stored and encrypted

5.4. Reliability

- 📖 The system should have a backup and recovery mechanism in case of data loss
- 📖 Scheduled maintenance should be performed during off-peak hours

5.5. Compatibility

- 💻 The system should be accessible from various devices (desktop, tablet, mobile)
- 💻 Support modern web browsers (Chrome, Firefox, Safari, Edge)

6. Constraints

- 💻 The system will be developed using the React Framework. Data will be stored and retrieved from a SQLite database.
- 💻 The budget for development and implementation is limited to \$500.00.

7. Assumptions

- 💻 Users have basic computer literacy skills.
- 💻 The club staff will provide the necessary information for member profiles and activity details.

8. Dependencies

- 💻 Integration with the club's existing website for user authentication.

9. Risks

- 💻 Data privacy concerns storing member information.
- 💻 Technical challenges in data migration and storage.

10. Conclusion

The after-school Code Mate Club Management System aims to enhance the overall experience of club members and staff by providing a user-friendly platform for efficient management, communication, and resource allocation. By fulfilling the outlined requirements, the system will contribute to the successful operation of the after-school club.

SQLITE3 DB

This database was used for the simplicity of sending and fetching data from the app to the DB. It can be run through both the CLI and there is also a GUI.

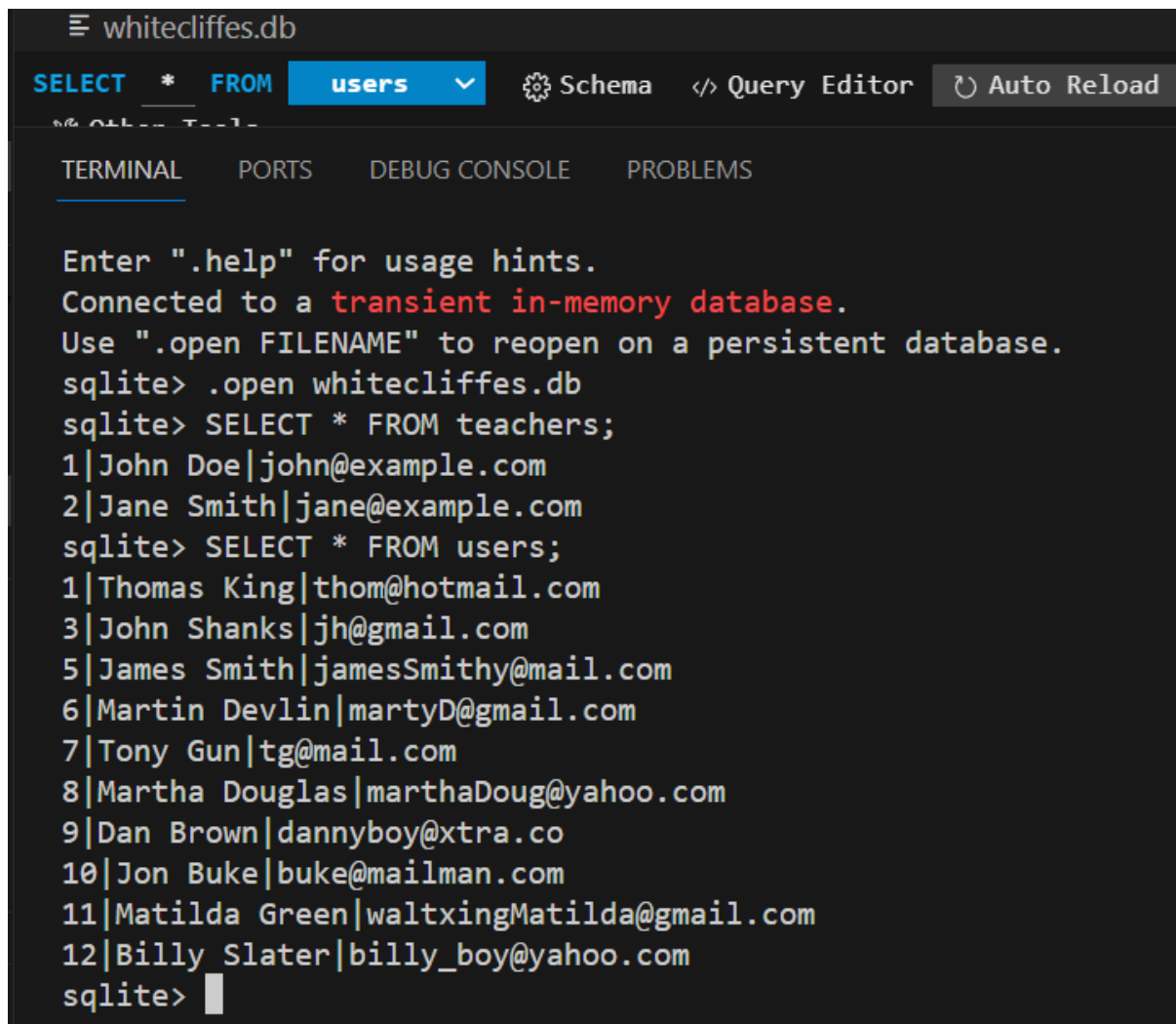
whitecliffes.db

SELECT * FROM **users** ⚙ Schema </> Query Editor ↻ Auto Reload

✂ Other Tools...

	id <i>INTEGER PRIMARY KEY AUTOINCREMENT</i>	name <i>TEXT</i>	email <i>TEXT</i>
1	1	Thomas King	thom@hotmail.com
2	3	John Shanks	jh@gmail.com
3	5	James Smith	jamesSmithy@mail.c...
4	6	Martin Devlin	martyD@gmail.com
5	7	Tony Gun	tg@mail.com
6	8	Martha Douglas	marthaDoug@yahoo.c...
7	9	Dan Brown	dannyboy@xtra.co
8	10	Jon Buke	buke@mailman.com
9	11	Matilda Green	waltxingMatilda@gm...
10	12	Billy Slater	billy_boy@yahoo.com
+			

Figure 2 GUI of the DB



```
whitecliffes.db
SELECT * FROM users Schema Query Editor Auto Reload

TERMINAL PORTS DEBUG CONSOLE PROBLEMS

Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open whitecliffes.db
sqlite> SELECT * FROM teachers;
1|John Doe|john@example.com
2|Jane Smith|jane@example.com
sqlite> SELECT * FROM users;
1|Thomas King|thom@hotmail.com
3|John Shanks|jh@gmail.com
5|James Smith|jamesSmithy@mail.com
6|Martin Devlin|martyD@gmail.com
7|Tony Gun|tg@mail.com
8|Martha Douglas|marthaDoug@yahoo.com
9|Dan Brown|dannyboy@extra.co
10|Jon Buke|buke@mailman.com
11|Matilda Green|waltxingMatilda@gmail.com
12|Billy Slater|billy_boy@yahoo.com
sqlite> 
```

Figure 3 CLI of Whitecliffes.db file

The SQL database language is used to access data from the 'users' and 'teachers' tables in the 'whitecliffes. db' database file.

The command steps to access the database are:

1. sqlite3
2. .open whitecliffes.db
3. SELECT * FROM users; OR SELECT * FROM teachers;

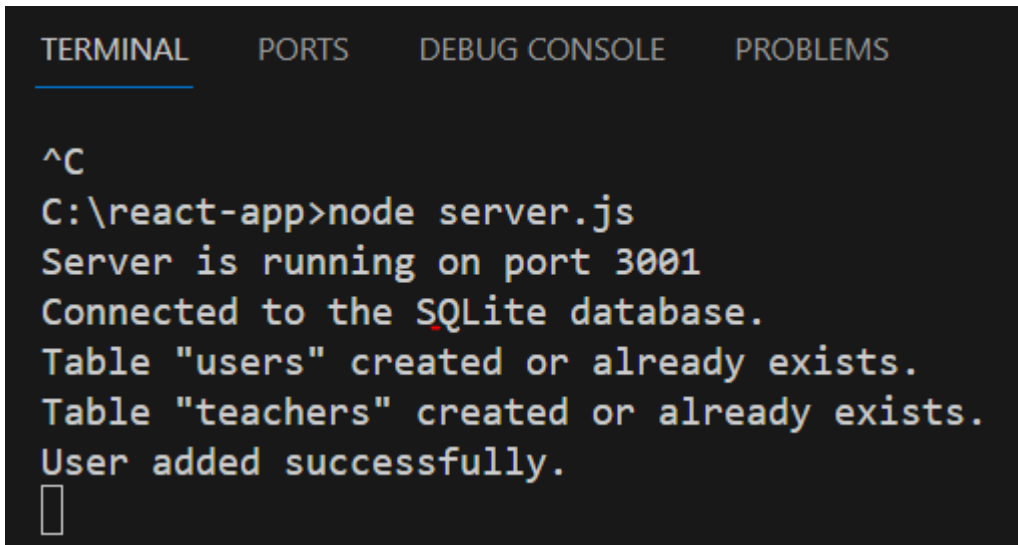
Node JS

Node JS was installed to run the server. It allows data to be captured from the web app and sent to the whitecliffes.db database. It also allows data to be fetched and inserted into the web app.

A server.js file was created to run both the server and to connect the app with the db. The file was run through the VSC command prompt terminal.

The command steps to run the file using Node JS through the terminal were:

1. node server.js



```
TERMINAL  PORTS  DEBUG CONSOLE  PROBLEMS

^C
C:\react-app>node server.js
Server is running on port 3001
Connected to the SQLite database.
Table "users" created or already exists.
Table "teachers" created or already exists.
User added successfully.
█
```

Figure 4 Running the server.js file through Node JS.

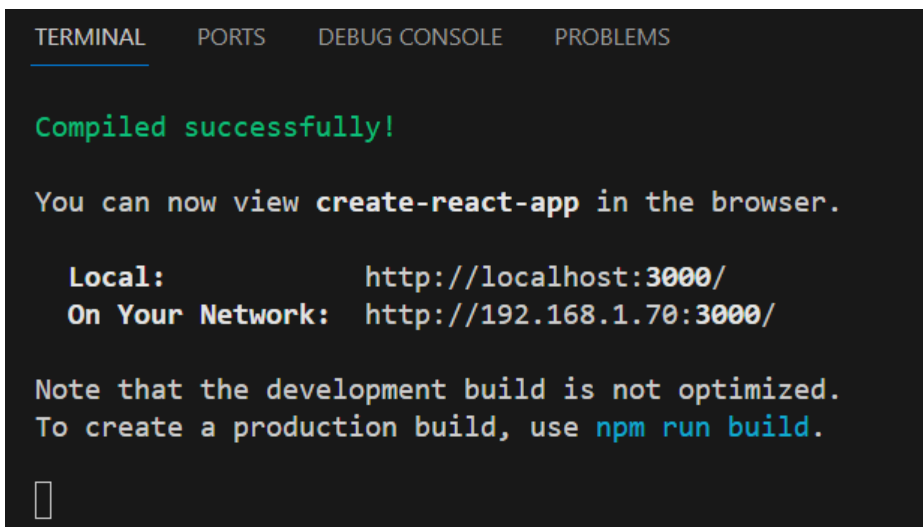
NPM

NPM was installed for dependency management and for running the app.

The app was run through the CLI terminal in VSC.

The command steps to run the app were:

1. npm start



```
TERMINAL  PORTS  DEBUG CONSOLE  PROBLEMS

Compiled successfully!

You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.1.70:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.

█
```

Figure 5 The app files were compiled successfully.

Local host

The app was built to run on the localhost and could be viewed at the following ports.

Local: <http://localhost:3000/>

On Your Network: <http://192.168.1.70:3000/>

Server

The server was set to run on port 3001 to ensure it did not interfere with the app running on port 3000.

The Application

The application was designed to be intuitive for users. Key registration is presented above the fold, and the app is intended to be device friendly.

The application is an SPA containing only the key content that young users require to register and learn about the club.

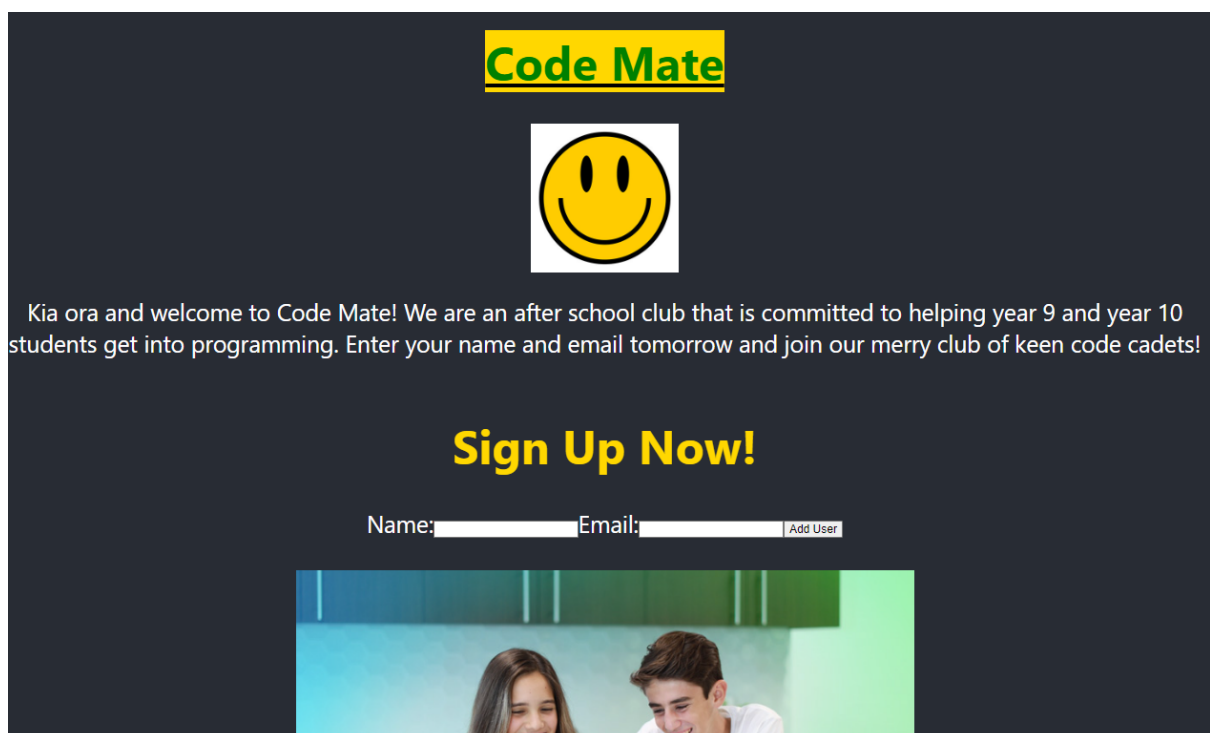


Figure 6 All key information is designed to be presented above the fold

Accessibility and Cultural Factors

The act is designed to be accessible to users with reading or visual conditions. The text is bold, clear, and simple text is used.

Te Reo Māori has been applied to create an inclusive experience for Māori users.