**Expense Tracker App**

**INTRODUCTION:**

Welcome to our expense tracker app! Our app is designed to help you easily track and manage your expenses, making it convenient for you to stay on top of your financial transactions. Whether you are a student, a working professional, or a business owner, our app is designed to meet your expense tracking needs.

With a user-friendly interface and intuitive design, our expense tracker app allows you to effortlessly record your expenses, categorize them, and analyze your spending patterns. You can easily add new transactions, specify the category, and add relevant details such as date, vendor, and payment method.

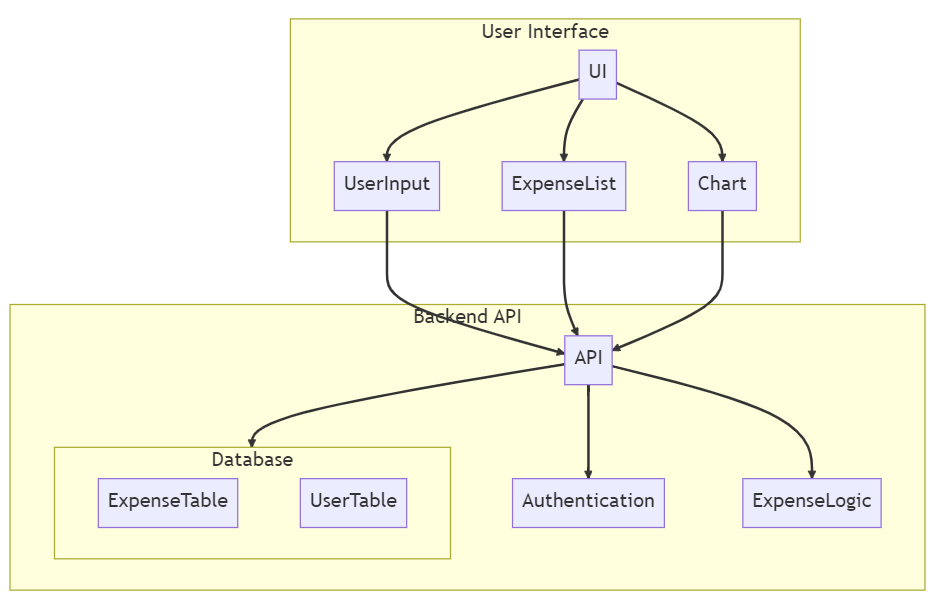
We prioritize user satisfaction and aim to provide a seamless and hassle-free experience for managing your finances. Our app enables you to set budgets, track your spending against those budgets, and receive notifications when you exceed your limits. You can also generate reports and visualize your expenses through graphs and charts for better financial insights.

For business users, our app offers robust backend functionalities. You can manage multiple accounts, track expenses for different projects or departments, and generate detailed expense reports for accounting purposes. Administrators can efficiently handle user inquiries, ensure data accuracy, and monitor the overall performance of the app.

With a strong focus on security and privacy, our expense tracker app ensures that your financial data remains protected. We employ encryption techniques to secure your information and provide password protection for your account. Your personal and financial information is kept confidential, and we strive to build trust with our users by providing a secure platform for managing expenses.

We are excited to have you on board and look forward to providing you with a reliable and user-friendly expense tracking experience. Start tracking your expenses with our app today and take control of your finances.

**Technical Architecture:**

****

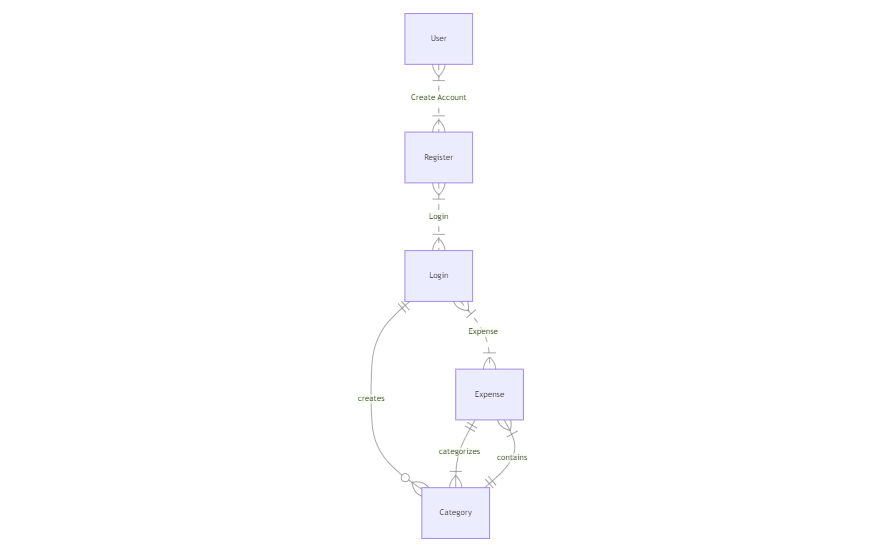
The technical architecture for an expense tracker typically consists of three main components: the User Interface, the Backend/API, and the Database.

**User Interface:** This component is responsible for providing a user-friendly interface for users to interact with the app. It includes features such as input forms for adding expenses, displaying expense lists, and generating charts or visualizations of expense data. The User Interface communicates with the Backend/API to send and receive data.

**Backend/API:** The Backend or API serves as the core logic and functionality of the expense tracker app. It handles business operations, data processing, and communication between the User Interface and the Database. The Backend/API includes modules for authentication, expense management, and any other necessary functionalities.

**Database:** The Database stores and manages the data for the expense tracker app. It includes tables or collections to store expense records, user information, and any other relevant data. The Database interacts with the Backend/API to store and retrieve data as needed.

**ER-DIAGRAM:**

****

**User entity:**

* user\_id (Primary Key): A unique identifier for each user.
* username: The username chosen by the user.
* email: The email address associated with the user.
* password: The password for the user's account.
* created\_at: The timestamp indicating when the user account was created.

**Expense entity:**

* expense\_id: A unique identifier for each expense.
* user\_id (Foreign Key): A reference to the user who made the expense.
* amount: The cost or amount spent for the expense.
* category: The category or type of expense (e.g., food, transportation, entertainment).
* date: The date when the expense was incurred.

The relationship between the "User" and "Expense" entities is represented by the user\_id attribute in the "Expense" entity, which serves as a foreign key referencing the primary key (user\_id) of the "User" entity. This indicates that each expense is associated with a specific user.

**Key Features:**

**Expense Tracking:** Our app allows you to easily track and monitor your expenses. You canrecord each expense, including the amount, category, date, and additional details. This feature helps you keep a comprehensive record of your spending habits.

**Add Expense Month wise**: With the month wise expense feature, you can organize your expenses based on specific months. This allows for better categorization and tracking of your expenses over time. You can view and manage expenses for each month separately.

**Add Earnings:** In addition to tracking expenses, our app also enables you to add and monitor your earnings. You can record the income you receive, such as salary, freelance payments, or any other sources of income. This feature helps you maintain a complete overview of your financial transactions.

**Delete Month:** If you no longer need to track expenses for a particular month, our app allows you to delete that month's data. This feature gives you the flexibility to manage and organize your expense records according to your needs.

**Export Feature:** The export feature enables you to generate reports and export your expense data in various formats. You can export your expense records to PDF, CSV, or Excel files, making it convenient for further analysis, sharing with others, or for record-keeping purposes.

**Adding Rows for Earnings and Expenses:** Our app provides a user-friendly interface where you can easily add rows for both earnings and expenses. This allows you to add multiple entries for each category, ensuring that you can accurately capture and track all your financial transactions.

By utilizing these key features, you can effectively track your expenses, categorize them monthwise, record your earnings, delete unnecessary data, export your records for analysis, and conveniently add multiple entries for both earnings and expenses. This comprehensive set of features empowers you to maintain a clear and organized financial record, making it easier to manage your personal or business finances

**PRE REQUISITES:**

To develop a full-stack expense tracker app using React js, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

* Download: <https://nodejs.org/en/download/>
* Installation instructions: <https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

* Download: <https://www.mongodb.com/try/download/community>
* Installation instructions: <https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

* Installation: Open your command prompt or terminal and run the following command: **npm install express**

**React**: React is a JavaScript library for building user interfaces. To create and manage your React project, you can use Create React App, a popular tool for bootstrapping React applications.

**Install Create React App:**

React offers a command-line tool called Create React App that simplifies project setup and development.

**To install Create React App globally, run the following command:**

npm install -g create-react-app

**Create a new React project:**

Choose or create a directory where you want to set up your React project.

Open your terminal or command prompt.

Navigate to the selected directory using the cd command.

**Create a new React project by running the following command:**

npx create-react-app my-react-app

Replace my-react-app with your preferred project name. Wait for the project to be created.

Navigate into the project directory:

After the project creation is complete, navigate into the project directory by running the following command:

cd my-react-app

**Start the development server:**

To launch the development server and see your React app in the browser, run the following command:

npm start

Create React App will compile your app and start the development server.

Open your web browser and navigate to http://localhost:3000 to see your React app running.

You have successfully set up React on your machine and created a new React project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for React. You can explore more advanced configurations and features by referring to the official React documentation: https://reactjs.org

**HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

**Front-end Framework:** Utilize React to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

* Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

* Visual Studio Code: Download from [https://code.visualstudio.com/download](%20https:/code.visualstudio.com/download%20)
* Sublime Text: Download from <https://www.sublimetext.com/download>
* WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

**To Connect the Database with Node JS go through the below provided link:**

• Link: [https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/](https://www.section.io/engineering-education/nodejs-%20mongoosejs-mongodb/)

**To run the existing grocery-web app project downloaded from github:**

**Follow below steps:**

**1. Clone the Repository:**

* Open your terminal or command prompt.
* Navigate to the directory where you want to store the grocery-webapp app.
* Execute the following command to clone the repository:

**git clone** <https://github.com/Bharath136/Grocery-Web-App-MERN>

**2. Install Dependencies:**

* Navigate into the cloned repository directory:

cd grocery-webapp

* Install the required dependencies by running the following command:

npm install

**3. Start the Development Server:**

* To start the development server, execute the following command:

npm run dev or npm run start

* The e-commerce app will be accessible at http://localhost:5100 by default. You can change the port configuration in the .env file if needed.

**4. Access the App:**

* Open your web browser and navigate to http://localhost:5100.
* You should see the grocery-webapp app's homepage, indicating that the installation and setup were successful.

**Video Tutorial Link to clone the project: -** <https://drive.google.com/file/d/1KTGK0XZj0XWOiDeNKJVRKQHXLyVWZYLM/view?usp=sharing>

**Project Repository Link:** <https://github.com/Bharath136/Grocery-Web-App-MERN>

Congratulations! You have successfully installed and set up the grocery-webapp app on your local machine. You can now proceed with further customization, development, and testing as needed.

**Roles and Responsibilities:**

**Account Creation and Setup:**

* Registering for an account by providing necessary information such as username, email, and password.
* Setting up and managing personal account details, including profile information.

**Expense Tracking:**

* Adding and managing expenses by entering details such as amount, category, and date.
* Categorizing expenses appropriately for better organization and analysis.
* Editing or deleting existing expenses when necessary.
* Providing accurate and up-to-date information regarding expenses.

**Budgeting and Goal Setting:**

* Setting personal budget limits and goals for different expense categories.
* Monitoring and tracking expenses against the defined budget.
* Receiving alerts or notifications when approaching or exceeding budget limits.

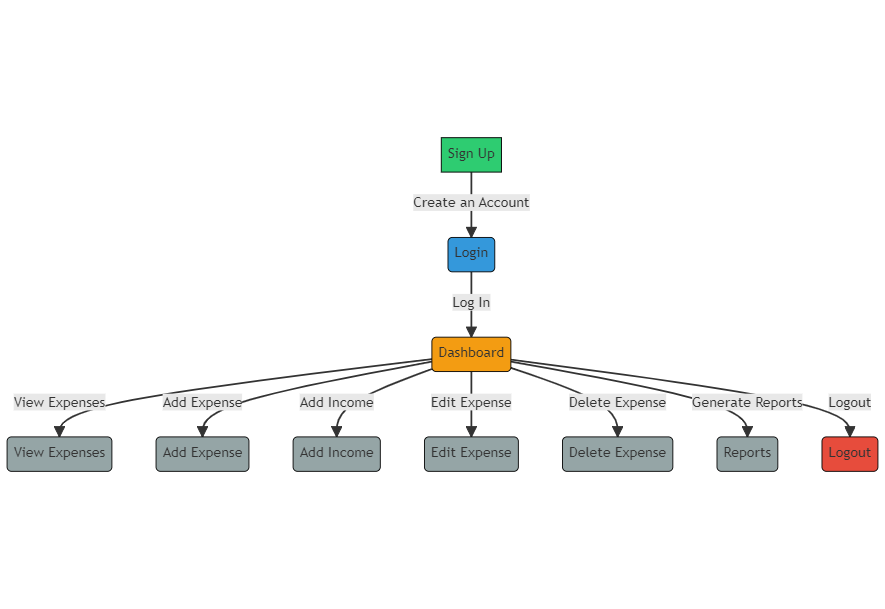
**Reporting and Analysis:**

* Accessing reports and summaries of expenses, including spending patterns and trends.
* Analyzing expenses based on different categories, time periods, or custom filters.
* Gaining insights into spending habits to make informed financial decisions.

**Data Security and Privacy:**

* Keeping login credentials secure and ensuring the confidentiality of personal financial information.
* Following best practices to protect personal data, such as using strong passwords and enabling two-factor authentication.
* Reporting any security concerns or suspicious activities to the app's support or security team.

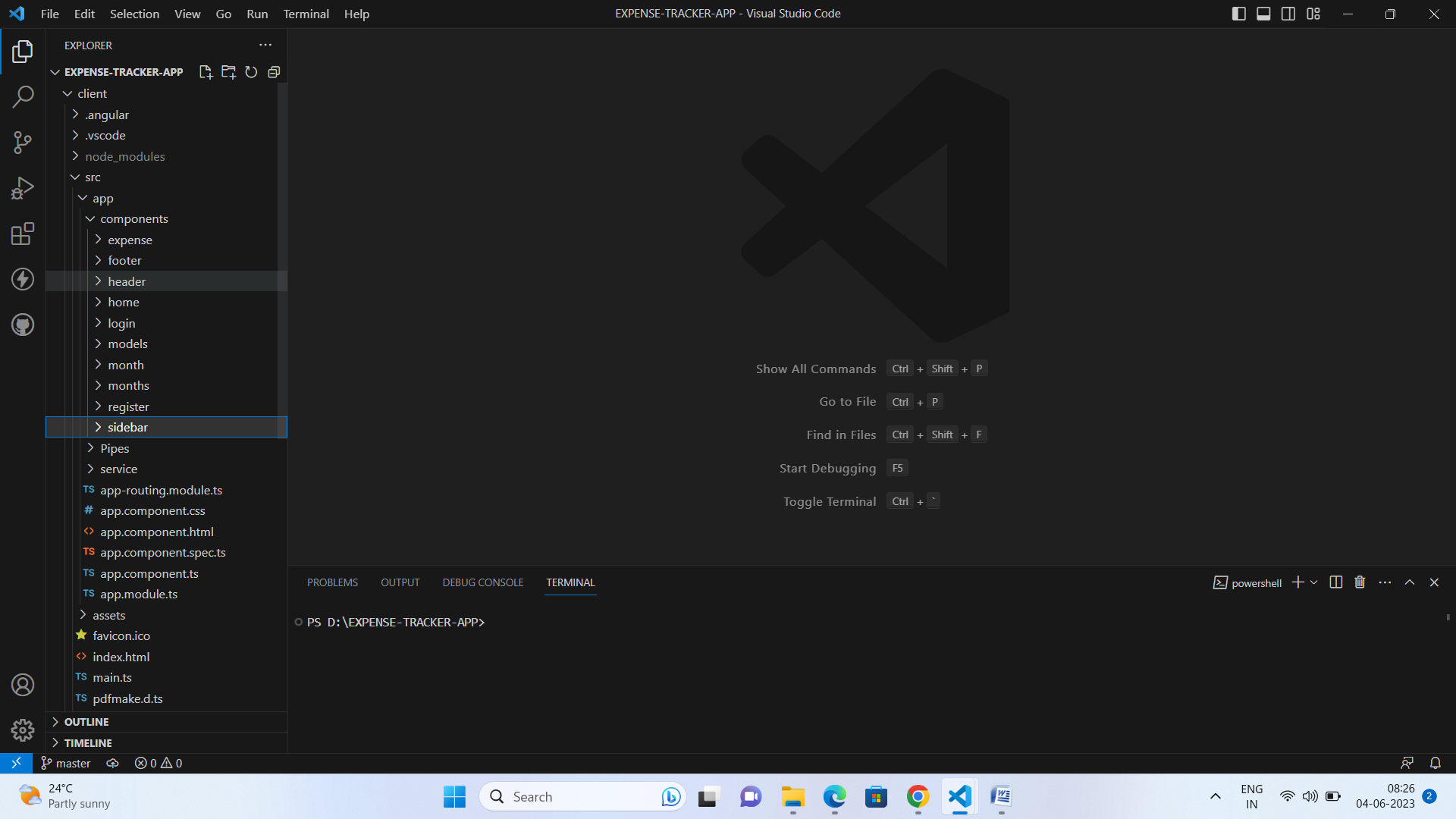
**User Flow:**

****

* User starts by signing up (A) and proceeds to register (B).
* After a successful registration, the user is directed to the login page (C).
* Once the user successfully logs in, they are directed to the dashboard (D).
* From the dashboard, the user can view their expenses in the expense list (E), add a new expense (F), view categories (G), or add a new category (H).
* The user can also logout, which will take them back to the login page (I).
* In the expense list, the user can edit an expense (F), or delete an expense, which prompts a confirmation (J). If the user confirms the deletion, they are redirected back to the expense list (E).
* Similarly, in the category list, the user can edit a category (H) or delete a category, which prompts a confirmation (K). If the user confirms the deletion, they are redirected back to the category list (G).

This user flow provides a basic outline of the steps a user may take while interacting with the Expense Tracker app based on the provided ER diagram. It's important to adapt and expand the user flow according to your specific app's functionality and requirements.

**PROJECT STRUCTURE:**

****

The project structure may vary depending on the specific framework, programming language, or development approach used. It's essential to organize the files and directories in a logical and consistent manner to improve code maintainability and collaboration among developers.

app/app.component.scss, src/app/app.component.spec.ts: These files are part of the main AppComponent, which serves as the root component for the Angular app. The component handles the overall layout and includes the router outlet for loading different components based on the current route.

**PROJECT FLOW:**

**Milestone 1: Project Setup and Configuration:**

**1. Install required tools and software:**

* Node.js.
* MongoDB.
* Angular CLI.

**2. Create project folders and files:**

* Client folders.
* Server folders.

**Milestone 2: Backend Development:**

**Setup express server:**

* Install express.
* Create app.js file.
* Define API’s

**Configure MongoDB:**

* Install Mongoose.
* Create database connection.
* Create Models.

**Implement API end points:**

* Implement CRUD operations.
* Test API endpoints.

**Milestone 3: Web Development:**

**1. Setup Angular Application:**

* Create Angular application using angular CLI.
* Configure Routing.
* Install required libraries.

**2. Design UI components:**

* Create Components.
* Implement layout and styling.
* Add navigation.

**3. Implement frontend logic:**

* Integration with API endpoints.
* Implement data binding.

**Create database in cloud video link:-** <https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view?usp=sharing>

**To Setup the frontend development and to connect node.js with MongoDB Database Go through this video link: -**

<https://drive.google.com/file/d/1b5bMvnqmASXLnSZ74B2t3EzNjuWHj63g/view?usp=drive_link>

**Backend:**

**1. Set Up Project Structure:**

* Create a new directory for your project and set up a package.json file using npm init command.
* Install necessary dependencies such as Express.js, Mongoose, and other required packages.

**2. Database Configuration:**

* Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
* Create a database and define the necessary collections for hotels, users, bookings, and other relevant data.

**3. Create Express.js Server:**

* Set up an Express.js server to handle HTTP requests and serve API endpoints.
* Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

**4. Define API Routes:**

* Create separate route files for different API functionalities such as hotels, users, bookings, and authentication.
* Define the necessary routes for listing hotels, handling user registration and login, managing bookings, etc.
* Implement route handlers using Express.js to handle requests and interact with the database.

**5. Implement Data Models:**

* Define Mongoose schemas for the different data entities like hotels, users, and bookings.
* Create corresponding Mongoose models to interact with the MongoDB database.
* Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

**API Design and Development:**

* Identify the necessary functionality and data required by the frontend.
* Design a set of RESTful APIs using a framework like Express.js or Django REST Framework.
* Define API endpoints for user management, product catalog, shopping cart, order management, payment gateway integration, shipping integration, etc.
* Implement the API routes, controllers, and data models to handle the corresponding operations.
* Ensure that the APIs follow best practices, are secure, and provide appropriate responses.

**User Management and Authentication:**

* Implement user registration and login functionality.
* Choose an authentication mechanism like session-based authentication or token-based authentication (e.g., JWT).
* Store and hash user credentials securely.
* Implement middleware to authenticate API requests and authorize access to protected routes.

**Add Expense:**

The "Add Expense" functionality in the backend involves receiving a request with the necessary information for the expense, such as the amount, category, date, and any additional details. The backend server would then process this request, validate the data, and save the expense to the database, associating it with the user who added it. This could involve updating the appropriate tables or collections in the database and performing any necessary calculations or aggregations.

**Add Income:**

Similar to adding an expense, the "Add Income" functionality involves receiving a request with the income details, validating the data, and saving it to the database. The income may have properties such as the amount, source, date, and any additional information. The backend would handle the request, update the relevant data in the database, and ensure it's associated with the user.

**Delete Month:**

Deleting a month in the backend would typically involve removing all expenses and income records associated with that month for a specific user. This action could be triggered by a user request or based on specific business rules. The backend would handle the request, locate and delete the corresponding data from the database, and update any relevant calculations or aggregations affected by the removal.

**Download Report:**

The "Download Report" functionality allows users to generate and download a report summarizing their expenses, income, or other relevant financial information. The backend would receive the request, retrieve the required data from the database, and generate a report in a specific format (e.g., PDF, CSV, or Excel). The generated report would then be sent back to the user as a downloadable file.

**Add Row:**

The "Add Row" functionality typically refers to the ability to add additional rows or entries within a specific expense or income record. For example, if a user wants to add multiple items or details to an expense, they can add new rows to accommodate this information. The backend would handle the request, update the corresponding record in the database, and ensure data integrity and consistency.

**Delete Row:**

The "Delete Row" functionality allows users to remove a specific row or entry within an expense or income record. This could be useful if a user wants to remove a specific item or detail from a larger expense. The backend would process the request, locate the corresponding row in the database, delete it, and update any relevant calculations or aggregations affected by the removal.

These functionalities involve interacting with a backend server, handling requests, validating and processing data, and performing operations on the database. The exact implementation details may vary depending on the specific backend technology stack and database system you are using.

**Schemas Usecase:**

**User:**

Schema: userSchema

Model: 'User'

Purpose: Represents the schema and model for user data. It includes fields such as firstname, lastname, username, email, and password. This schema is used to store user details for authentication and personalization within the Expense Tracker app.

**Expenses:**

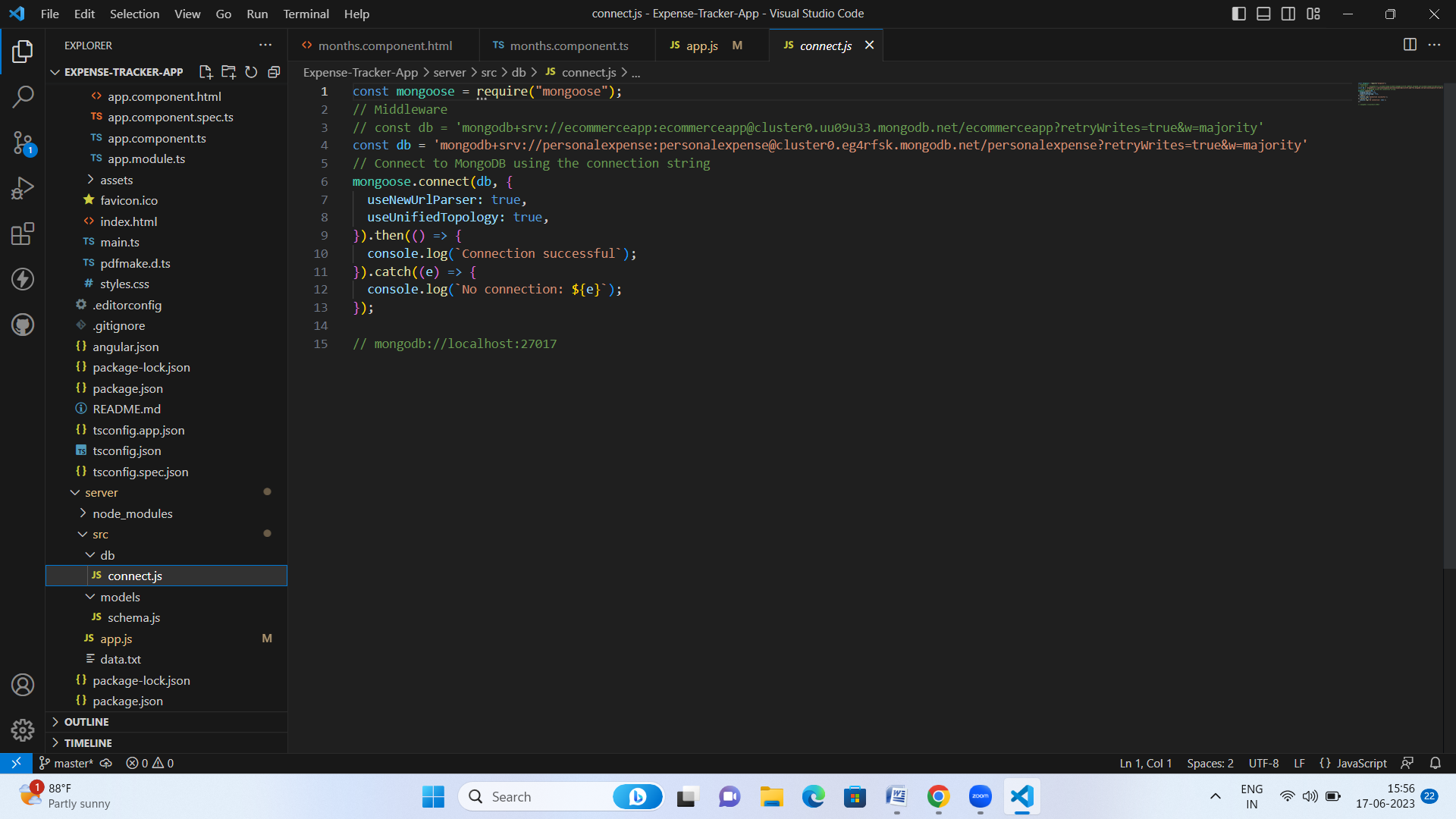
Schema: expenseSchema

Model: 'Expense'

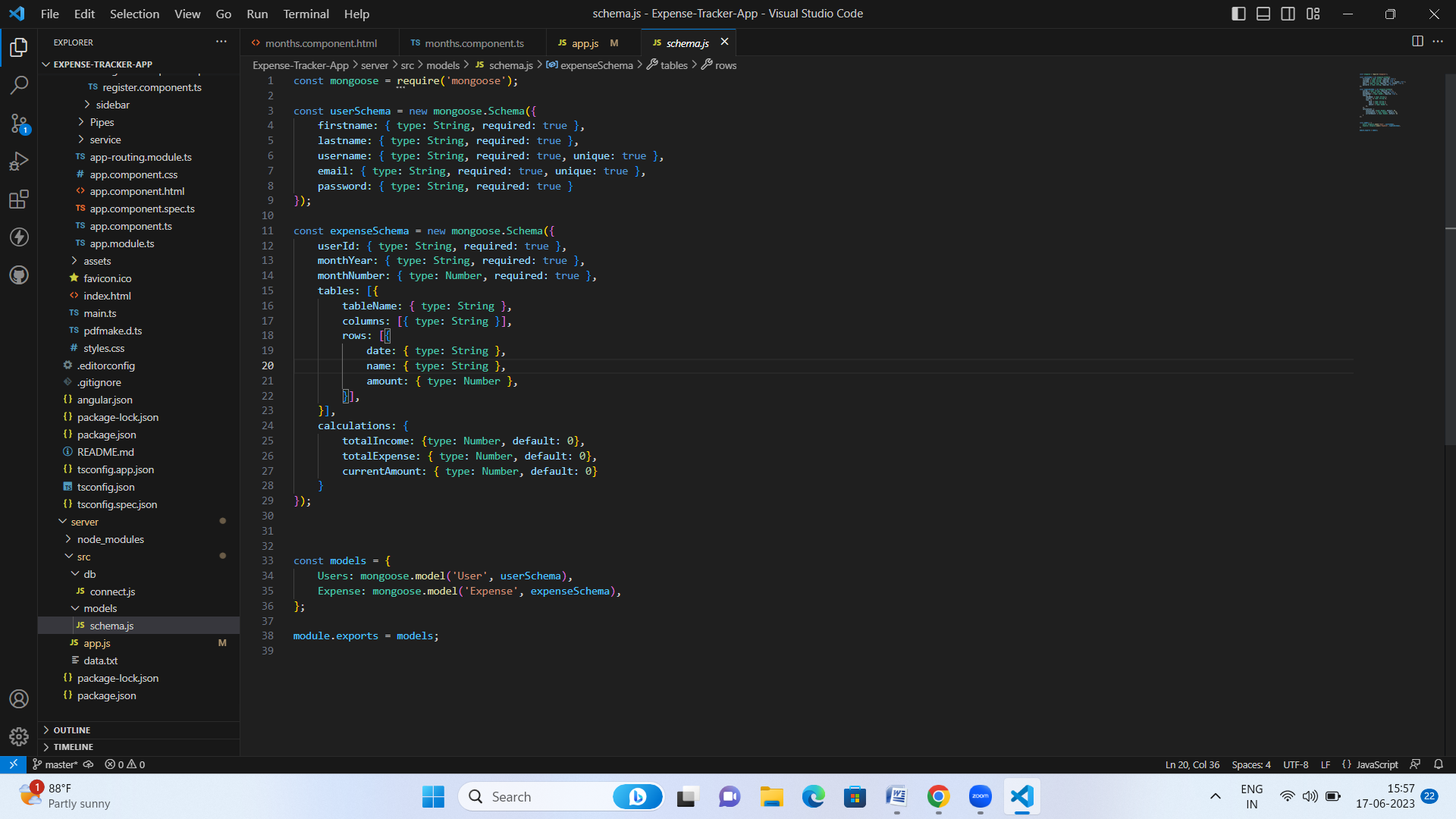
Purpose: Represents the schema and model for expense data. It includes fields such as userId, monthYear, monthNumber, tables, and calculations. This schema is used to store and organize the expenses associated with a user. The tables field allows for storing multiple tables with their columns and rows for expense details. The calculations field stores calculated values such as total income, total expense, and current amount. The Expense model is used to manage expense records within the app.

**Backend Explanation with code snippets:**

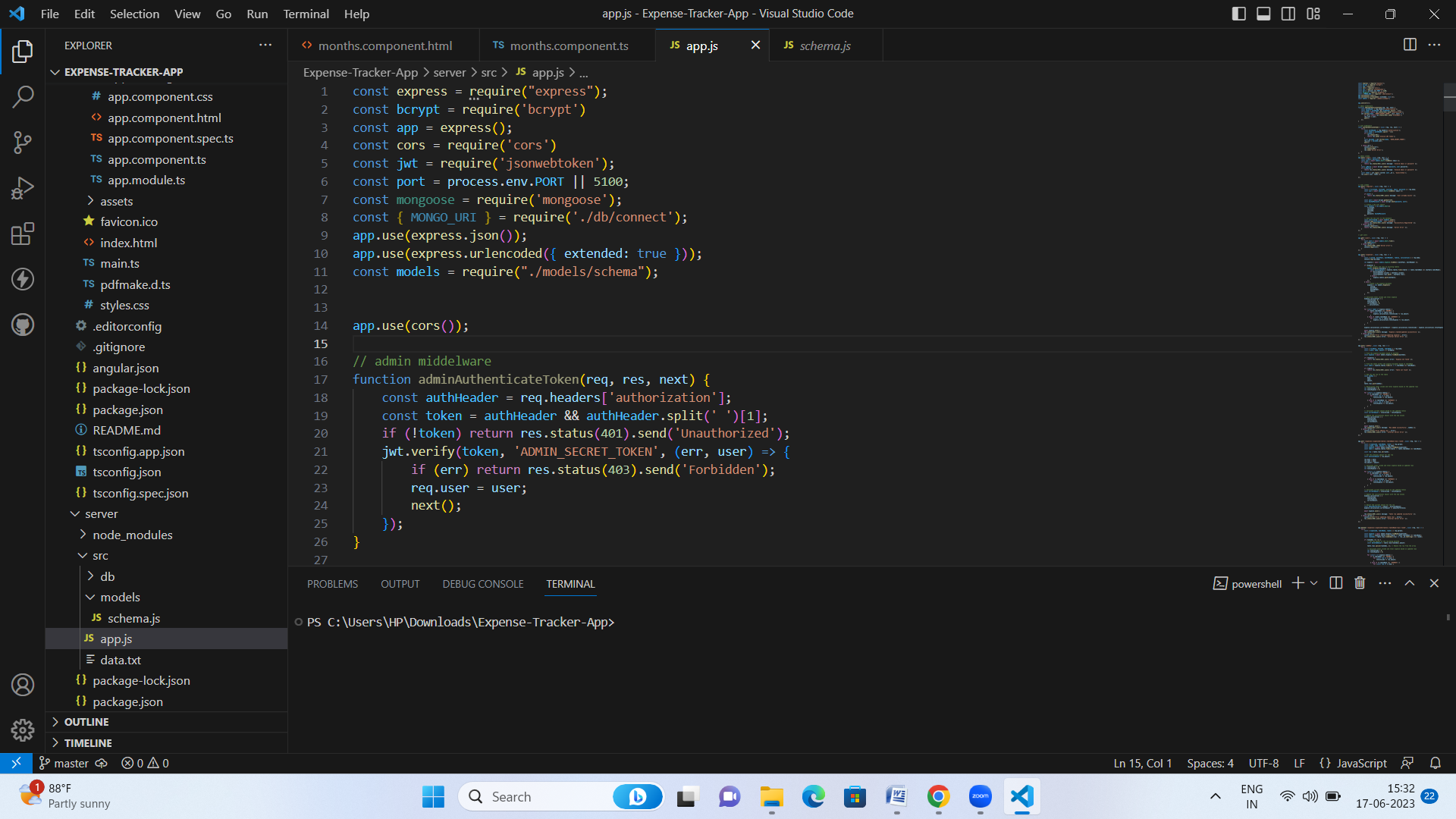
**Database connection:**

****

**Schemas:**

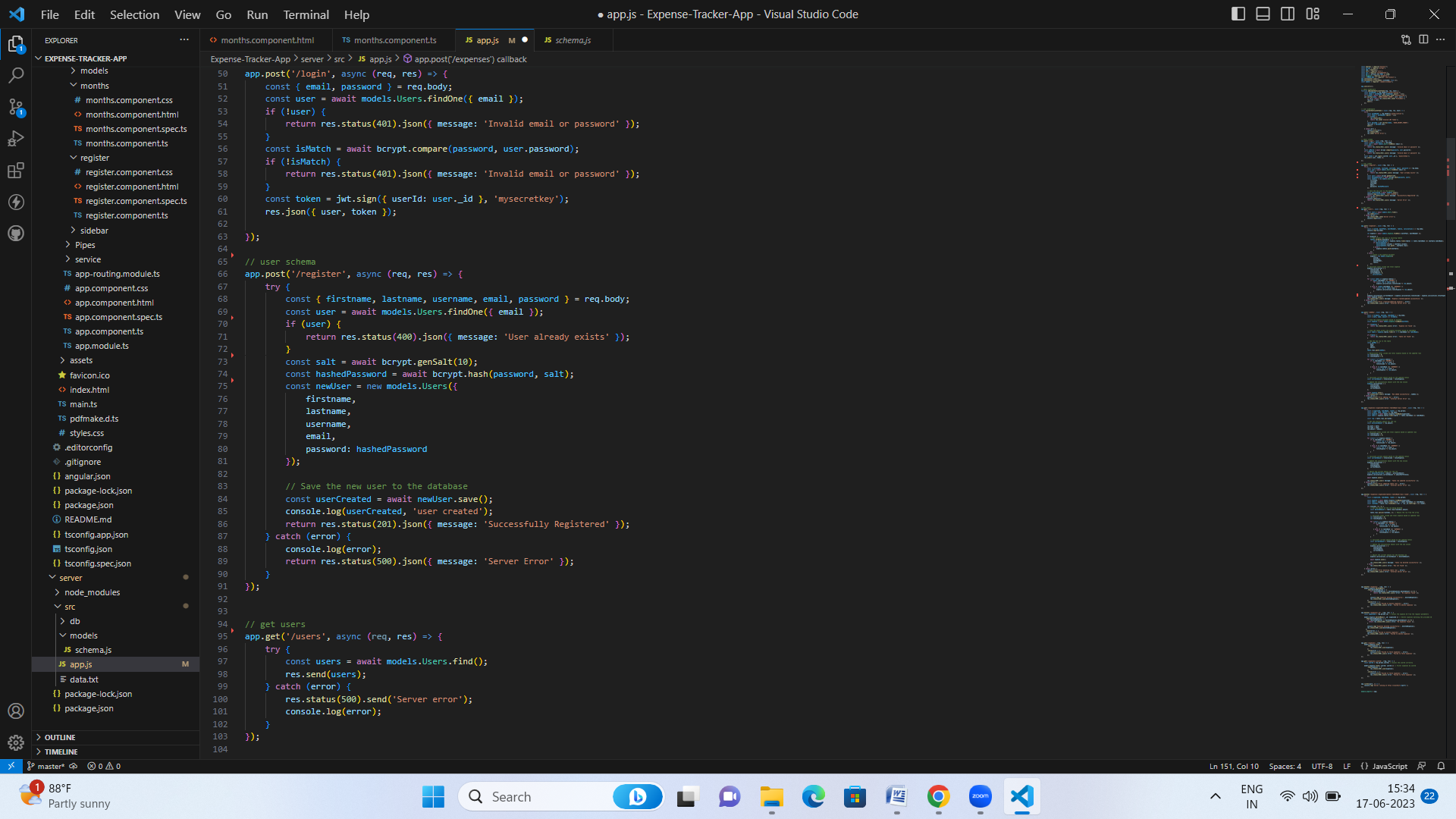
****

**Import Packages:**

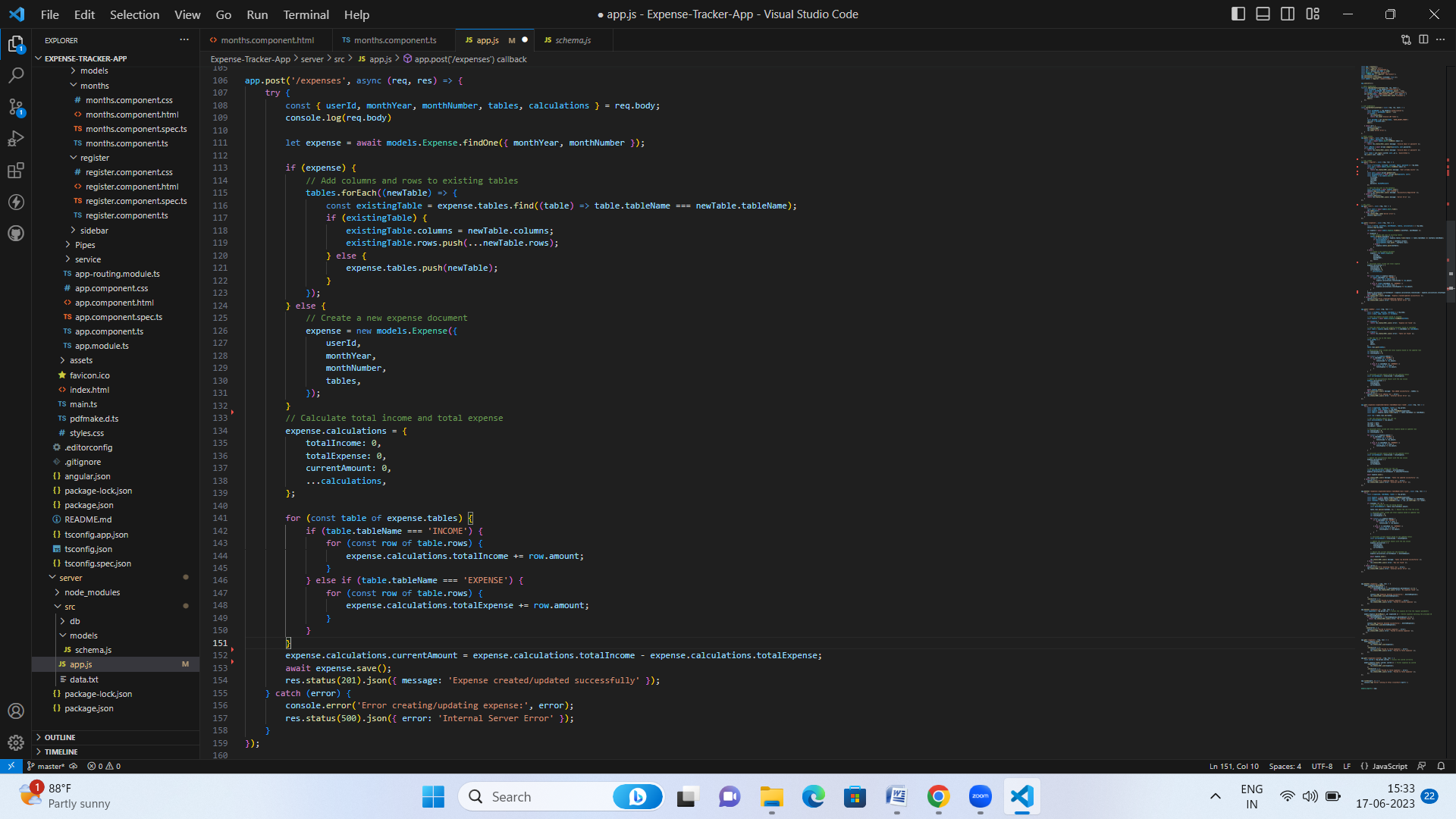
****

**API’s for user register and login:**

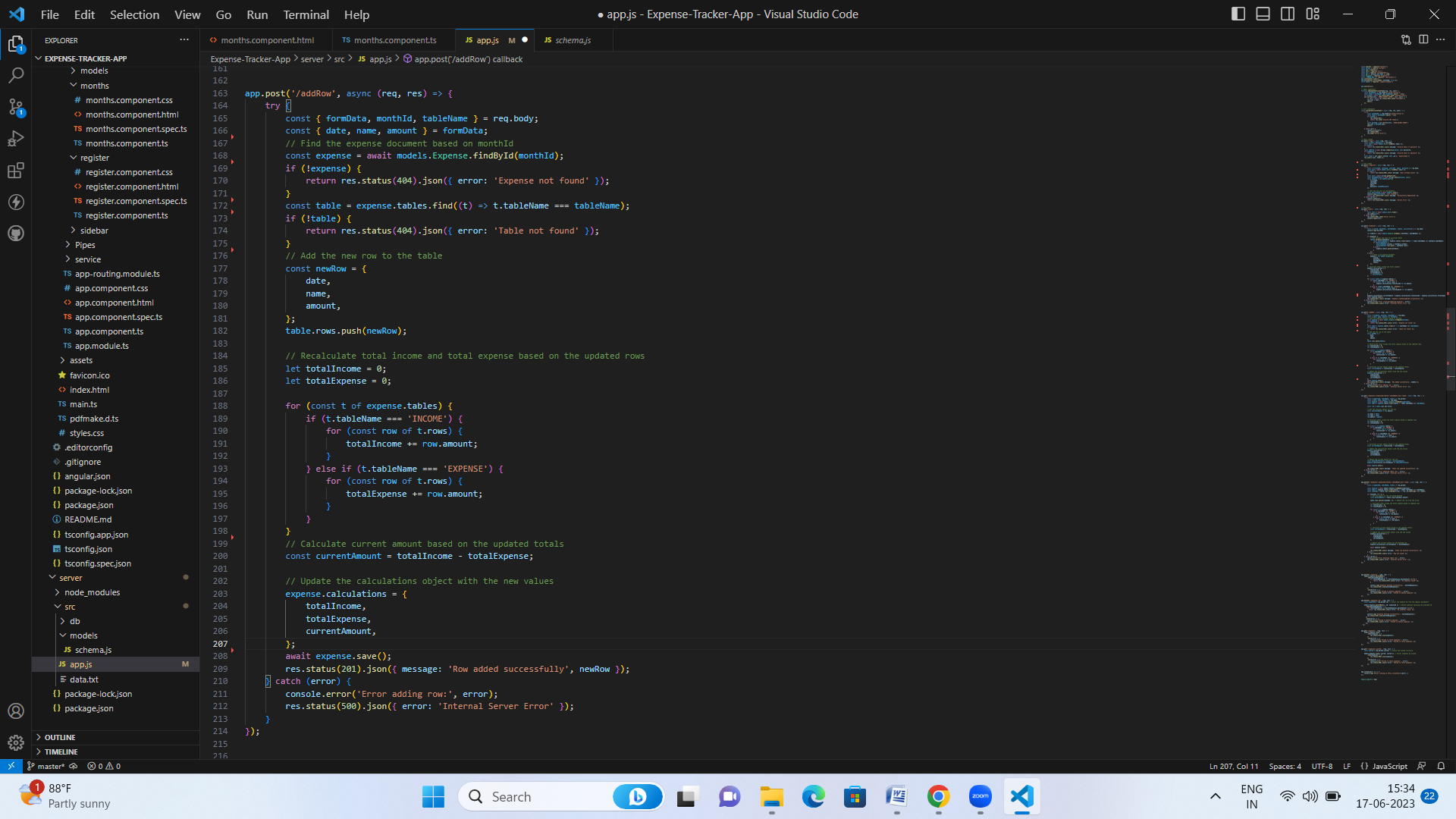
* Register: The process of creating a new account or profile by providing necessary information.
* Login: The act of accessing an existing account or profile by entering valid credentials.

****

**API’s for post Expenses:**

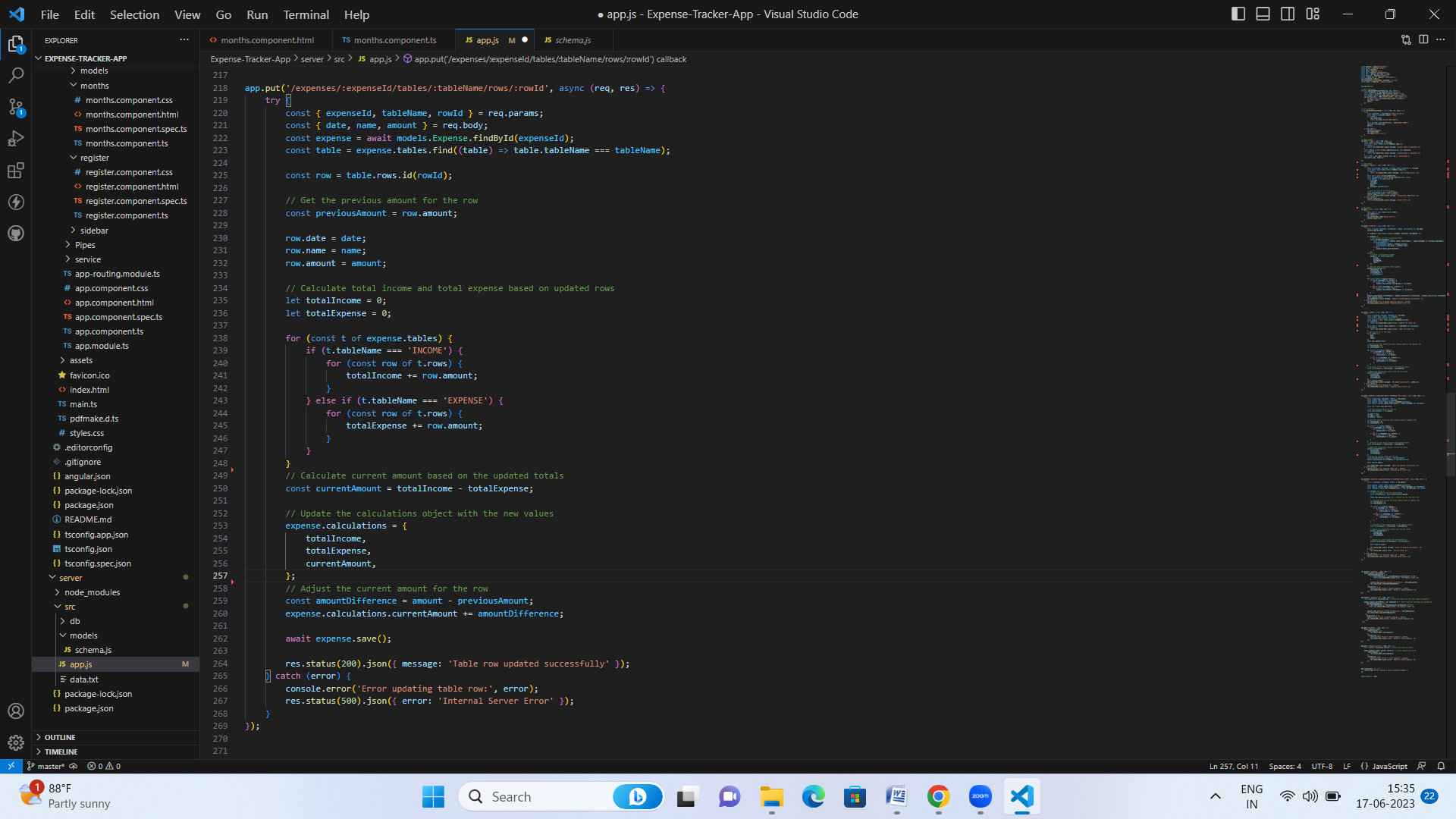
Expenses API: Provides endpoints to create, retrieve, update, and delete expense records for a user in the Expense Tracker app, allowing efficient management and tracking of expenses.****

**API for add-row:**

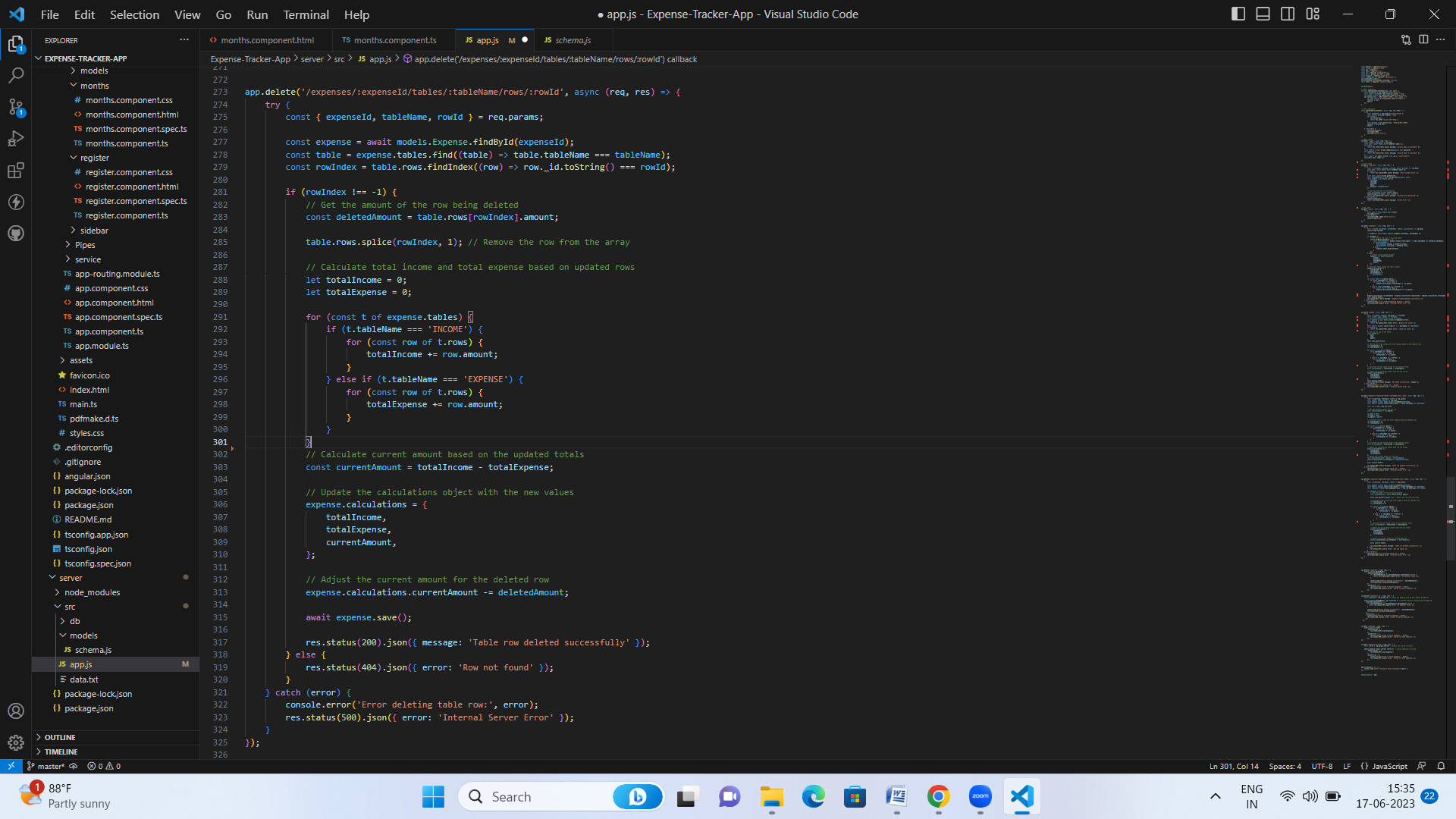
The "add-row" API allows the addition of a new row or entry to a specific table within the Expense Tracker app's expense record, updating the database accordingly.****

**API for update row based on month id:**

The "Update Row" API allows modifying and updating specific row data within a table in the Expense Tracker app's backend. It receives the updated row data and the row's unique identifier, then applies the changes to the corresponding row in the database.



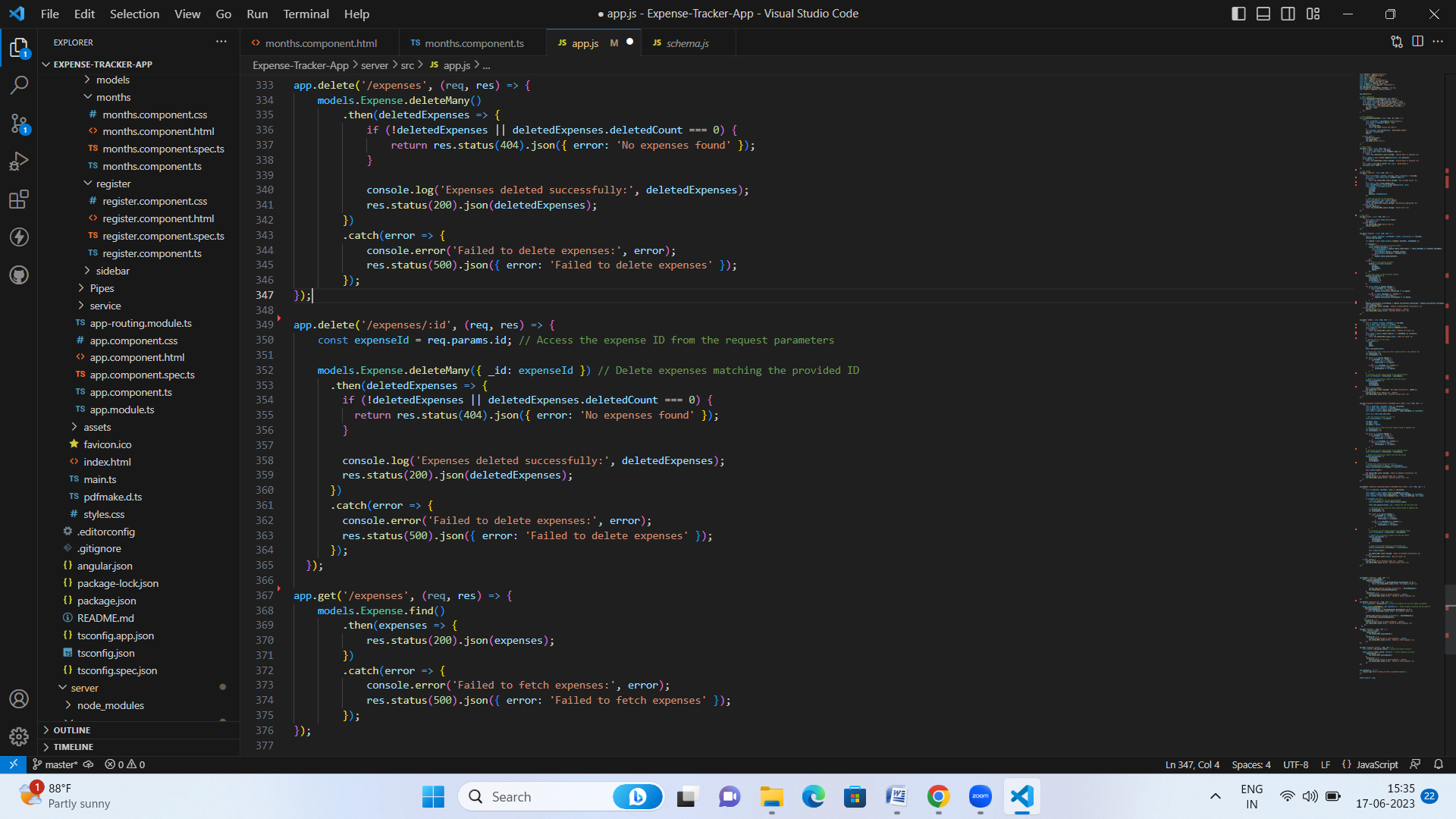
The "Delete Row" API allows removing a specific row from a table in the Expense Tracker app's backend. It receives the unique identifier of the row to be deleted and removes the corresponding row from the database.

****

**API’s for delete expense, get expense:**

The "Delete Expense" API enables deleting a specific expense from the Expense Tracker app's backend. It receives the unique identifier or ID of the expense to be deleted and removes it from the database.

The "Get Expense" API retrieves a specific expense from the backend of the Expense Tracker app. It typically requires the unique identifier or ID of the expense as input and retrieves the corresponding expense from the database for further processing or display.



The "Get Expense by ID" API retrieves an expense record from the Expense Tracker app's backend based on the provided expense ID. It fetches the expense with the specified ID from the database and returns the corresponding expense data to the caller.