First page: **Vision statement** (Remove this text later)
ASSIGNED TO: Abdulkadir Ahmed

Student-HUB Vision Statement

Known as a pivotal time in life, university can be filled with challenges that extend beyond academics. Many students struggle to stay organized, connect with peers, and access the resources they need. StudentHUB aims to transform this experience by providing a centralized solution that meets students' academic, social, and organizational needs.

Student-HUB is a mobile application designed specifically for university students. The application offers an integrated suite of tools to help students stay organized, collaborate, and access support. A customizable academic calendar allows users to efficiently manage their course schedules and deadlines. In future iterations, this feature may include automation to extract deadlines directly from uploaded course outlines, saving students valuable time.

The second selling point for our app is the group chat formation feature which allows users to join course-specific chats by selecting courses they are enrolled in. Currently, students rely on offline socializing for peer support, which can be a hurdle for introverted students who suffer through a certain level of social anxiety. In those cases students tend to hesitate to reach out for help and sometimes even suffer in silence. In some classes, students are manually creating chats on Discord or WhatsApp and sharing through email or eclass private messages which is a very tedious and inefficient process on its own.

The third selling point would be the degree planning feature. It would consist of an AI chatbot capable of answering common academic advising questions regarding degree planning or tracking their current progress for EECS programs. This feature addresses the limited availability and long wait times of the EECS undergraduate advising office, especially during course enrollment periods. Often, students struggle to find relevant information on department websites due to poor navigation and lack of SEO, even though the answers to many common questions are already available. Additionally, it would include a dynamic degree planner tailored to a student’s current progress. It would recommend courses for the upcoming semester based on degree requirements, prerequisites of the courses and future courses that require the recommended courses. While both features are planned, we will prioritize implementing only one based on feasibility for this project.

This app aims to save students time by automating tedious and time-consuming tasks, allowing them to focus on their education and skill development instead of worrying about organization, collaboration, and degree planning.

Second page: **Big user stories** (Remove this text later)
ASSIGNED TO: Daksh Patel

View Course and Timeline Calendar

As a Student, I want to be able to check and update my calendar throughout my school journey for up-to-date information and timely reminders, for quality organization

Priority: High Cost: 10 days
Assigned: Abdul and Daksh

Access and Use Student Group Chat

As a Student, I want to be able to efficiently talk to other students in my courses and programs, to receive and give help, without having to manually ask everyone

Priority: High Cost: 10 days
Assigned: Jay and simon

Academic Advising Assistant

As a Student, I want to be able to ask questions regarding my current progress in the degree and receive recommendations regarding future semesters.

Priority: High Cost: 10 days
Assigned: Sandeepon and Fardad

Third page: **Iteration 1 detailed user stories** (Remove this text later)
ASSIGNED TO: Daksh Patel

Display Taken Courses Show all courses of students with room, professor and times for the current academic year Priority: High Cost: 4 days
Update Reminders and Calendar Show assignment and test due dates during the year with a clear and visual calendar Priority: Medium Cost: 5 days
Safe Login and Sign Up Safe and secure procedure for logging in and signing up for an account Priority: High Cost: 2 days
Communicate in Group Chat The ability to match-up with and communicate through texts with other students in same program and courses Priority: High Cost: 6 days
Building a chat front-end Develop a single chat front end. The input will be treated as prompts and the model's response will be displayed as answers. Priority: High Cost: 7 days Assigned: Sandeepon and Fardad
Update my Profile Be able to create and update your student profile for public and private usage Priority: Low Cost: 6 days

Big Story: Academic Advising Assistant **Assigned:** Sandeepon and Fardad

ITR1:

Priority:

1. Building a chat front-end
   Develop a single chat front end. The input will be treated as prompts and the model's response will be displayed as answers.

Planned:

1. Collecting academic advising information
   Communicate with the EECS department regarding where to find the information regarding the requirements of the EECS programs including course lists and pre-requisite information.
2. Converting information received in to knowledge base documents
   The information provided by the EECS department is required to be converted into documents that the AI model can use as it's knowledge base
3. Fine-tune the model
   Choose a functioning LLM model and define its knowledge base with the converted documents and ensure it is capable of answering questions from the documents.

ITR2:

Priority:

1. Memory Separation
   The access point to the AI model could potentially just be one from the app. Therefore, memory would have to be separated for each user either on the database level or within the AI model.

Planned:

1. Building a degree planner back-end
   A back-end which can analyze the student's current degree progress and requirements along with the course information to generate recommendations.
2. Extracting course information
   Use an LLM that can generate the course information from the converted documents.

ITR3:

Priority:

1. Building a degree planner front-end
   A front-end which will show recommendations for courses to take in the upcoming semester/s.

Planned:

1. iOS release requirements
   Meeting apple app store requirements for launch
2. Android release requirements
   Meeting Google Play store requirements for launch

Big Story: View Course and Timeline Calendar
Assigned: Abdulkadir and Daksh

ITR1: Building the Core Calendar Feature

Priority:

Building the Calendar UI

Develop a simple user interface where students can manually input their courses and deadlines. The calendar will display this information in a visually organized format.

This is essential for the foundation of the app's functionality.

Planned:

Creating the Calendar Interface

Use React Native and libraries like react-native-calendars to implement a calendar where users can interact with the interface and add course details. This will serve as the backbone for showing courses on the calendar.

Course Input Form

Implement a TextInput and Button form where students can add their courses and deadlines. The form will capture this data and display it on the calendar.

ITR2: Data Persistence and Integration

Priority:

Data Persistence

Ensure that the course data entered by students is saved even after the app is closed or restarted.

Planned:

Using AsyncStorage

A back-end which can analyze the student's current degree progress and requirements along with the course information to generate recommendations.

Data Retrieval

Implement functionality to retrieve the saved course data from AsyncStorage and display it again when the app is opened.

ITR3: Cross-Platform Compatibility and Testing

Priority:

Cross-Platform Calendar Access

Ensure the calendar is fully functional and responsive on both Android and iOS devices.

Planned:

Mobile Compatibility

Ensure that the calendar is mobile-friendly and works smoothly across different screen sizes (mobile and tablet).

Cross-Platform Testing

Test the calendar's features on both Android and iOS devices to ensure that the interface, data persistence, and functionality are working properly.

Big Story: View Course and Timeline Calendar

Tech Stack:

Development Platform: VSCode

1. **Back End: Node.js**
    a. **Why:** We use Node.js because it allows us to write server-side JavaScript, which simplifies the development process. Express.js helps set up routing and handle HTTP requests efficiently.
    b. **How it connects:** The backend (Node.js with Express) will handle API requests from the mobile app (React Native) for course data and other related information.
2. **Front End: React Native**
    a. **Why:** React Native enables building mobile apps using JavaScript, which helps create a cross-platform app for both iOS and Android with a single codebase.
    b. **How it connects:** React Native will render the CourseCalendar on mobile devices and fetch course-related data from the backend.
3. **Database: MongoDB**
    a. **Why:** MongoDB is ideal for storing flexible, semi-structured data like courses, deadlines, and user profiles.
    b. **How it connects:** MongoDB stores course data, and the backend (Node.js/Express) will interact with MongoDB to store and retrieve course and user data.

React Native has four options for development on PC:

a. React Native CLI
b. Expo CLI

Android Studio **and** xCode (for React Native CLI route)

Libraries:

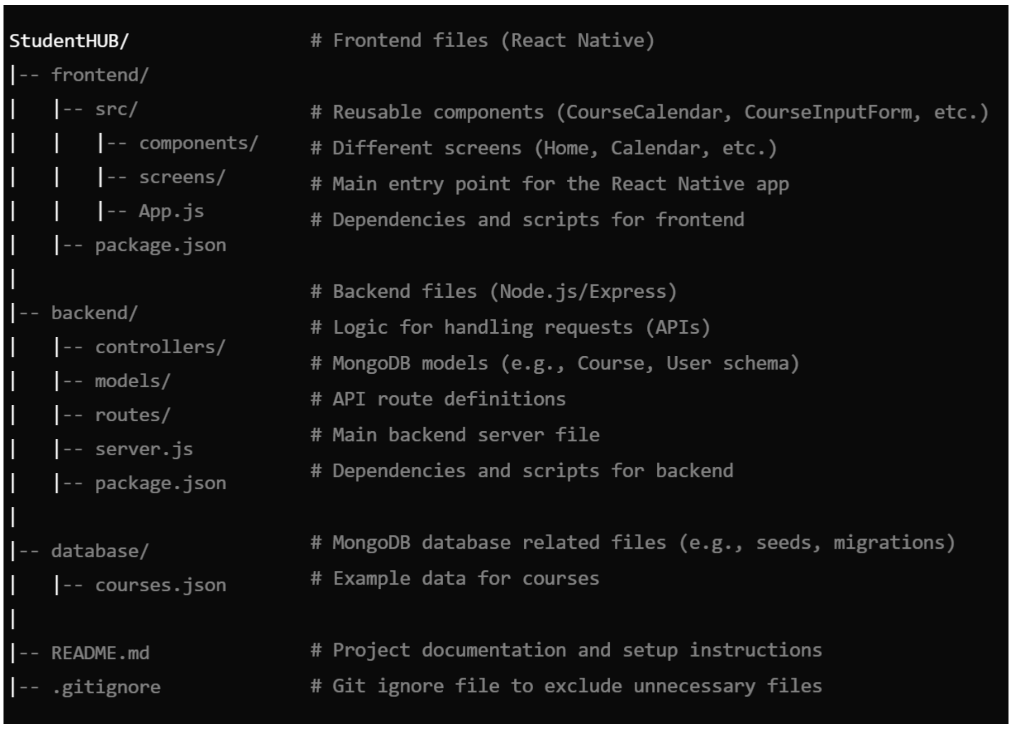
1. **Backend:**
    a. Node.js (runtime for server-side JavaScript)
    b. Express.js (web framework for handling APIs)
    c. MongoDB (for storing course and user data)
    d. Cors (to handle cross-origin requests)
    e. Body-parser (to parse incoming request bodies)

2. **Frontend:**
    a. React Native (for building cross-platform mobile apps)
    b. React Navigation (for app navigation)
    c. React Native Calendars (for the calendar UI)
    d. React Native AsyncStorage (for saving data locally on the device)

Testing and Deployment:

a. Render: For deploying the backend (Node.js + Express).
b. Expo: For developing and testing the React Native app.
c. Postman: For API testing and ensuring proper backend functionality.

Project Structure (On VS):

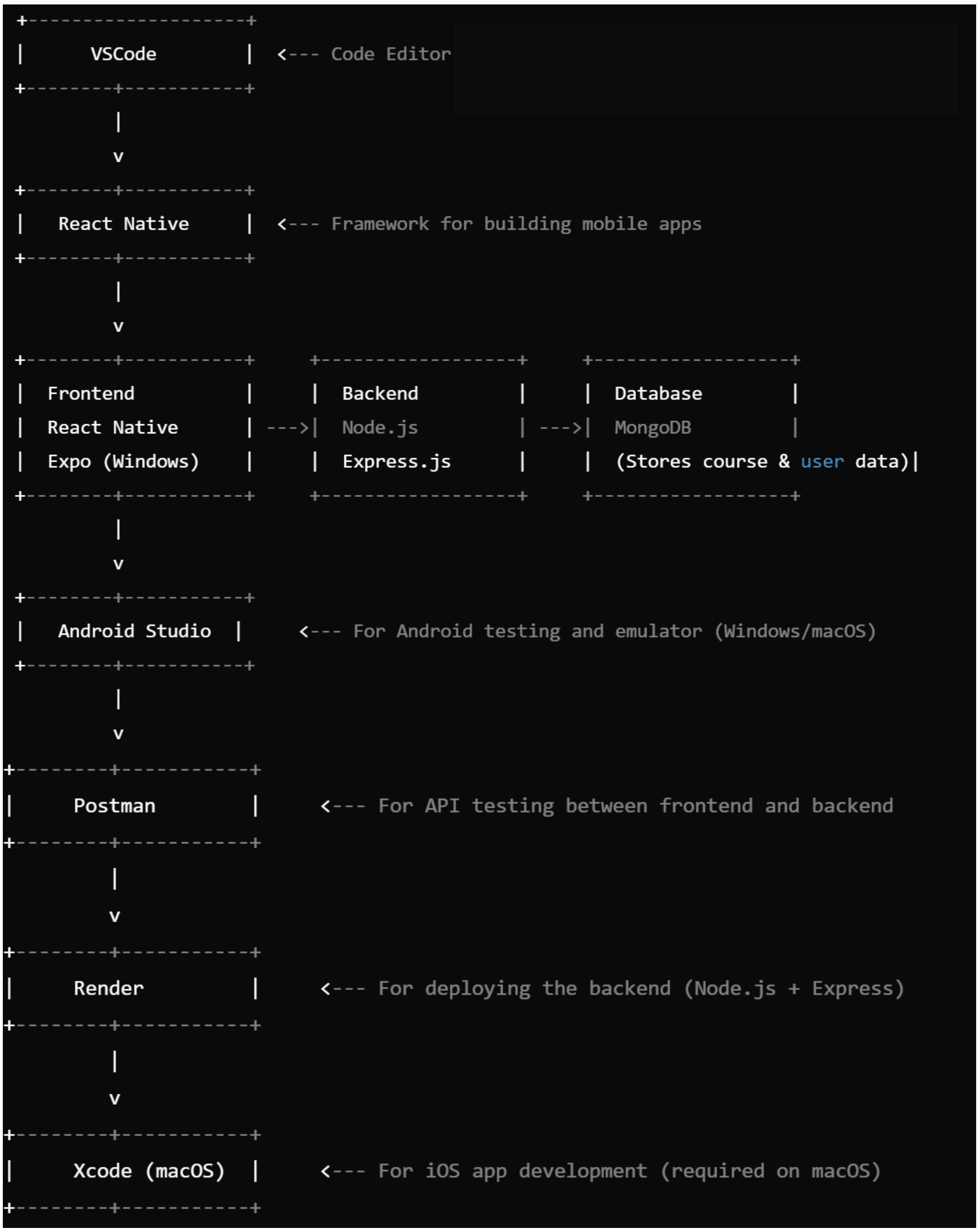


```
StudentHUB/                  # Frontend files (React Native)
|-- frontend/
|   |-- src/                 # Reusable components (CourseCalendar, CourseInputForm, etc.)
|   |   |-- components/      # Different screens (Home, Calendar, etc.)
|   |   |-- screens/         # Main entry point for the React Native app
|   |   |-- App.js           # Dependencies and scripts for frontend
|   |-- package.json
|                            # Backend files (Node.js/Express)
|-- backend/                 # Logic for handling requests (APIs)
|   |-- controllers/         # MongoDB models (e.g., Course, User schema)
|   |-- models/              # API route definitions
|   |-- routes/              # Main backend server file
|   |-- server.js            # Dependencies and scripts for backend
|   |-- package.json
|
|-- database/                # MongoDB database related files (e.g., seeds, migrations)
|   |-- courses.json         # Example data for courses
|
|-- README.md                # Project documentation and setup instructions
|-- .gitignore               # Git ignore file to exclude unnecessary files
```

How It All Connects:

1. Frontend (React Native) sends GET requests to the backend (Express) to retrieve the user's courses and other related data.
2. Backend (Node.js/Express) handles the request, retrieves the data from MongoDB, and sends it back to the frontend.
3. The React Native app displays the retrieved data (e.g., course names, deadlines) on the CourseCalendar screen for the user to view and interact with.

Project Structure Visual:



```
+-------------------+
|      VSCode       |  <--- Code Editor
+--------+----------+
         |
         v
+--------+----------+
|   React Native    |  <--- Framework for building mobile apps
+--------+----------+
         |
         v
+--------+----------+     +-----------------+     +------------------+
|  Frontend         |     |  Backend        |     |  Database        |
|  React Native     | --->|  Node.js        | --->|  MongoDB         |
|  Expo (Windows)   |     |  Express.js     |     |  (Stores course & user data)|
+--------+----------+     +-----------------+     +------------------+
         |
         v
+--------+----------+
|   Android Studio  |     <--- For Android testing and emulator (Windows/macOS)
+--------+----------+
         |
         v
+--------+----------+
|     Postman        |     <--- For API testing between frontend and backend
+--------+----------+
         |
         v
+--------+----------+
|     Render         |     <--- For deploying the backend (Node.js + Express)
+--------+----------+
         |
         v
+--------+----------+
|     Xcode (macOS)  |     <--- For iOS app development (required on macOS)
+--------+----------+
```