**CLI Application Documentation**

**Overview**

This CLI application will allow users to register, log in, and perform various operations such as saving and reading data from a database. The application will be structured as follows:

1. User Registration
2. User Login
3. Main Menu after Login
4. Data Management (Saving and Reading from DB)
5. **Command Prompt Menu Flow**

* **Welcome Message**: Greet the user when the application starts.
* **Main Menu**: Provide options for registration and login.
* **Registration**: If the user chooses to register, prompt for a username and password.
* **Login**: If the user chooses to log in, prompt for their username and password.
* **Post-Login Menu**: After successful login, show the main options.
* **Exit**: Allow the user to exit the application gracefully

1. **Post-Login Menu Options**

* **Save Data**: Allows the user to input and save data into the database.
* **Read Data**: Fetches and displays saved data from the database.
* **Update Data**: Enables the user to modify existing data.
* **Delete Data**: Allows the user to remove specific data from the database.
* **View Profile**: Displays the user's profile information (e.g., username).
* **Logout**: Logs the user out and returns them to the main menu.

1. **Signal Menu Options**

* **Create a New Signal**
  + **Signal Name**: A descriptive name for the signal.
  + **Signal Type**: (e.g., buy/sell, alert type).
  + **Value**: The actual value of the signal (e.g., price point, threshold).
  + **Timestamp**: The time when the signal was generated.
  + **Notes/Comments**: Additional information or context about the signal.
* **View Existing Signals**
  + Option to display all saved signals with relevant details.
* **Update a Signal**
  + **Signal ID**: Identify which signal to update.
  + **New Signal Name**: (if applicable).
  + **New Value**: Update the value of the signal.
  + **New Notes/Comments**: Any updates to the context.
* **Delete a Signal**
  + **Signal ID**: Identify which signal to delete.
* **Filter Signals**
  + **Signal Type**: Filter based on the type of signal.
  + **Date Range**: Filter based on the time period the signals were created.

**Steps for Documentation:**

1. Problems

2. Actors

3. Responsibility of each actor

4. What is the menu if an actor login

5. How many tables required for your application

6. SQL code to create a DB Structure

7. DB queries for each operation done by an actor.

**Expenses Tracker Python Project**

**Project Overview**

The Expenses Calculator is a Python-based tool designed to help users manage and analyze their personal expenses. It allows users to add, edit, and track their daily expenses while providing insights through summaries and charts.

**Features**

* Add, update, and delete expenses
* Categorize expenses (e.g., food, transportation, entertainment)
* View total expenses over time (daily, weekly, monthly)
* Generate expense reports and insights
* Export reports in various formats (e.g., CSV, PDF)

**1. Problems to Solve**

* Tracking of personal expenses over time (monthly and yearly)
* Generating reports to show spending trends and patterns
* Storing data securely for future reference
* Enabling easy data entry, update, and deletion by the user

**2. Actors**

* **User**: The person using the CLI application to track and view their expenses.
* **Admin (Optional)**: If you wish to include different roles, an admin could be added for managing configurations, user data, or categories.

**3. Responsibilities of Each Actor**

* **User**:
  + Add, update, delete, or view expenses
  + Generate monthly and yearly expense reports
  + View spending patterns and trends
* **Admin (Optional)**:
  + Configure settings (such as currency, categories)
  + View consolidated reports of all users (if multi-user)

**4. Menu Options When an Actor Logs In**

* **User**:
  + Add Expense
  + View Monthly Expenses
  + View Yearly Expenses
  + Edit or Delete an Expense
  + Generate Reports (Monthly/Yearly)
* **Admin** (Optional):
  + Manage Categories
  + Manage Users
  + View All User Reports

**5. Tables Required**

* **Users** (optional if multi-user support is needed)
* **Expenses**
* **Categories** (optional, if you want to categorize expenses, e.g., "Food," "Travel")

**How It Works**

1. **Input Your Expenses**: Log every expenditure, providing details such as date, description, category, and amount.
2. **Monthly and Yearly Summaries**: Obtain comprehensive breakdowns of your spending at regular intervals.
3. **Visual Analytics**: Utilize graphical representations to better understand your financial trends.

**Code Structure**

* **main.py** : The main entry point for the project
* **expenses.py** : Module for handling expense data
* **report.py** : Module for generating reports and charts
* **database.py** : (Optional) Module for database interaction using Postgres SQL

**Application Creation steps:**

**Step 1: Create DB and tables**