6.1 Introduction

This document presents the architecture and detailed design for the software for the Async project. This project will be built with Javascript across the board - React as the primary front-end library, and Node.js as the backend framework. Postgres will serve database needs, though this may evolve as the project develops.

6.1.1 System Objectives

The objective of this application is to provide a clean, user-friendly focused resource for professors and students or other professionals looking to inspire discussions online within their immediate circles. Users will be able to create accounts and view their discussion groups within a homepage dashboard, organized alphabetically or by usage. From there, they can easily navigate to specific discussion boards within these groups. User-to-User chat, search & filtering, and a rich text editor are planned as key features.

6.1.2 Hardware, Software, and Human Interfaces

6.1.2.1 Hardware Interfaces

6.1.2.1.1 Development Machine:

6.1.2.1.1.1 Operating System: Windows, macOS, Linux

6.1.2.1.1.2 Processor: Intel Core i5 or equivalent

6.1.2.1.1.3 Storage: 256 GB minimum

6.1.2.1.1.4 RAM: 8GB minimum

6.1.2.2.2 Testing Machine

6.1.2.2.2.1 Operating System: Windows, macOS, Linux

6.1.2.2.2.2 Processor: Intel Core i5 or equivalent

6.1.2.2.2.3 Storage: 256 GB minimum

6.1.2.2.2.4 RAM: 8GB minimum

6.1.2.2.2.5 Network Interface: Equipped with WiFi and ethernet capabilities to ensure stable network connections for testing real-time updates and database queries.

6.1.2.2 Software Interfaces

6.1.2.2.1 Text Editor / Integrated Development Environment:

6.1.2.2.1.1 Visual Studio Code (VSCode) - Version 1.93

6.1.2.2.2 Frontend Development:

6.1.2.2.2.1 React.js - Version 19

6.1.2.2.2.1.1 Installation: Vite

6.1.2.2.2.2 Tailwind CSS - Version 3.3+

6.1.2.2.2.2.1 Installation: npm install -D

tailwindcss postccss autoprefixer

6.1.2.2.3 Backend Development:

6.1.2.2.3.1 Node.js - Version 18

6.1.2.2.3.1.1 Installation: Node.js official site

6.1.2.2.3.2 Fastify (Node Framework)

6.1.2.2.3.2.1 Installation: Fastify official site

6.1.2.2.3.3 PostgreSQL - Version 14+

6.1.2.2.3.3.1 Installation: PostgreSQL official

Site

6.1.2.3 Human Interfaces

6.1.2.3.1 Login Interface

6.1.2.3.1.1 Description: The login interface provides secure access for all users. It features fields for username and password entry and supports third-party login options to simplify authentication.

6.1.2.3.1.2 Error Handling: User-friendly messages are displayed for failed login attempts (e.g., incorrect password or non-existent accounts)

6.1.2.3.2 Dashboard and Navigation Interface

6.1.2.3.2.1 Description: Upon logging in, users are greeted with a dashboard that serves as a central hub, displaying their discussion groups and relevant navigation options.

6.1.2.3.2.2 Features:

6.1.2.3.2.2.1 Discission Group Display: Users can see a list of all available discussion groups, organized by most recent, usage or alphabetical order, allowing quick access to their preferred discussions.

6.1.2.3.2.2.2 Navigation Bar: A persistent navigation bar provides access to key features, including profile management, discussion boards, and user settings, ensuring consistent and easy navigation.

6.1.2.3.3 Discussion Interface

6.1.2.3.3.1 Description: The discussion interface is the core of the Async platform, enabling users to read, create, and participate in discussions.

6.1.2.3.3.2 Features:

6.1.2.3.3.2.1 Create New Discussion: Users can initiate new discussions, by entering titles and content.

6.1.2.3.2.2.2 Commenting System: Users can add comments to discussions.

6.2 Architectural Design

The design of this system can be split into two sections: client-side, and back-end. For the client side, the Javascript library, React, will be used as the foundation for the design of the web application and enable it to be a dynamic application. Several tools that are well integrated with React will also be used. Tailwind CSS and shadcn are design and styling libraries that will allow us to better meet one of the core goals of this project, which is to craft a sleek interface. For data fetching needs, the use of custom hooks and the useEffect API are options.

The back-end can be independent of the client-side in terms of choice of framework, but for the sake of consistency, Javascript will also be used. Specifically, the Node.js framework Fastify. Node.js has the benefit of being well tested and maintained, and on top of that, Fastify provides better performance and developer experience.

Then, there is PostgreSQL for database needs. PostgreSQL is a tool that has been in use for many years, and is compatible with Node.js.

6.2.1 Major Software Components

6.2.1.1 The web application shall first display the login main page for students and teachers.

6.2.1.2 The web application shall then display a dashboard of classes for students and teachers.

6.2.1.3 The web application shall utilize back-end authentication for users.

6.2.1.3 The web application shall include a navigation bar for key app features.

6.2.1.3 The web application shall allow users to upload notes for class.

6.2.1.3 The web application should allow users to chat with students and teachers.

6.2.1.3 The web application should allow CRUD operations related to specific features.

6.2.2 Major Software Interactions

The primary software interactions will occur between the front end and back end. Because we are making a dynamic application, any changes in user information, or discussion board content, or class information will be properly updated in the backend as well. As mentioned in 6.2, Node.js frontend changes will be recorded and documented in PostgreSQL due to their compatibility.

6.2.3 Architectural Design Diagrams

6.2.4 Detailed CSC and CSU Descriptions

The Async website is a classroom discussion board platform for classrooms. Using Node.js, ReactNative, and PostgreSQL there are multiple Computer Software Components(CSCs) that are made up of Computer Software Units(CSUs).

6.2.4.1 User Interface CSC

The User Interface handles the interactive page of the website. This is what the users will be interacting with when they are using the website. It uses React Native and Tailwind CSS to create an intuitive and user friendly design.

6.2.4.1.1 User Interface CSUs:

6.2.4.1.1.1 Discussions List: All discussions ordered from the top being most recent to the bottom being the oldest.

6.2.4.1.1.2 Create New Discussion: Allows for creation of a new discussion, with title and text box.
6.2.4.1.1.3 Discussion: Features title and text attached.

6.2.4.2 Backend CSC

The Backend CSC will be handled by PostgreSQL. It will remember the discussion board text, title, author, timestamp, and other attributes to ensure data consistency and history.

6.2.4.2.1 Backend CSUs:

6.2.4.2.1.1 Discussion Object: Defines a *Discussion* object. It will feature attributes like title, author, time, etc.

6.2.4.3 Authentication CSC

The Authentication CSC is a layer of security for the website to ensure that the people that have access to the discussion boards are allowed to have access.

6.2.4.3.1 Authentication CSUs

6.2.4.3.1.1 Authentication CSU: Logins will be handled by Google to ensure classroom security.
6.2.4.3.1.2 Class Codes: Class codes will be provided by the administrator to verify identity.

6.2.5 Detailed Class Descriptions

The following are descriptions of the classes used in the Async application. These classes are in progress and are subject to change.

6.2.5.1 Discussion Class

The Discussion class represents a single discussion on the website
Fields:

- Id: Provides an identifier for the discussion thread.
- Title: Title of the discussion.
- Content: The text of the discussion.
- CreatedAt: Timestamp for discussion creation.
- Comments: Array of comments attached to the discussion.

6.2.5.2 Discussion Component

This component of Discussions displays the list of available

Discussions.
Fields:

- selectedDiscussion: Holds the currently selected discussion.
- Discussions: An array containing all discussions available.
- newDiscussionTitle: Creates a new discussion.
- newDiscussionContent: Creates new text for a discussion.
- newComment: Creates a new comment on a discussion.
- showForm: Shows or hides a discussion.

Methods:

- handleDiscussionClick(discussion): Selects discussion.
- handleAddDiscussion(): Allows for discussion to be viewed.
- handleSubmit(e): Processes submission for new discussion.
- handleAddComment(): Adds a comment.

6.2.5.3 Login Component

The login component is what allows users to login through a UI.
Methods:

- Login: Allows user to login using Username and Password.

6.2.6 Detailed Interface Descriptions Section

6.2.6.1 Login Interface

The login interface provides security to the website by authenticating user access through usernames and passwords
Inputs:

- Username: Entered by user. Processed by backend.
- Password: Entered by user. Processed by backend.

Outputs:

- Authentication token: If successful login, show user access has been granted.
- Error Message: In case of invalid access(incorrect username or password, or unauthorized) .

6.2.6.2 Discussions Interface

The Discussions interface is where the users will be able to view discussions.
Inputs:

- User Clicks: Selects discussions that the user wants to be displayed.
- Discussion Creator: Opens a text box where users can create a new discussion.
- Comments Creator: Opens a new text box where users can create a new comment to be attached to a discussion.

Outputs:

- Discussions List: A list of discussions on the left of the board.
- Discussion: The discussion selected by the user to be displayed.
- Discussion Comments: Comments attached by users to the selected discussion.

6.2.7 Detailed Data Structure Descriptions

6.2.7.1 Data Structures

The following are data structures associated with the Async website. There are User, Discussions, Comment, and Authentication data structures.

6.2.7.1.1 User Data Structure:

This contains the necessary information for a user to be stored in data.
Fields:

- userID: Unique id for each user.
- Username: Username for login for user.
- Password: Password for login for user.

6.2.7.1.2 Discussion Data Structure:

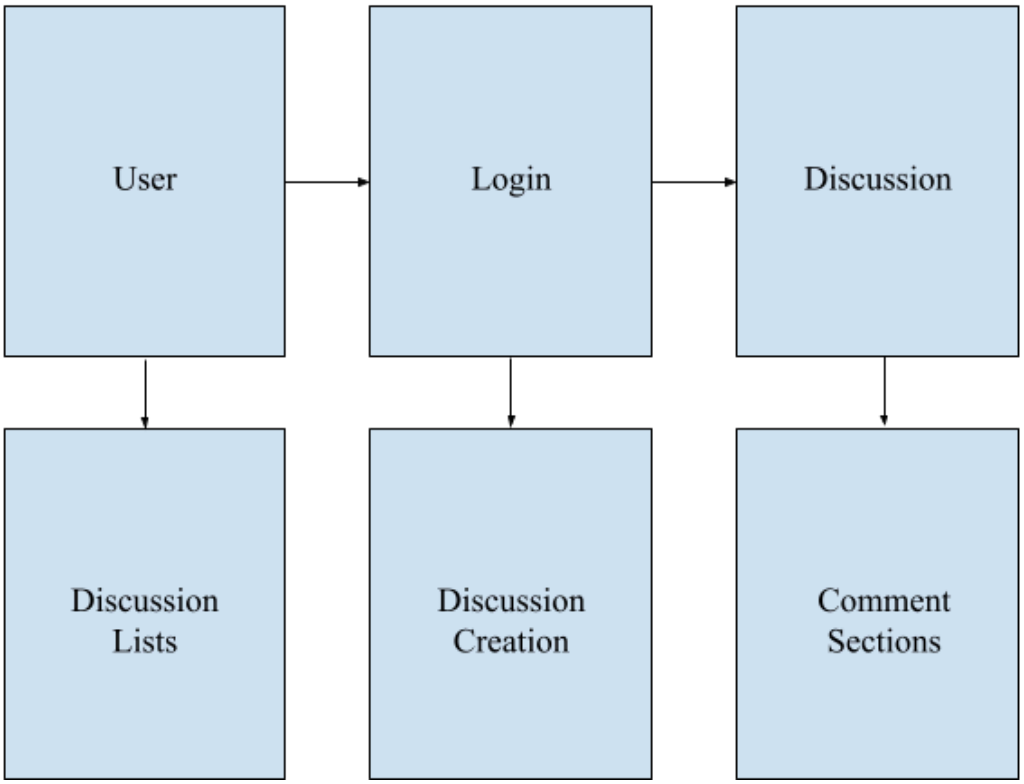
This contains the necessary information for discussion to be stored as data.
Fields:

- Id: Unique id for each discussion.
- Title: Title for each discussion.
- Content: Content for each discussion.
- Author: The author of each discussion.
- CreatedAt: A timestamp for the creation of the discussion.
- Comments: A list of comments that were attached to the discussion.

6.2.8 Detailed Design Diagrams Section

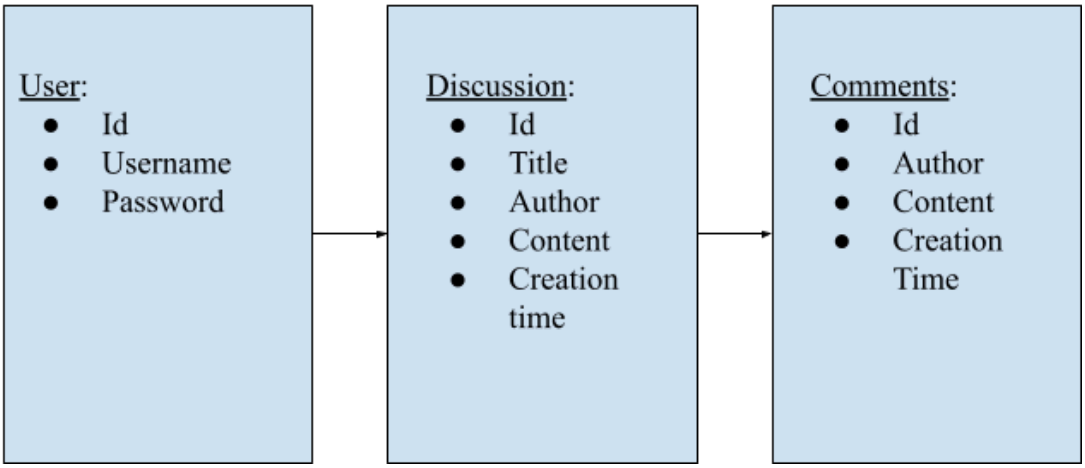
6.2.8.1 Front-End UML Diagram

UML Diagram (Front-End)

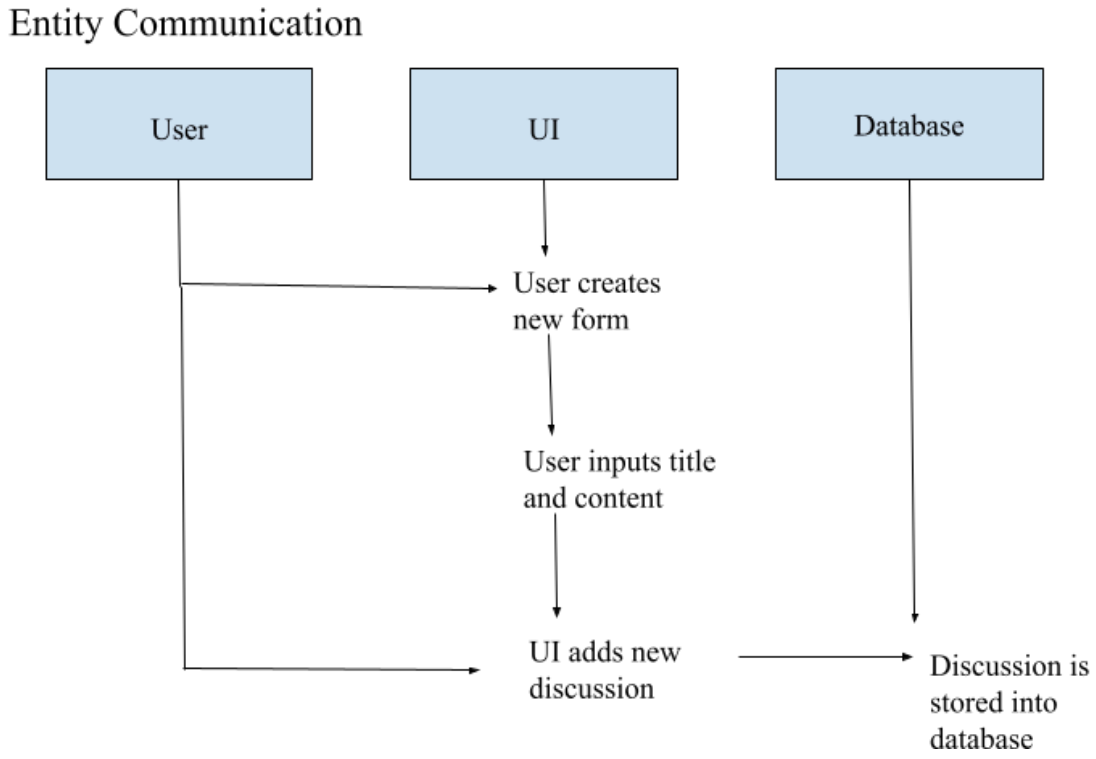


6.2.8.2 Entity Database Interactions

Entity Database Interactions



6.2.8.3 Entity Communication

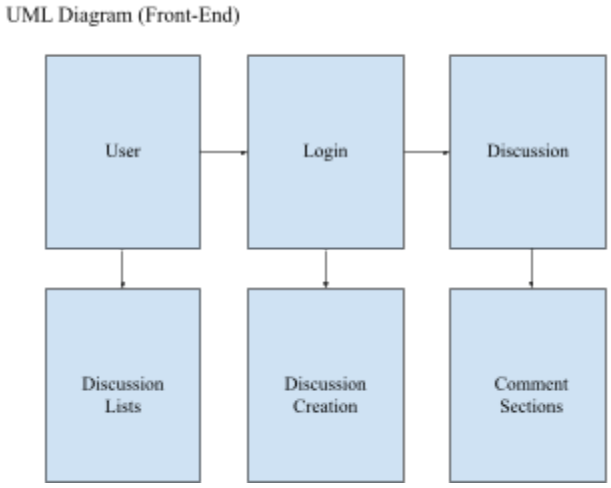


6.2.9 Database Design and Description Section

The database utilizes a postgreSQL database to store data.

6.2.9.1 Database Entity Relationship

The entity-relationship diagram illustrate the relationships between the the user, discussions, and comments.



Users:

- Users can login, see discussion lists(left side of screen), and a discussion.

Login:

- After a user logs in, they can create discussions and view discussions and comment sections.

Comments:

- The final entity viewable are comment sections, which requires a user to login, view the discussion lists, select a discussion, then view the comments added to the discussion.