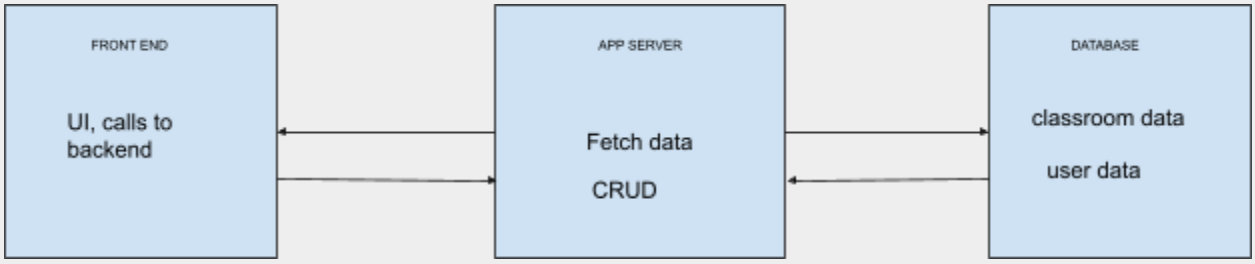
Requirements Specification Document

Async

5.1 Introduction

The design of this system will focus on creating a polished and feature-rich application in the education space. Brightspace, one of the primary learning management systems, has features that provide the bare minimum, but leaves much to be desired in regards to user experience. This system will focus on providing improved discussion board and chat features for both students and teachers. The following diagram presents the basic system-structure of this application:



5.2. CSCI Component Breakdown

CSCI *Async* is composed of the following CSCs:

5.2.1 Front-End
5.2.1.1 Client Main Screen
5.2.1.2 Client Discussion Board
5.2.1.3 Client Notes Upload
5.2.1.4 Client Chat Interface
5.2.1.5 Client-Side Routing Library
5.2.1.6 Client-State Management Library (optional)

5.2.2 Back-End
5.2.2.1 Server-State Management Library
5.2.2.2 Database Integration Library
5.2.2.3 Query Definition Module
5.2.2.4 Endpoint Structure Module

5.3 Functional Requirement

5.3.1 The web application shall first display the login main page for students and teachers.
5.3.2 The web application shall authenticate users and ensure that their information is secure.
5.3.3 The web application shall then display a dashboard of classes for students and teachers.
5.3.4 The web application shall utilize back-end authentication for users.
5.3.5 The web application shall include a navigation bar for key app features.
5.3.6 The web application shall allow users to upload notes for class.
5.3.7 The web application should allow users to chat with students and teachers.
5.3.8 The web application should allow CRUD operations related to specific features.

5.4 Performance Requirements

5.4.1 Re-Renders Should Be Minimized As Much As Possible

- Client State management will be a very important consideration as it relates directly to the performance of the React front-end.

5.4.2 Responsive Rendering

- The application should load smoothly and be responsive to user clicks and events. Client-Side Routing and Server-Side Rendering are potential tools to reach and optimize this goal.

5.4.3 Class Homepage Design Must Be Intuitive

- Users must be able to navigate their class pages easily and intuitively.

5.4.5 Follow Data Fetching Best-Practices

- Data that is fetched from the database or other external services should be handled procedurally with error handling and fallbacks.
- Loading States must be taken into consideration.

5.4.5 Build a Robust Backend

- A well-built backend will improve performance across the application and will lead to less errors during fetching.

5.5 Environment Requirements

5.5.1 Client Laptop or Desktop Computer:

5.5.2 Client OS: Windows, macOS, Linux

5.5.3 Client Minimum Processor: Intel Core i5 or equivalent

5.5.4 Client Minimum Storage: 256 GB

5.5.5 Client Minimum RAM: 8 GB

5.5.6 Developer Software Requirements: VSCode, NPM, Node, Fastify, Supabase, React, Vitest, and various libraries.