System Design Document

CampusConnect

ULearn

Arina Azmi
Anusha Karkhanis
Miri Huang
Aarushi Doshi
Michael Walker

Table of Contents

Frontend

Class Name: Main.jsx	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities ● Imports necessities required to use React. ● Imports clerk publishable key from .env.local file. ● Allows React elements to render using createRoot().render(). ● Defines routing for all React pages using BrowserRouter.	Collaborators: ● React: Front-end framework ● ReactDOM: Allows react to render to DOM. ● React Router Dom (Route, Routes, BrowserRouter): for routing pages. ● Clerk-react (ClerkProvider): imports Clerk for authentication ● PersonalInfoPage: Page for personal info ● WhoAreYouPage: Page for gathering info for signup ● App: First page of website ● TutorPage: Page for gathering info for tutor signup. ● HomePage: Home page for logged-in users.

Class Name: App.jsx	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities ● Sends user to different pages depending on whether or not they're logged in. ● If not logged in, launch the clerk authenticator. ● If logged in, check if user is in the MongoDB database. If not, redirect to create account page (WhoAreYouPage). ● If logged in, redirect to HomePage.	Collaborators: ● React (useEffect): Hook to detect variable changes. ● React Router Dom(useNavigate): Function to redirect user. ● Clerk-react (SignedOut, useUser, RedirectToSignIn): Functions used for checking if user is signed in and for obtaining user info from Clerk. ● Axios: For making HTTP requests to the backend to save and retrieve user information.

Class Name: HomePage

Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Currently placeholder/setup for future sprint sprint ● It should hold a list of tutors available to teach courses in the user's university ● Showcase that the user has completed signing up (new user) or has logged in (existing user)	Collaborators: ● UserButton from @clerk/clerk-react: Manages user authentication controls and displays the user profile button.

Class Name: PersonalInfo	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Provide a form for users to input additional signup information, including languages they speak, university, and year of study. ● Validate the input data to ensure all required fields are filled correctly. ● Communicate with the backend to save or retrieve personal information. ● Upload profile picture	Collaborators: ● UserButton and useUser from @clerk/clerk-react: Manage user authentication and profile. ● Axios: For making HTTP requests to the backend to save or retrieve user information. ● PersonalInfoPage.css: Provides styling for the component. ● constants.jsx: Supplies constant values for university, year, and language options.

Class Name: Constants	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities ● maintains constants lists for PersonalInfo page such as list of languages, universities, and years ● Store, organize, and componentize our PersonalInfo page for further additions and updates	Collaborators: N/A

Class Name: WhoAreYou	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Display options for users to select their role (student or tutor). ● Manage and store the selected role in the component's state. ● Navigate the user to the appropriate page based on their selected role. ● Ensure that the user cannot proceed without selecting a role.	Collaborators: ● UserButton: Provides user authentication controls. ● whoAreYouPage.css: Provides styling for the component. ● student.png: Image asset for the student role option. ● tutor.png: Image asset for the tutor role option. ● React and clerk: To maintain signed in status and be able to sign out

Class Name: TutorInfo	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Take in user input for tutor-specific information including: selected courses, hourly rate, description. ● Allow tutors to upload a transcript to be verified, ensuring criteria is met for each selected course. ● Manage and store the inputted information in the component's state. ● Ensure that the user cannot proceed without entering all required information and being verified for all selected courses.	Collaborators: ● UserButton: Provides user authentication controls. ● tutorInfoPage.css: Provides styling for the component. ● tutorVerification.jsx: Verifies the credentials of the transcript uploaded by the tutor. ● Axios: to establish connection to MongoDB via POST request. ● Clerk: To maintain signed in status and be able to sign out. ● React: ReactToastify, Reactpdf-to-text, Reactselect

Backend

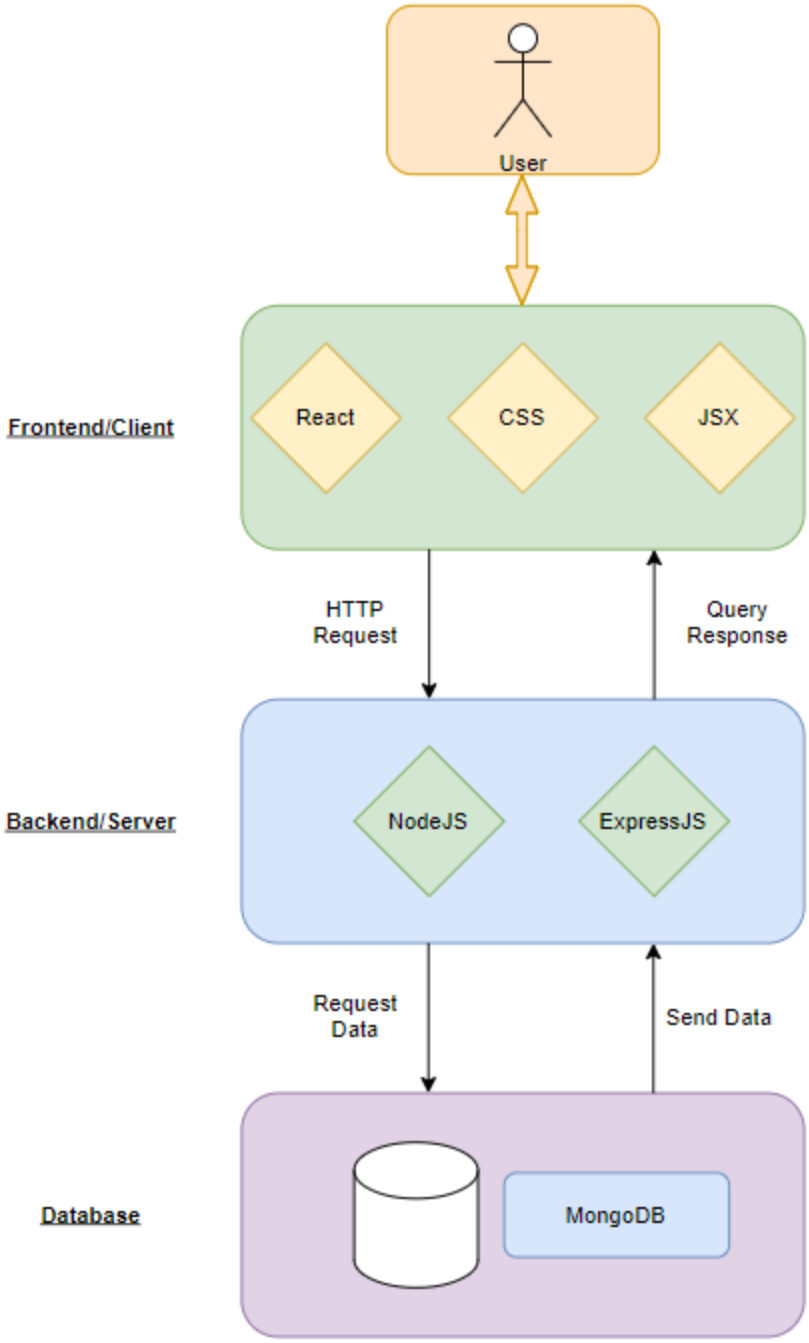
Class Name: UserModel (User.js)	
Parent Class(if any): N/A Subclass (if any): Tutor.js	
Responsibilities ● Define the schema for user data with the following fields: ○ clerkId: String ○ email: String ○ name: String ○ image: String ○ university: String ○ year: Number ○ languages: [String] ● Ensure data validation and enforce constraints (e.g., required fields, unique values).	Collaborators: ● mongoose: Provides the functionality to define schemas and interact with MongoDB.

Class Name: index.js	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities ● Set up and configure the Express application. ● Connect to the MongoDB database using Mongoose and connection URL. ● Use middleware for JSON parsing and CORS handling. ● Set up routes for various HTTP requests for interacting with the database. ● Start the database on port 3001	Collaborators: ● express: Express.js library. ● mongoose: MongoDB library ● cors: Handles Cross-Origin Resource Sharing to allow requests from different domains. ● UserModel: User model schema for MongoDB ● TutorModel: Tutor model schema for MongoDB ● app: Express application instance

Class Name: TutorModel (Tutor.js)	
Parent Class(if any): User.js Subclass (if any):	
Responsibilities	Collaborators:

● Define the schema for tutor-specific data with the following fields: ○ email (matching the email stored in their User profile) ○ verified courses they will teach ○ hourly rate ○ description ● Ensure data validation and enforce constraints (e.g., all required fields, hourly rate must be a number, can have multiple verified courses).	● mongoose: Provides the functionality to define schemas and interact with MongoDB.

System Architecture



The system uses 3 Tier Architecture, following the MERN stack.

The system is divided into frontend and backend components. The user interacts with the frontend, built with React. It handles user interactions and form submissions and sends HTTP requests to the backend, built with Node.js and Express.js.

The backend manages data storage and retrieval using and accessing MongoDB.

System Decomposition

There are 3 components to the system architecture, the frontend, backend, and database. The users interact with the frontend, developed using React and its components. The backend was developed with NodeJS and ExpressJS, Express allowed the system to handle HTTP requests to allow for communication with our database, MongoDB. There are currently distinct collections in the database for users and tutors. To deal with errors and exceptional cases, error handlers

were used to alert the system. In anticipation of receiving invalid inputs from users, input validation was coded in to prevent invalid database submissions. This also alerts the user of invalid inputs, and to correct them.