Daily Expenses Sharing Application - Backend

This project implements the backend for a daily expenses sharing application. It allows users to add expenses, split them using different methods, and generate a balance sheet. The application supports the following features:

- User management (create and retrieve users).
- Expense management (add expenses, retrieve user-specific and overall expenses).
- Expense split methods: Equal, Exact, Percentage.
- Balance sheet generation.
- API endpoints for managing users and expenses.

Features

- **User Management**: Each user has a name, email, and mobile number.
- **Expense Splitting**:
  - **Equal Split**: Expenses are split equally among participants.
  - **Exact Split**: Specific amounts can be assigned to participants.
  - **Percentage Split**: Expenses are split based on percentages (must add up to 100%).
- **Balance Sheet**: A balance sheet showing individual and overall expenses can be generated and downloaded.

API Endpoints

Endpoint	Method	Description
/api/users/	POST	Create a new user
/api/users/<int:user_id>/	GET	Retrieve user details by ID
/api/expenses/	POST	Add a new expense
/api/expenses/user/<int:user_id>/	GET	Get all expenses for a specific user
/api/expenses/all/	GET	Get all expenses
/api/expenses/balance-sheet/	GET	Download the balance sheet

Installation and Setup

Prerequisites

- Python 3.x
- Django 4.x
- SQLite (default Django database)

Step 1: Clone the repository

bash
Copy code
git clone https://github.com/your-username/expense-sharing-app.git
cd expense-sharing-app

Step 2: Create a virtual environment

bash
Copy code
```
python -m venv env
source env/bin/activate   # On Windows use `env\Scripts\activate`
```

Step 3: Install dependencies

bash
Copy code
```
pip install -r requirements.txt
```

Step 4: Set up the database

Run the following commands to create the necessary database tables:

bash
Copy code
```
python manage.py makemigrations
python manage.py migrate
```

Step 5: Run the development server

bash
Copy code
```
python manage.py runserver
```

The server will be available at http://127.0.0.1:8000/.

Step 6: Access the admin panel (Optional)

You can create an admin user and access the Django admin panel to manage users and expenses.

bash
Copy code
```
python manage.py createsuperuser
```

Go to http://127.0.0.1:8000/admin/ and log in with the admin credentials you created.

API Usage

1. Create a User

bash
Copy code
```
POST /api/users/
```

Request Body:

json
Copy code
```
{
  "name": "John Doe",
  "email": "john@example.com",
  "mobile": "1234567890"
}
```

2. Add an Expense

bash
Copy code
POST /api/expenses/

Request Body:

json
Copy code
```
{
  "description": "Dinner with friends",
  "total_amount": 3000,
  "split_method": "equal",  # or "exact", "percentage"
  "participants": [
    {"user_id": 1, "amount": 1000},  # Only required for "exact"
    {"user_id": 2, "amount": 1000},  # Only required for "exact"
    {"user_id": 3, "amount": 1000}   # Only required for "exact"
  ]
}
```

3. Get User Expenses

bash
Copy code
GET /api/expenses/user/<user_id>/

4. Download Balance Sheet

bash
Copy code
GET /api/expenses/balance-sheet/

Data Validation

- For the percentage split method, ensure that the percentages add up to 100%. If not, an error message will be returned.

Testing

You can test the API using tools like Postman or curl.

Example request using curl:

bash
Copy code
```
curl -X POST http://127.0.0.1:8000/api/users/ \
    -H "Content-Type: application/json" \
    -d '{"name": "John Doe", "email": "john@example.com", "mobile": "1234567890"}'
```

Future Improvements

- **Authentication**: Implement user authentication (e.g., JWT).
- **Error Handling**: Add robust error handling for input validation.
- **Unit Tests**: Add unit tests and integration tests.
- **Performance**: Optimize the system for handling large datasets.