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 ● Holds a list of tutors available to teach courses in the user's language ● User can search tutors based on course codes ● Will also include a filter feature which should filter tutors by rating and price	Collaborators: ● UserButton from @clerk/clerk-react: Manages user authentication controls and displays the user profile button. ● useUser from @clerk/clerk-react: Provides user information and authentication status. ● axios: Used for making HTTP requests to fetch tutors' data. ● TutorCard from '../../components/TutorCard/tutorCard': Component used to display individual tutor information.

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 ● Display options for users to select their role (student or tutor), navigating the user to the appropriate page based on their selected role.	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	Collaborators: N/A

● Store, organize, and componentize our PersonalInfo page for further additions and updates	

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

Class Name: createCallButton.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Button component for creating calls. ● Ensures that only a tutor has access to the button to avoid potential student spamming (invisible to students). ● Creates and sends a link to a fresh video call in the chat and redirects the tutor who created the call to the video call screen.	Collaborators: ● useUser: object of currently logged in user from Clerk. ● useStreamVideoClient: video client for video calling by Stream IO. ● axios: used for HTTP requests ● Phone: Icon for button ● useEffect, useState: vanilla react hooks for checking and setting values ● useNavigate: used to send tutor to new video call page ● Toast: Used to display errors on the client side ● useChatContext: used to detect the current channel so the button can send the link to the correct chat.

Class Name: endCallButton.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Button accessible to video call creators (tutors) only. ● Removes everyone from the call and ends the call	Collaborators: ● useCall, useCallStateHooks: used to get current call information and the state of certain parts of the call, like the current call's creator id. ● React: react library to use the react framework ● useNavigate: used to redirect to homePage after clicking it

Class Name:MeetingRoom.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● The video call component that displays the video call in Meeting.jsx ● Allows participants to change the view (speaker left/right). ● Displays loading screen when user isn't fully joined yet. ● Sends the user to homePage when they leave the call.	Collaborators: ● CallControls, CallParticipantsList, CallStatsButton: components for pre-defined video calling functionalities. ● CallingState: Predefined values that are used for comparisons to detect the current call's state. SpeakerLayout: defines how the video call is displayed for a user. ● useCallStateHooks: used to get information of the current state of the call. ● useState: react hook for storing values ● Users: icon for participants list button. ● EndCallButton: custom component that ends the call for everyone. ● useNavigate: used for navigating between urls.

Class Name: MeetingSetup.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Component that allows user to adjust	Collaborators: ● DeviceSettings: Button to change mic

mic and camera before joining a call. ● Asks for permission if camera and mic were not previously enabled. ● Allows users to exit the call without joining. ● Checkbox for enabling/disabling both cam and mic before joining	and camera for call. ● VideoPreview: displays preview of call without actually being in the call yet. ● useCall: used to set mic and camera status for the actual video call. ● useNavigate: used for navigating urls.

Class Name: useGetCallById.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Custom hook that fetches a specific call based on call id. ● isCallLoading sends info if call still needs time to be fetched.	Collaborators: ● useStreamVideoClient: used for obtaining information stored about Stream IO for this application. ● useEffect, useState: hooks for storing values and for conditionals.

Class Name: Meeting.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Sets endpoint for video call rooms. ● Displays meeting setup or meeting room components depending on whether or not the user is set up. ● Sets pre-defined theming for Stream IO components.	Collaborators: ● useState, useEffect: react hooks for storing values and for conditionals. ● useParams: grabs url parameters ● useUser: object of current logged-in user from Clerk. ● StreamCall: component that allows other Stream IO video call components to be used. ● StreamTheme: default theming for Stream IO components ● MeetingSetup: component for setting up video call before joining a call. ● MeetingRoom: component for displaying video call ● useGetCallById: Used to get the call that the user is currently attempting to join. The Id is from the url.

Class Name: StreamVideoCallProvider.jsx	
Parent Class(if any): React.Component	

Subclass (if any): N/A	
Responsibilities ● Allows all users to access video calling using the Stream API. ● Generates tokens that allows access to Stream IO.	Collaborators: ● StreamVideo, StreamVideoClient: components used for functionality for connecting users to Stream IO. ● useEffect, useState: react hooks for storing values and for conditionals. ● useUser: the object of the current logged-in user from Clerk. ● Axios: used to make HTTP requests.

Class Name: Chatroom.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities ● Supports real-time chat messaging. ● Generates user tokens that allow access to chatrooms in Stream IO. ● Fetches user information for both the logged-in user and the tutor that was selected from the homepage. ● Creates a new chat channel with the 2 users if a channel between them does not already exist. ● Redirects users back to the homepage via navigation bar.	Collaborators: ● StreamChat: from Stream IO. Allows user token generation to create a new chat channel. Provides components such as Chat, MessageInput and ChannelList which provide comprehensive chat messaging features including image/file upload, read receipts, message reactions, etc. ● React - useEffect, useState: react hooks for storing values and for conditionals. ● useUser: the object of the current logged-in user from Clerk. ● Axios: used to make HTTP requests. ● Nav.jsx: allows for navigation to homepage from chat. ● chatroomLayout.css: handles styling for chatrooms.

Backend

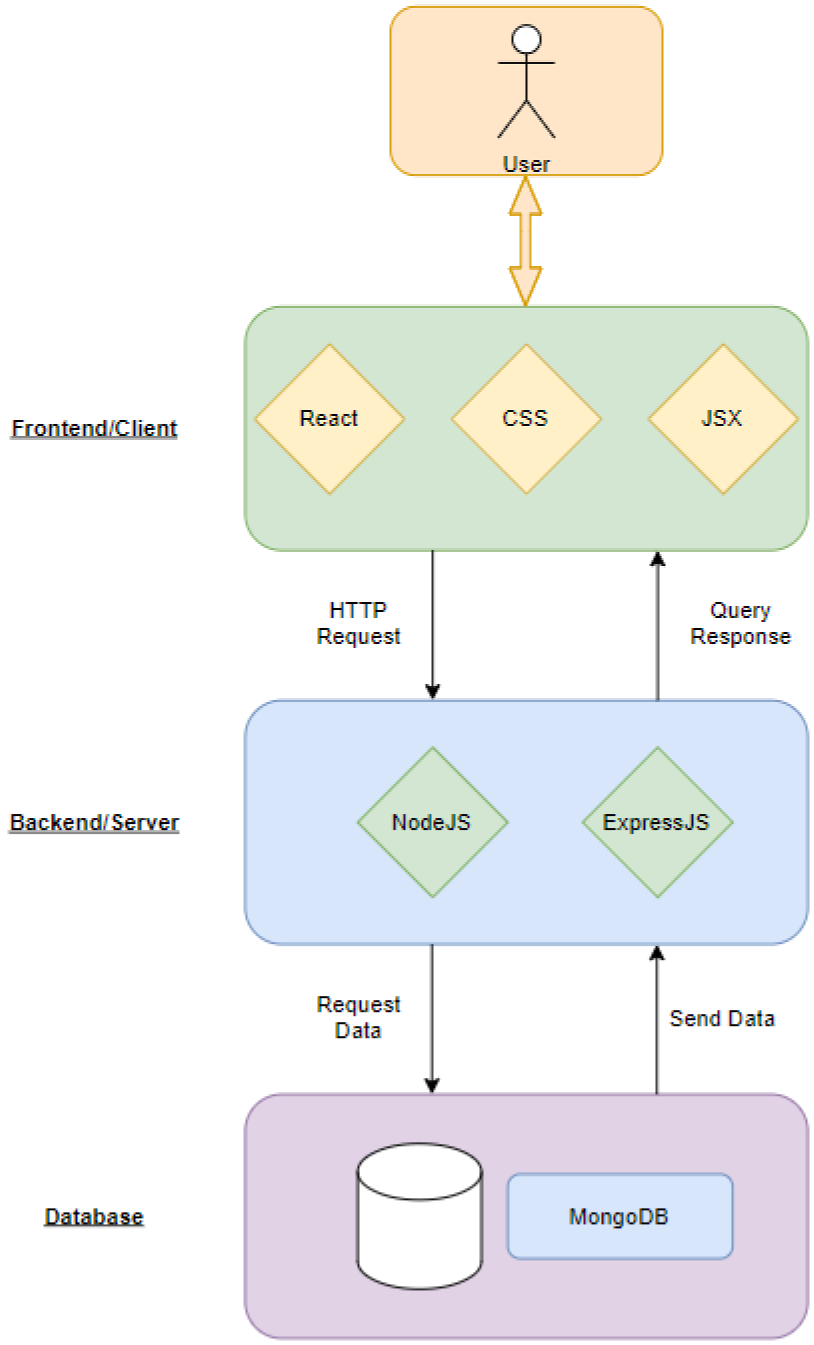
Class Name: UserModel (User.js)	
Parent Class(if any): N/A Subclass (if any): Tutor.js	
Responsibilities ● Define the schema for user data with the following fields: ○ clerkId: String ○ email: String ○ name: String ○ image: String ○ university: String ○ year: Number ○ languages: [String] ● Ensure data 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 ● Necessary api points for fetching, inserting, and updating data from mongodb	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



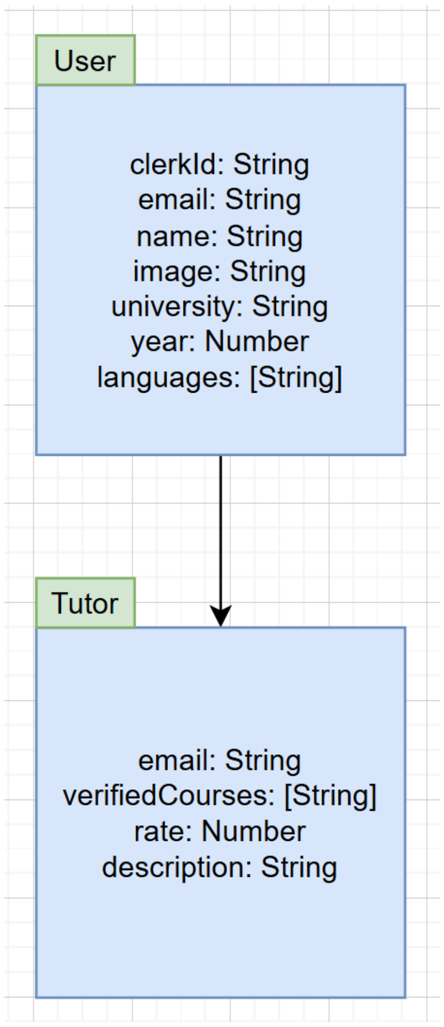
The system uses 3 Tier Architecture, following the MERN stack.

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.


User
clerkId: String
email: String
name: String
image: String
university: String
year: Number
languages: [String]
Tutor
email: String
verifiedCourses: [String]
rate: Number
description: String