**API Documentation for "PocketSense" - College Financial Management System**

**By – Pratyush Puri Goswami**

**ppurigoswami2002@gmail.com**

**1. Introduction**

The **PocketSense** API is designed to provide a robust financial management platform for students, enabling them to effectively track, manage, and share financial responsibilities. The system is built with scalability, security, and user experience in mind, ensuring that students can easily track their expenses, resolve disputes, and collaborate with others on shared financial goals.

**Core Features:**

* **Expense Tracking**: Students can record and categorize their expenses, making it easy to track spending habits over time.
* **Smart Expense Splitting**: The platform leverages AI algorithms to automatically suggest optimal ways to split group expenses, ensuring fairness and transparency.
* **Savings Management**: Students can set savings goals and track their progress, with gamified elements to make saving more engaging.
* **Dispute Resolution**: Users can raise disputes over expenses and payment splits, with a voting mechanism to resolve discrepancies.
* **Collaborative Financial Management**: The system facilitates collaboration within groups, such as project teams or social circles, to share financial responsibilities and manage shared expenses.

**Technologies Used:**

* **Django**: The backend framework used to handle API requests, data modeling, and business logic.
* **Django REST Framework**: The tool used to build RESTful APIs for interaction with the frontend and external services.
* **PostgreSQL**: The relational database used to store all system data, ensuring high availability and scalability.
* **JWT Authentication**: Ensures secure communication and authorization between the client and server.
* **AI Algorithms**: Used for smart expense splitting based on historical spending patterns and group dynamics. I used pandas and numpy for analysis.

**2. Database Design and Models**

The core data models of **PocketSense** are designed to reflect the various aspects of financial management, including expenses, groups, settlements, and categories. Below is an expanded breakdown of these models:

**Expenses Model**

* **Fields**:
  + amount (float): The total cost of the expense.
  + category (foreign key): A link to the **Categories** model, categorizing the expense (e.g., food, travel).
  + split\_type (char): Defines how the expense is split among group members (e.g., equally, custom percentages).
  + date (datetime): The date and time when the expense was recorded.

This model allows users to record individual expenses and assign them to categories like food, travel, and entertainment. It also supports flexible split types, ensuring that expenses can be divided fairly among participants.

**Students Model**

* **Fields**:
  + college (char): The name of the college the student is affiliated with.
  + semester (integer): The semester in which the student is currently enrolled.
  + default\_payment\_methods (JSON): A list of default payment methods (e.g., UPI, credit card) linked to the student's profile for easier payment processing.

The **Students** model extends the default Django User model, adding student-specific information to help manage financial interactions. By linking students with their colleges, this model ensures that only authenticated college students can access the platform.

**Groups Model**

* **Fields**:
  + group\_name (char): The name of the group (e.g., "Project Team A", "Trip Group").
  + group\_type (char): The type of group (e.g., social, project, trip, etc.).
  + members (many-to-many relation): A list of students that belong to the group.

This model is used to organize students into groups, which are essential for tracking shared expenses. Whether for social outings or collaborative projects, each group can independently manage expenses and track payments.

**Settlements Model**

* **Fields**:
  + payment\_status (char): The status of the settlement (e.g., "pending", "paid", "overdue").
  + settlement\_method (char): The method used for settlement (e.g., UPI, bank transfer, cash).
  + due\_date (datetime): The date by which the settlement must be completed.

The **Settlements** model is responsible for tracking how group expenses are settled between members. Each expense can be associated with one or more settlements, which track the progress and method of payment.

**Categories Model**

* **Fields**:
  + food (boolean): Whether the category is related to food expenses.
  + travel (boolean): Whether the category is related to travel expenses.
  + academics (boolean): Whether the category is related to academic-related expenses.
  + entertainment (boolean): Whether the category is related to entertainment expenses.

The **Categories** model helps classify and organize expenses into predefined categories. This allows users to filter and analyze expenses based on type, making it easier to identify spending patterns and track financial goals.

**3. Authentication System**

The authentication system of **PocketSense** is designed to ensure security and provide seamless access to the platform. It leverages modern technologies for both secure login and social authentication.

**JWT Token-based Authentication**

* **JWT (JSON Web Tokens)** is used to secure communication between the client and the server. When a user logs in or signs up, they receive a JWT token that is used for subsequent API requests.
* Tokens are short-lived for added security and can be refreshed to ensure a smooth user experience.

**How it works:**

* A user sends a login request with their credentials (username/email and password).
* If the credentials are valid, the system generates a JWT token and returns it to the client.
* The client includes the token in the **Authorization** header for all subsequent requests. The server validates the token to ensure the user is authorized to perform the requested actions.

**How it works:**

* The user is redirected to the chosen social media provider (Google/Facebook) for authentication.
* Upon successful authentication, the provider sends an authorization code to the server.
* The server exchanges this authorization code for an access token, which can be used to fetch user information (like email and name) and create or link a user account in the system.

**College Email Verification**

* To ensure that only students from accredited colleges can access the platform, **PocketSense** requires users to verify their college email address during registration.
* The system sends a verification email containing a unique token to the provided college email address.
* Upon confirming their email, the user is granted full access to the platform.

**Why it’s necessary:**

* Restricting access to college students ensures that the platform remains secure and relevant for its intended user base.
* By requiring a valid college email, **PocketSense** can prevent misuse by non-students or unauthorized users.

**UPI Payment Linking**

* Students can link their **UPI (Unified Payments Interface)** accounts to the platform for easy, direct payments within the app.
* This feature simplifies the process of settling expenses between group members, allowing users to transfer money directly through the platform without needing to switch to external apps.

**How it works:**

* Users are prompted to link their UPI ID during the setup process.
* Once linked, UPI payments can be made directly from within the app, facilitating quick and secure financial transactions.

**Context-Aware Permissions**

* The platform incorporates **context-aware permissions**, ensuring that access to certain features is granted only when necessary.
* For example, permissions for creating group trips or planning collaborative activities are automatically granted during group creation and revoked when the activity ends.
* This helps maintain security and ensures that users have access only to the features relevant to them at any given time.

I have tested some key features and responses in Postman app.

Since, this is development stage for now, when your team have important changes to make and some key things we want to fix, we can meet and fix them.

This is my current submission, I hope the inclusions so far impressed you, if any.