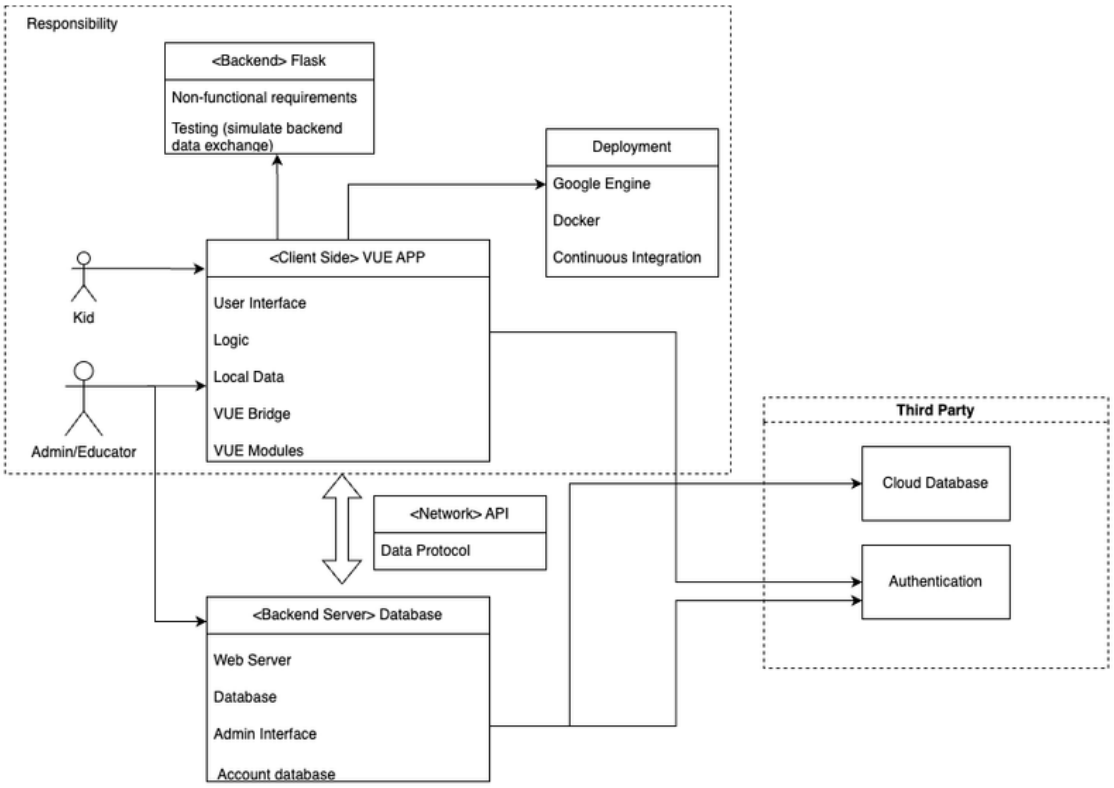
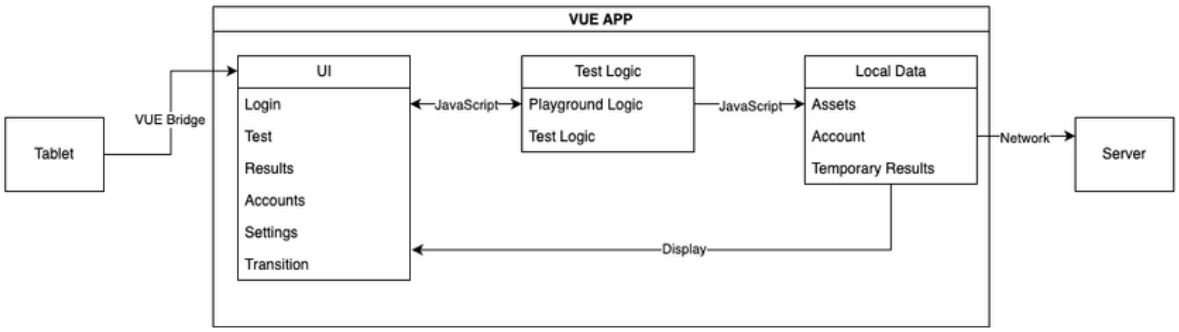
High-Level Architecture

Owned by Fiona Zhang •••
Last updated: Oct 27, 2023• See how many people viewed this page





Layer	Components	Function	Development Tools	Responsible
VUE App	User Interface	Login: schools login to their accounts. Test: presents visual cognition tasks. Results: summary of test results before sending them to the server (can be turned off)	Figma, IDE, VUE	☑

		Accounts: a page to select which student is taking the test. With sub pop ups such as adding Settings: a page for settings, such as turn off BGM. Transitions: intermediate page between different levels of the test.		
	Logic	Test logic: playground logic, test implementation.	VUE	☑
	Local Data	Accounts: class and kids info per school account. Assets: app assets. Temporary Results: to be displayed and/or sent to server.	VUE	☑
	VUE Bridge	Connect JavaScript with native modules (eg. change voice, touchscreen).	VUE	☑
	VUE Modules	Sending results to remote server.	VUE	☑
Backend Server	Web Server	API Endpoints: handle incoming data from the React Native app and save to the database. Authentication: ensure data is coming from an authenticated source.		☐
	Database	Test Results: store individual test results.		☐
	Admin Interface	Web-based interface for educators to view aggregated results.		☐
	Account database	School Profiles: store all classes, kids' information of the schools.		☐
Third-Party	Cloud Database Service	Storing and managing the application data.		☐
	Authentication Service	Ensuring data integrity and security.		☐
Deployment	dist		npm	☑
	Continuous Integration	Automating the build, test, and deployment process.	Git actions	☑
	Server	Host the website.	AWS, Heroku	☑

Network	HTTPS	Ensure data transmission between the app and the server is secure.		☑
Documentation	Motivational Model	Explains the conceptual framework.	Draw.io	☑
	Architecture Diagram	Explains the software modules.	Draw.io	☑
				☑

https://drive.google.com/file/d/1Ymd_UC27PZB8rb0gEovNNZsoO3VtZ1e9/view?usp=sharing

Connect Google to Atlassian to view more details of your work and collaborate from one place. Learn more about Smart Links.

Google Connect to Google

State machines to represent Client Side VUE App

In our application, the User Interface is a critical component responsible for delivering an intuitive and interactive experience to the users. To better demonstrate how the different pages interact with each other, we present a **state machine diagram**.

ℹ As our user interface will continuously evolve throughout our development, it will consequently have different pages, flows and transitions by the end of every sprint. We will be creating state machine diagrams illustrating our UI **by the end of every sprint**, to keep a record and more clearly observe our development progress. Hence, the final state machine created at the end of Sprint 3 will be the "official" state machine illustrating our user interface.

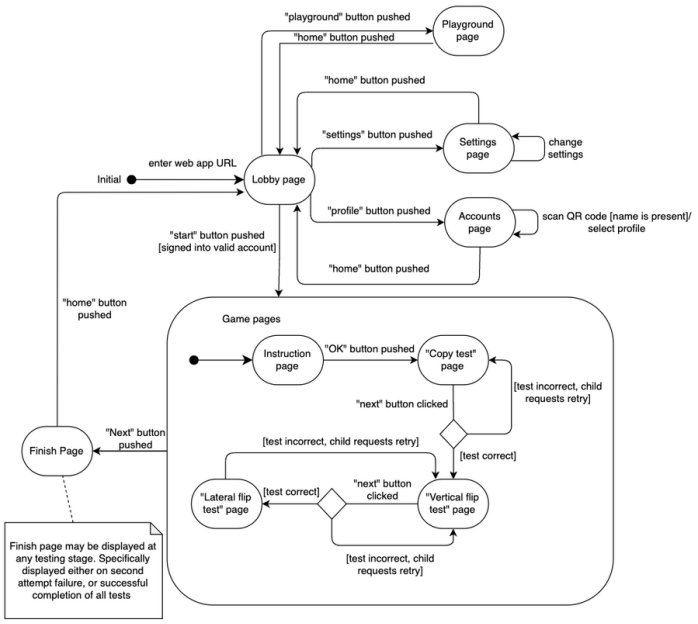
ℹ For reference:

States: Each node in the state machine represents a specific page within our UI.

Transitions: the lines and arrows between states depict the allowed transitions; specifically how users can move between pages based on their interactions. Where applicable, we have included conditions that determine the possibility of transitions between states.

Sprint 1 (Week 3 - 6)

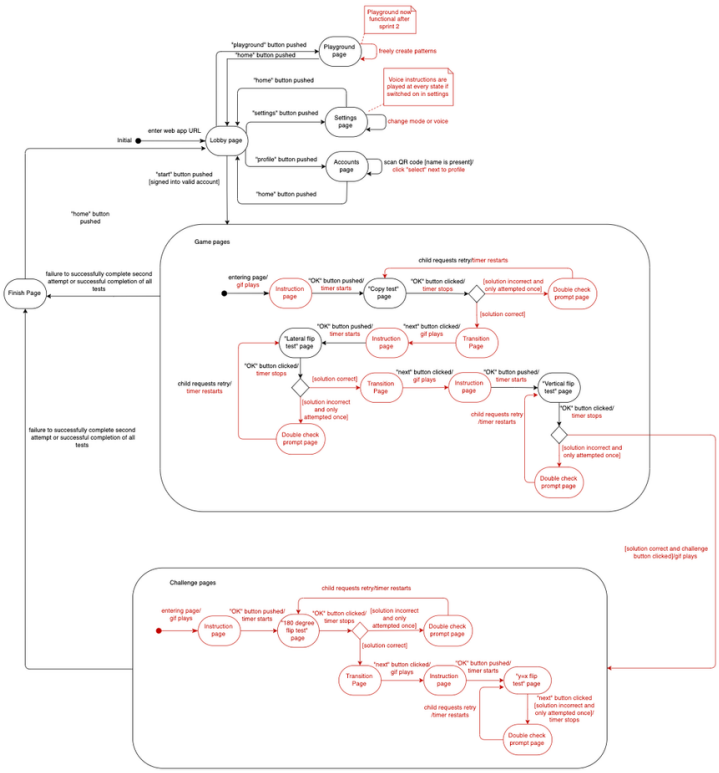
By the end of Sprint 1, we had created a basic interface which partially satisfied the client's requirements. Those using the application can already experience the the basic UI components such as lobby, settings, and accounts pages, as well as the testing components (where test scores are sent to a dummy backend) and their transitions. In addition, users are able to scan a QR code to load their information.



Sprint 2 (Week 6 - 9)

By the end of Sprint 2, we added more functionalities that make the application more well-rounded and relevant to the client's requirements. Key updates include:

1. Addition of instructional GIFs and double-checking prompts to aid the test taker's understanding
2. Timer on tests
3. Addition of voice instructions
4. Addition of challenge tests (not a core requirement but client approved)
5. Addition of a functional playground page
6. Refactored accounts page

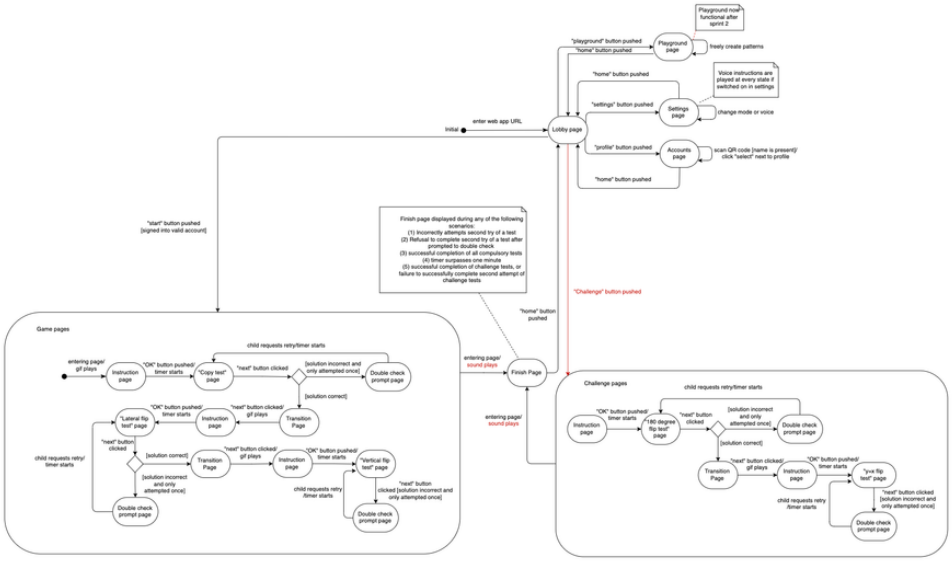


Content written in red represent the changes/additions we made to the UI since the end of the last sprint.

Sprint 3 (Week 9 - 12)

By the end of Sprint 2, we had already satisfied most, if not all of the business requirements requested by the client. Therefore, by the end of Sprint 3, there were no major changes to the features of our application, with most of our efforts directed towards user-based testing, deployment, the final presentation, and handover documentation. Changes to the UI are listed below:

1. Addition of a Challenge button in the lobby page, which directs the user to challenge tests. Users will no longer be directed to the challenge tests immediately after successfully completing the normal tests.
2. Addition of celebratory music when finish page is loaded
3. Updated appearance of sound button



Purpose of incorporation

The incorporation of these state machines into our documentation serves several key purposes:

1. Clarity and understanding: it provides a clear visual representation of how users can move between different pages within our application. This visual aid allows us to communicate complex UI behaviours effectively.
2. Maintenance and enhancements: When making enhancements to our UI at the start of each subsequent sprint, the state machine visualises how changes to the application may impact the overall user flow. This ensures that modifications are made such that a seamless user experience can be maintained.