

System Design Document

CampusConnect *ULearn*

Arina Azmi
Anusha Karkhanis
Miri Huang
Aarushi Doshi
Michael Walker

Table of Contents

Frontend..... 3

Backend.....9

System Architecture..... 11

Frontend

Class Name: Main.jsx	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none">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: <ul style="list-style-type: none">React: Front-end frameworkReactDOM: Allows react to render to DOM.React Router Dom (Route, Routes, BrowserRouter): for routing pages.Clerk-react (ClerkProvider): imports Clerk for authenticationPersonalInfoPage: Page for personal infoWhoAreYouPage: Page for gathering info for signupApp: First page of websiteTutorPage: 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 <ul style="list-style-type: none">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: <ul style="list-style-type: none">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 <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • maintains constants lists for PersonalInfo page such as list of languages, universities, and years 	Collaborators: N/A

<ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • Button accessible to video call creators (tutors) only. • Removes everyone from the call and ends the call 	Collaborators: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • Component that allows user to adjust 	Collaborators: <ul style="list-style-type: none"> • DeviceSettings: Button to change mic

<p>mic and camera before joining a call.</p> <ul style="list-style-type: none"> • 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 	<p>and camera for call.</p> <ul style="list-style-type: none"> • 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	
<p>Responsibilities</p> <ul style="list-style-type: none"> • Custom hook that fetches a specific call based on call id. • isCallLoading sends info if call still needs time to be fetched. 	<p>Collaborators:</p> <ul style="list-style-type: none"> • 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	
<p>Responsibilities</p> <ul style="list-style-type: none"> • 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. 	<p>Collaborators:</p> <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> Allows all users to access video calling using the Stream API. Generates tokens that allows access to Stream IO. 	Collaborators: <ul style="list-style-type: none"> 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 <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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.

Class Name: editTutorProfile.jsx	
Parent Class(if any): React.Component Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Displays hourly rate, verified courses, description, name and profile picture for logged-in user - retrieved from server Updates and stores user information 	Collaborators: <ul style="list-style-type: none"> React - useEffect, useState, useRef: react hooks for storing values and for conditionals. useUser: the object of the current logged-in user from Clerk, used this to

<p>when change is detected in an input field</p> <ul style="list-style-type: none"> Handles verification of newly selected courses, ensuring user's transcript contains the selected courses 	<p>search for tutor information in the database</p> <ul style="list-style-type: none"> Axios: used to make HTTP requests, handles retrieving data from MongoDB and writing newly inputted data to MongoDB. Nav.jsx: allows for navigation between homepage and edit profile page editTutorProfile.css: handles styling for the page. tutorVerification.jsx: Verifies the credentials of the transcript uploaded by the tutor.
---	---

Class Name: Feedback.jsx	
Parent Class(if any): React Subclass (if any): N/A	
<p>Responsibilities</p> <ul style="list-style-type: none"> Capture and submit user feedback and ratings for a tutor. Display a form for users to enter feedback and select a rating. Handle form submission and send the feedback data to the server. Provide options to close the feedback form 	<p>Collaborators:</p> <ul style="list-style-type: none"> React Hooks: used for storing values and managing submit Axios: Used to make HTTP POST requests to submit the feedback data to the server. React Router (useLocation, useNavigate): useLocation is used to get the current location and retrieve state data. useNavigate is used to programmatically navigate to different routes. Clerk (useUser): useUser retrieves the current logged-in user's information. Feedback.css: Handles the styling of the feedback form and its elements.

Class Name: reviewPost.jsx	
Parent Class(if any): React, tutorProfile Subclass (if any): N/A	
Responsibilities	<p>Collaborators:</p> <ul style="list-style-type: none"> React Hooks (useEffect, useState):

<ul style="list-style-type: none"> Fetch and display user reviews based on the provided email. Sort the reviews by date, displaying the most recent first. Render the reviews with user information, rating, and description. Handle error cases when fetching reviews from the server. 	<p>useEffect is used to fetch reviews when the component mounts or the email prop changes. useState is used to manage the state of the reviews.</p> <ul style="list-style-type: none"> Axios: Used to make HTTP GET requests to fetch reviews from the server. ReviewPosts.css: Handles the styling of the review posts and their elements.
---	---

Class Name: Recordings.jsx	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Displays the list of recordings that the current user has attended 	Collaborators: <ul style="list-style-type: none"> React: JavaScript framework RecordingList: List of recordings from Stream IO API for the current user. Nav: Navigation bar.

Class Name: RecordingList.jsx	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Obtains list of recordings from Stream IO API calls that the currently logged-in user has attended. Organizes the recordings into cards based on call details. Fetches calls using a custom hook and filters the recordings. 	Collaborators: <p>React: JavaScript framework. useState, useEffect: hooks for storing state and detecting changes. useGetRecordings: Custom hook that queries calls that the currently logged-in user has been a part of. RecordingCall: Single item in the recording list, displays recording info. useNavigate: navigation library for react.</p>

Class Name: RecordingCard.jsx	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Obtains info for a singular recording from RecordingList.jsx and displays the information in a styled card. 	Collaborators: <ul style="list-style-type: none"> React: JavaScript framework. lolosPlay, lolosCopy: Icons from react-icons/io library

Class Name: useGetRecordings.js	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Hook that obtains calls from Stream IO API that the user has been apart of. 	Collaborators: <ul style="list-style-type: none"> useUser: object of the currently logged in user from Clerk. useStreamVideoClient: client for Stream IO, used to obtain info for calls. useEffect, useState: hooks to manipulate state and detect change.

Class Name: RequestTutorButton.jsx	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Button to open RequestTutorCard component 	Collaborators: <ul style="list-style-type: none"> useState: Hook to change state RequestTutorCard: component with form to request tutoring.

Class Name: RequestTutorCard.jsx	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> A form to request service for tutoring of currently viewed tutor Obtains availabilities from availability schema Upon submission, two objects representing the appointment details are stored in the database (these can be viewed on the view booking requests page). 	Collaborators: <ul style="list-style-type: none"> useUser: object of the currently logged in user from Clerk. Axios: Used for making API requests useEffect, useState: hooks to manipulate state and detect change.

Class Name: Calendar.jsx	
Parent Class(if any): React Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> • Define and manage the calendar view for tutor availability. • Retrieve and display availability events for the specific tutor. • Allow tutors to add, move, resize, and delete events. • Ensure events are fetched from and updated in the backend. • Provide user feedback and data validation for event manipulation. 	Collaborators: <ul style="list-style-type: none"> • useUser: object of the currently logged in user from Clerk. • Axios: Used for making API requests • useEffect, useState: hooks to manipulate state and detect change. • useLocation: Retrieves tutor email and name from the homepage. • DayPilotCalendar: Displays the calendar and handles event interactions. • DayPilotNavigator: Provides date navigation for the calendar. • Nav: Navigation component included in the UI.

Backend

Class Name: UserModel (User.js)	
Parent Class(if any): N/A Subclass (if any): Tutor.js	
Responsibilities <ul style="list-style-type: none"> • Define the schema for user data with the following fields: <ul style="list-style-type: none"> ◦ 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • Define the schema for tutor-specific data with the following fields: <ul style="list-style-type: none"> ◦ 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). 	Collaborators: <ul style="list-style-type: none"> • mongoose: Provides the functionality to define schemas and interact with MongoDB.

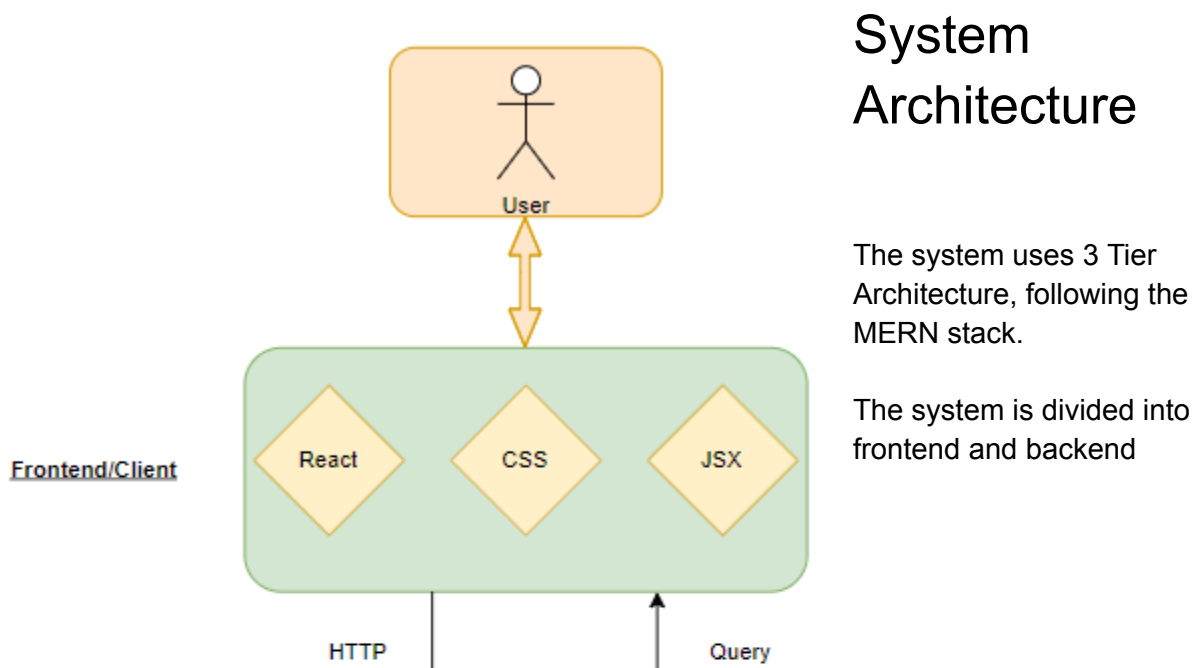
Class Name: ReviewModel (Review.js)	
Parent Class(if any): User.js Subclass (if any): N/A	

Responsibilities <ul style="list-style-type: none"> Define the schema for tutor reviews data with the following fields: <ul style="list-style-type: none"> tutorEmail: The email of the tutor being reviewed. starCountArray: An array to store star ratings. reviews: An array of reviews containing studentEmail, user (reference to User), rate, description, and createdAt Use Mongoose to provide schema definition and interaction with MongoDB. 	Collaborators: <ul style="list-style-type: none"> mongoose: Provides the functionality to define schemas and interact with MongoDB.
---	---

Class Name: AppointmentModel (Appointment.js)	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Define the schema for appointment data with the following fields: <ul style="list-style-type: none"> otherClerkId: String (required, unique) otherName: String (required) otherImage: String (required) topic: String (required) description: String startTime: Date (required) endTime: Date (required) Ensure data validation and enforce constraints (e.g., required fields, unique values). 	Collaborators: <ul style="list-style-type: none"> mongoose: Provides the functionality to define schemas and interact with MongoDB.

Class Name: AvailabilityModel (Availability.js)	
Parent Class(if any): N/A Subclass (if any): N/A	
Responsibilities <ul style="list-style-type: none"> Define the schema for availability data with the following fields: 	Collaborators: <ul style="list-style-type: none"> mongoose: Provides the functionality to define schemas and interact with

<ul style="list-style-type: none"> ○ tutorEmail: String (required, unique) ○ availableTimes: Array of objects containing: <ul style="list-style-type: none"> ○ eventId: String (required) ○ startTime: Date (required) ○ endTime: Date (required) ● Ensure data validation and enforce constraints (e.g., required fields, unique values). 	MongoDB.
---	----------



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.

