8.0 Software Test Plan

8.1 Introduction

This section details the testing strategy for the Async project. It outlines the unit, integration, and acceptance test plans, describing the methodologies, procedures, and test configurations used to ensure the quality and functionality of the software. This plan aims to verify that the software performs as expected.

8.2 Unit Test Plan

This section describes the unit testing procedures using Vitest, Supabase' testing support, and React Developer Tools. Vitest presents itself as a convenient unit testing framework. The Async project already uses Vite as its build tool and Vitest is a simple add-on to the existing configuration and API in our project. React Developer Tools is a web extension that allows users to React components and inspect performance problems. Supabase integrates pgTAP in its CLI, which is a suite of database functions to help write unit tests for the database.

8.2.1 Unit Tests Planned

Unit tests will verify the correct functionality for the frontend and the backend, ensuring that they perform their intended operations. Tests will be performed and recorded in a document for future reference. Failed tests will be highlighted and marked as priority.

8.2.2 Unit Test Procedures

Test #	Module	Test Name	Inputs	Outputs
1	User Login	Valid	Valid	pass

		Login	email/password	
2	User Login	Invalid Login	Non-registered email/password	fail
3	DB Fetch	Data Returned	Valid query request	pass
4	DB Fetch	No data returned	Query with no matching data	pass
5	Component Render	Successful Render	Valid component props	pass
6	Component Render	Render error	Missing component props	fail

8.3 Integration Test Plan

This section details the integration testing process, which verifies the interaction and communication between integrated CSUs to form Component Software Configurations (CSCs). The integration tests will focus on ensuring seamless data flow and correct operation of the combined modules.

8.3.1 Integration Tests Planned

Integration testing will cover the following key interactions:

- User Profile: Ensures that login and logout function correctly works by verifying backend authentication and user data retrieval.
- Auth + Calendar: Ensures that after logging in, the system retrieves and displays the user's scheduled events from the calendar module.

- Auth + Discussion Board: Verifies that authenticated users can create discussion posts that are stored in the database and displayed to other users.

8.3.2 Integration Test Procedures

Test #	Module	Test Name	Inputs	Outputs
1	User Profile	Data Retrieval and Rendering	Valid user ID	pass
2	User Profile	Render Error	Non-registered User ID	fail
3	Auth + Calendar	Login + Fetch Events	Valid user credentials	Events for that user are retrieved and displayed in UI
4	Auth + Discussion Board	Create Discussion post	Valid user session, new post content	Post is created, visible to all in the class discussion

8.4 Acceptance Test Plan

8.4.1 Acceptance Tests Planned

The Acceptance Test Plan ensures that the final system meets all functional requirements from the end user's perspective

- User Registration & Login: User signs up, logs in, and is presented with their personal dashboard/calendar
- Calendar Event Management: User adds or edits events in the calendar and sees them persist

- Discussion Board Participation: User navigates to a class discussion board, creates/edits/deletes a post.
- Real-time Chat Usage: User sends a message in the chat box and it appears to other connected users.
- Error Handling & Logout: Invalid inputs produce clear error messages, and users can log out successfully.

8.4.2 Final Acceptance Test Procedures

A end-to-end scenario may include:

1. Launch Application: Confirm landing page loads with Register/Login options.
2. Register New User: Provide valid email/password; expect a successful account creation
3. Login: After registration, login to the system
4. Manage Calendar: Add Events , then verify the events are displayed
5. Discussion Board: Navigate to discussion board , create a post, and verify post is visible to user and any watchers.
6. Chatbox: Send text, confirm message is delivered in real-time.
7. Logout: Log out and confirm user session is terminated, returning to login screen.

8.5 Test Configuration Control

Tests will be maintained in a version-controlled repository alongside the source code. However, the tests will be stored in a separate folder, with subfolders for specific unit testing tools. As outlined, this project utilizes Vitest Supabase's CLI for unit testing and React developer tools for component testing and debugging. With this set-up, tests will be reliably reproduced across multiple machines.

8.6 Items Not Tested

Items that won't be individually include the front-end libraries used for design, routing, and validation. These libraries come with extensive support, so any error and rendering issues are clearly described in the developer tools interface. Additionally, our project is written in Typescript and any syntax and type errors are caught in compile time, which helps reduce the amount of debugging we need to.

8.7 Test Verification Matrix

Requirement #	Requirement Description	Test #
5.3.1	The web application shall first display the login main page for students and teachers.	1
5.3.2	The web application shall authenticate users and ensure that their information is secure.	1
5.3.5	The web application shall include a sidebar for key app features.	5
5.4.1	Re-Renders Should Be Minimized As Much As Possible	5
5.4.2	Responsive Design	5
5.4.5	Follow Data Fetching Best-Practices	3
5.4.6	Build a Robust Backend	3