**Expense Tracker Project Report**

**Project Report**

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Topic** | **Page no** |
| 1 | Title of Research | 5 |
| 2 | Certificate | 2 |
| 3 | Abstract | 5 |
| 4 | Introduction | 6-9 |
| 5 | Objective | 9- |
| 6 | Programming Language | 9-18 |
| 7 | Data gathering | 19 |
| 8 | System design /UML Diagrams | 20-22 |
| 9 | Proposed system features | 23-25 |
| 10 | Evalution | 26-27 |
| 11 | Future scope | 28 |
| 12 | conclusion | 29 |
| 13 | Biography | 30 |
| 14 | References | 31 |

**Abstract**

**This report documents the development of a Budget Tracker Web Application designed to simplify personal finance management. The application provides:**

* Offline functionality using localStorage
* Intuitive transaction categorization
* Real-time balance calculations
* Responsive design for all devices

Built using HTML5, CSS3, and vanilla JavaScript, the application addresses the gap between complex financial tools and manual tracking methods.

### Introduction

In today’s fast-paced world, managing personal finances effectively has become increasingly important. Many individuals struggle to keep track of their income, expenses, and savings, leading to poor financial decisions and stress. An Expense Tracker is a digital solution designed to help users monitor their daily, weekly, and monthly spending habits.

This project aims to develop a user-friendly Expense Tracker application that allows users to record, categorize, and analyze their financial transactions. By providing real-time insights into spending patterns, the system empowers users to make informed financial decisions, set budgets, and achieve their financial goals. The project also emphasizes data visualization, ease of use, and secure data handling.

3. System analysis

### 3.1 PROBLEM DEFINITION

Managing personal finances is an essential yet often overlooked task for many individuals. With daily transactions such as food, transportation, bills, and shopping, people find it difficult to maintain a proper record of their expenses. Relying on memory or traditional methods like notebooks and spreadsheets is inefficient, error-prone, and lacks real-time visibility.

### 3.2 Preliminary Investigation Purpose

Many individuals face challenges in managing and monitoring their daily and monthly expenses. Existing manual methods such as notebooks or spreadsheets are not user-friendly and lack automation. Some mobile apps are either too complex or contain unnecessary features that confuse users instead of helping them. There is a need for a simple, efficient, and reliable expense tracking system.

* **Technical Feasibility**: The required tools and technologies (like mobile app or web platform) are readily available and accessible.
* **Operational Feasibility**: The system will be easy to use for non-technical users, with a clean and intuitive interface.
* **Economic Feasibility**: Development and maintenance costs are low, especially if open-source tools are used.

Operational Feasibility

* Examines whether the system will work in practice and be accepted by users.

* **For Expense Tracker:**

* + User-friendly interface suitable for all age groups.

* + Reduces manual effort and improves budgeting habits.

* + Supports goal setting, alerts, and reporting features.

## • Economic Feasibility

* Analyzes cost-effectiveness: do the benefits outweigh the costs?

✓ **For Expense Tracker:**

* + - * + Low development and maintenance cost.

* + - * + Open-source technologies reduce licensing fees.

* + - * + Helps users save money by tracking and managing spending.

* Schedule Feasibility
  + - * Determines whether the project can be completed within a given time.

* + - * **For Expense Tracker:**

* + - * + Project can be completed within academic or project deadline.

* + - * + Tasks like UI design, coding, testing, and deployment can be scheduled properly.

* Legal Feasibility
  + - * Ensures the system doesn’t violate any laws or regulations.

* + - * **For Expense Tracker:**

3.4 Project Planning

1. The project planning phase is essential for organizing the development process and

ensuring the successful completion of the Expense Tracker system. The main goal of this project is to build a simple, user-friendly application that allows users to record, categorize, and monitor their expenses efficiently.

* 1. Project Scheduling

Project scheduling involves planning the timeline for each phase of the Expense Tracker development, ensuring the project progresses smoothly and finishes on time. The schedule is divided into well-defined stages, each with a realistic timeframe based on the complexity of tasks.

The project starts with the **Requirement Analysis** phase, where the goals and features of the system are identified. This phase typically takes about one week. During this time, user needs are gathered and documented.

* 1. Software Requirement Specification (SRS)

The Expense Tracker system is designed to help users efficiently record, monitor, and analyze their daily expenses. It provides features to categorize spending, set budgets, generate reports, and send alerts, assisting users in better financial management.

System Overview

* + 1. **Expense:** Any amount of money spent.
    2. **Budget:** A financial limit set for a specific time period. 3. **Category:** Classification of expenses (e.g., Food, Transport).

Software & Hardware Requirements

Flask==3.0.2

python-dotenv=1.0.1

mysql-connector-python==8.3.0

pandas==2.2.1 openpyxl==3.1.2

report lab==4.1.0

Software Requirements

The application follows a 3-tier architecture:

* + - 1. Presentation Layer (HTML/CSS)
      2. Application Layer (JavaScript)
      3. Data Layer (localStorage)

* **Processor**: Intel i5 or higher
* **RAM**: Minimum 8GB
* **Storage**: At least 100GB for database and media files
* **Connectivity**: Internet access for real-time updates

3.7 Functional Requirements

**Add Expenses** – Enter details such as amount, category, date, and description.

**View Expenses** – View a list of all recorded expenses with filtering options (by date, category, amount).

**Search Transactions** – Use keywords or filters to find specific expenses.

1. Sub-Admin Module

Sub-Admins can:

* **Manage Expense Categories** – Add, update, or remove categories within the scope assigned.
* **User Support** – Respond to user queries and help with troubleshooting within their domain.
* **Monitor User Budgets** – View users' budgeting history and spending insights to assist with suggestions.

2 Software Engineering Paradigm

The development of the Expense Tracker system follows a structured yet flexible methodology to ensure clarity, reliability, and user satisfaction. A modified version of the **Waterfall Model** is used, which incorporates feedback loops and improvements during development.

Development Model: Adapted Waterfall Model

* + **Structured Phase Progression** – Clear steps are followed from planning to deployment.

**Iterative Refinement** – Feedback from testing and user trials is used to refine designs and features.

* + **Defined Milestones** – Each phase concludes with a review before proceeding to the next.



Phases of Development

**Requirement Analysis & System Study**

* + Identifying project goals, challenges, and functional specifications.
  + Gathering stakeholder requirements and defining core functionalities such as expense entry, category management, budget setting, and report generation.

**System Design**

* + Structuring the database, modules, and application architecture for an efficient flow of data.
  + Designing user interfaces that are simple, clean, and accessible for all types of users.
  + Creating wireframes and design layouts for mobile/web platforms.

**Implementation (Coding)**

* + Backend development using Python with frameworks such as Django or Flask to handle data logic and API operations.
  + Frontend design using HTML, CSS, JavaScript (or frameworks like React/Flutter for mobile apps).
  + Database integration using MySQL or Firebase for storing user data, expenses, and budgets.
  + Optional: Integration of analytical tools for insights (e.g., chart.js or Google Charts).

**Testing & Debugging**

* + Conducting unit testing on each individual module (expense addition, editing, deletion, etc.).
  + Performing integration testing to ensure modules work together smoothly.
  + Usability testing to check that the application is user-friendly and functions as expected.
  + Debugging to resolve any logical or performance issues before deployment.

**Deployment & Maintenance**

* + Hosting the application on a scalable platform such as Firebase Hosting, Heroku, or a cloud server.
  + Releasing the application to users via a web link or mobile store (if applicable).
  + Providing continuous updates and maintenance based on user feedback and analytics.

3.8 Data Flow Diagram:

**Data Flow Diagram (DFD) for Expense Tracker**

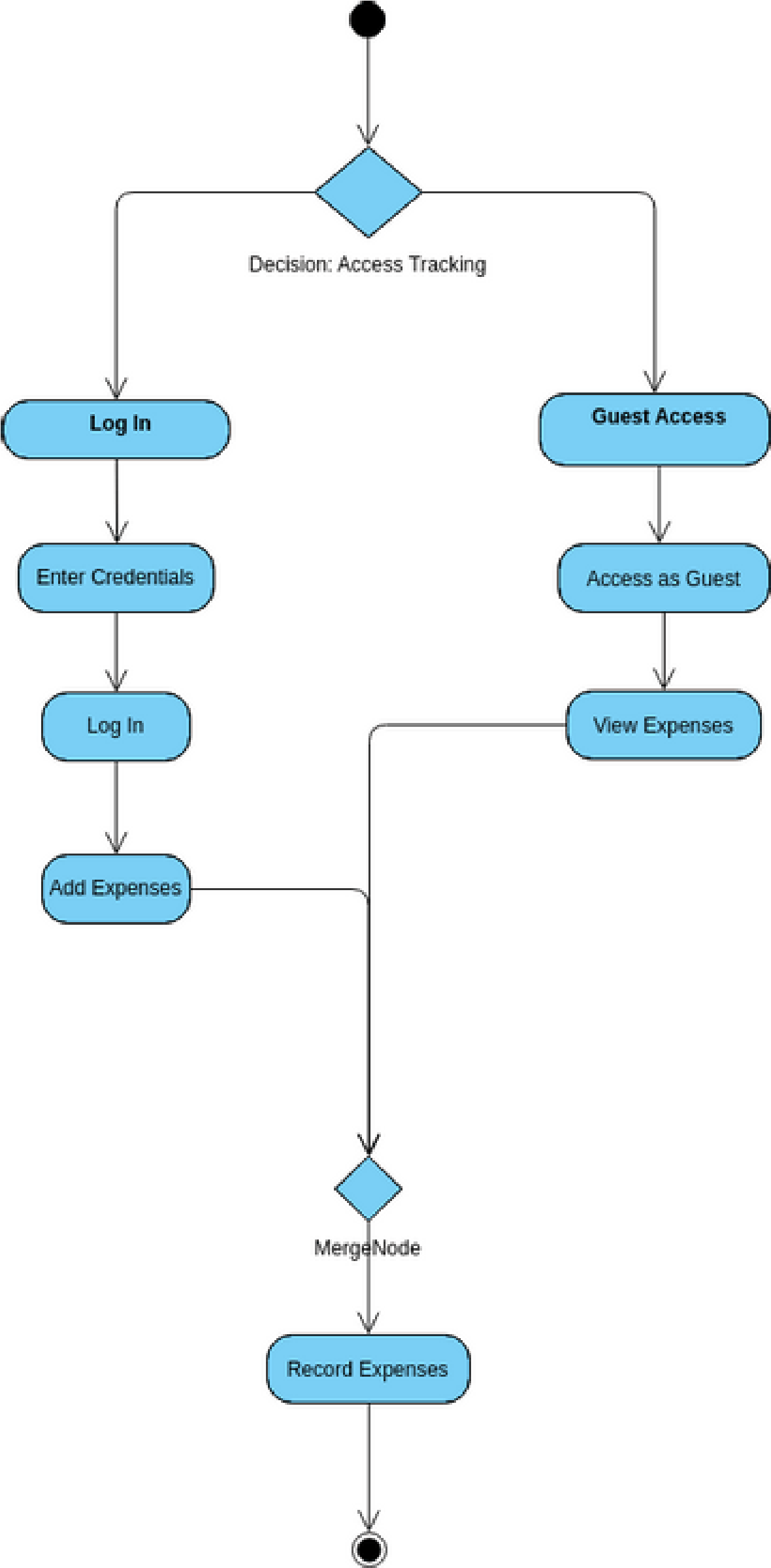
A **Data Flow Diagram (DFD)** is a traditional visual representation of the **information flow** within the **Expense Tracker System**. A neat and clear DFD can depict the right amount of system requirements graphically. It can represent manual, automated, or hybrid processes.

It shows:

* **How data enters** and **leaves** the system,
* **What processes** transform the data,
* And **where data is stored**.

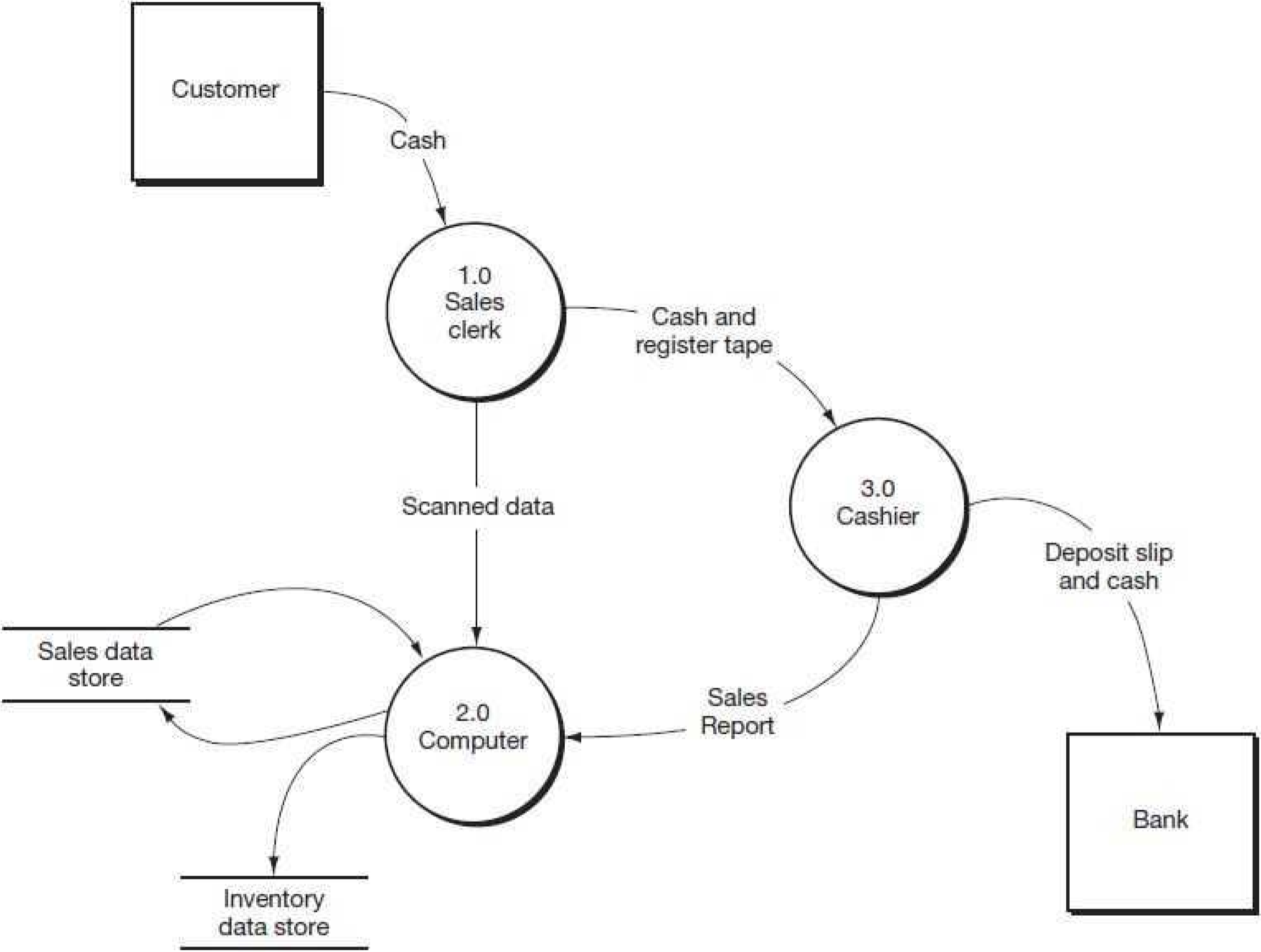
**Purpose of the DFD**

The primary objective of the DFD in the **Expense Tracker** project is to outline the **scope and boundaries** of the application. It serves as a **communication tool** between system developers, project stakeholders, and end users. It also acts as a starting point for **Analyzing and redesigning** a more efficient system



**Data Store:** A set of parallel lines shows a place for the collection of data items. A data store indicates that the data is stored which can be used at a later stage or by the other processes in a different order. The data store can have an element or group of elements.

**Source or Sink**: Source or Sink is an external entity and acts as a source of system inputs or sink of system outputs.



The **Zero Level DFD** of the Online News Portal System (ONPS) shows the main processes

interacting with the system, such as Admin, Sub-Admin, Category, Subcategory, News, Comments, Login, Password, Authorization, and Webpage Management. It highlights the flow of data between the central system (ONPS) and these modules, representing how each core function is managed within the portal.

Daily Expense Tracker Module

Admin Module

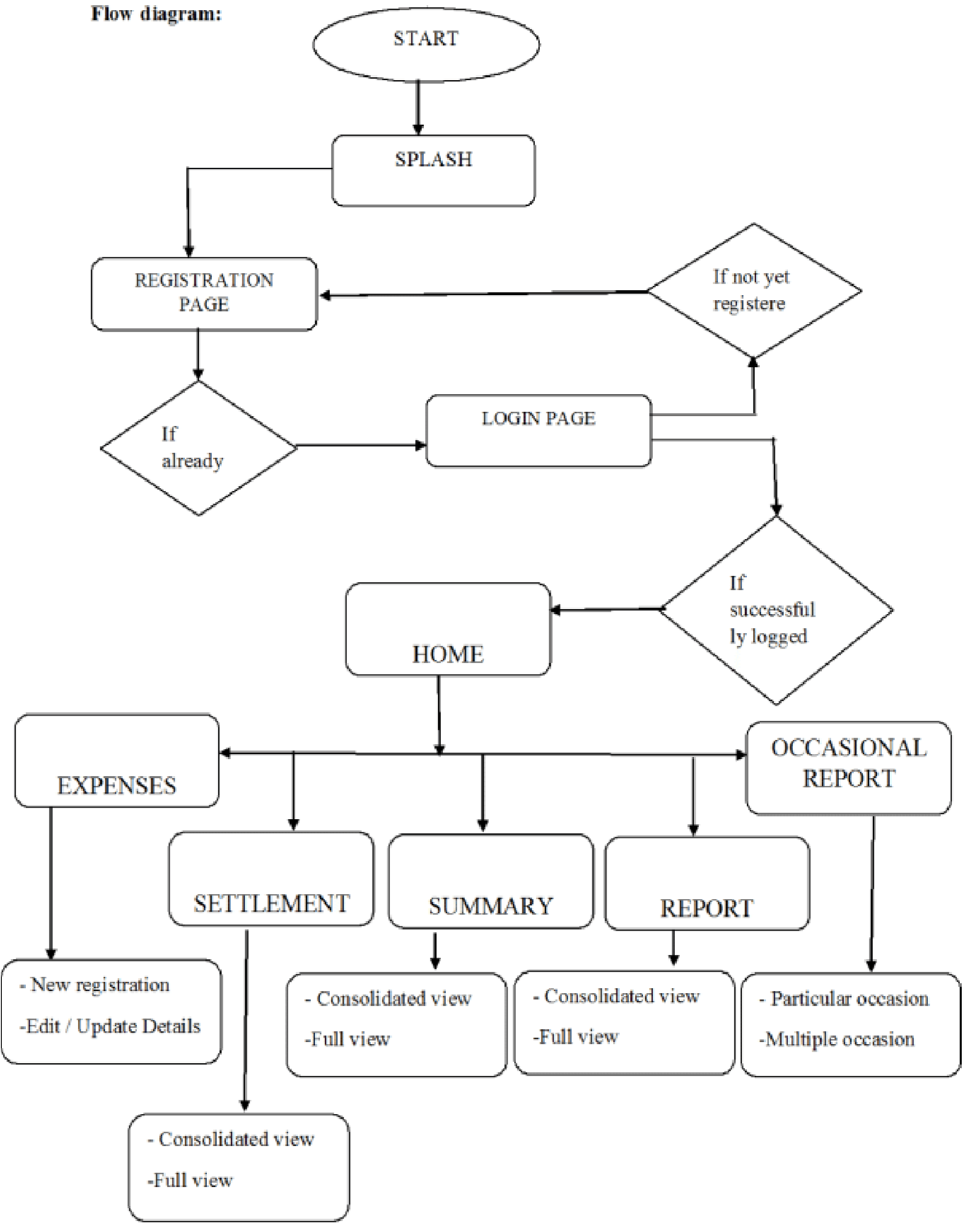
Daily Expense Tracker System

Registration Module

Details Of Users

User Module

The **First-Level DFD** of ONPS shows how the system handles key functions like login, admin, sub-admin, authorization, and password management, and connects them to content- related modules such as news, category, subcategory, webpage, and user comments management for smooth portal operations.



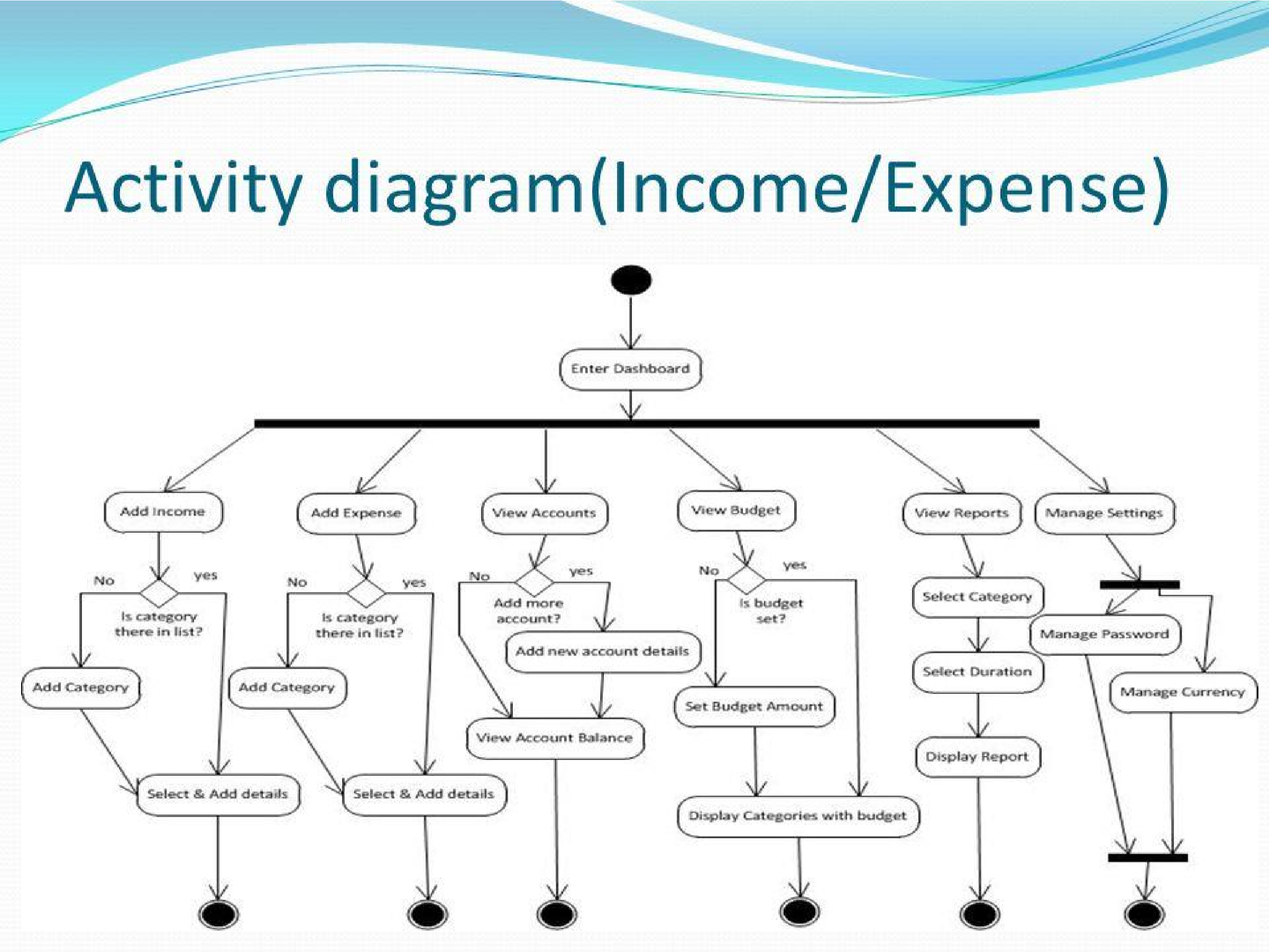


**Sentiment**

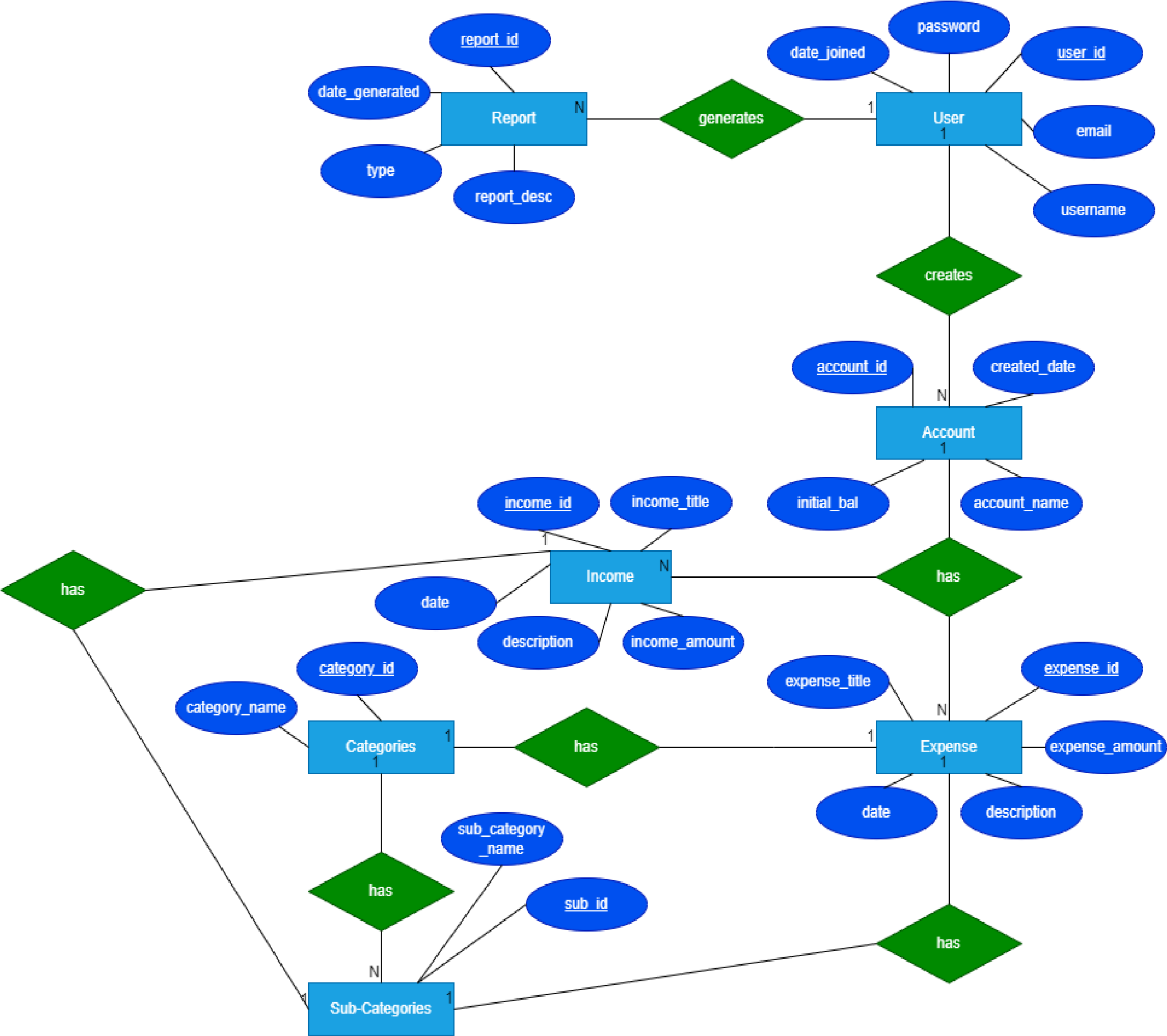
**analysis**

The **Second-Level DFD** shows the internal working of the admin in the ONPS system. It includes processes like login, credential checking, role-based access, and module management. The admin can manage subadmins, categories, subcategories, news, webpages, and user comments, along with profile updates, password changes, and performing sentiment analysis.

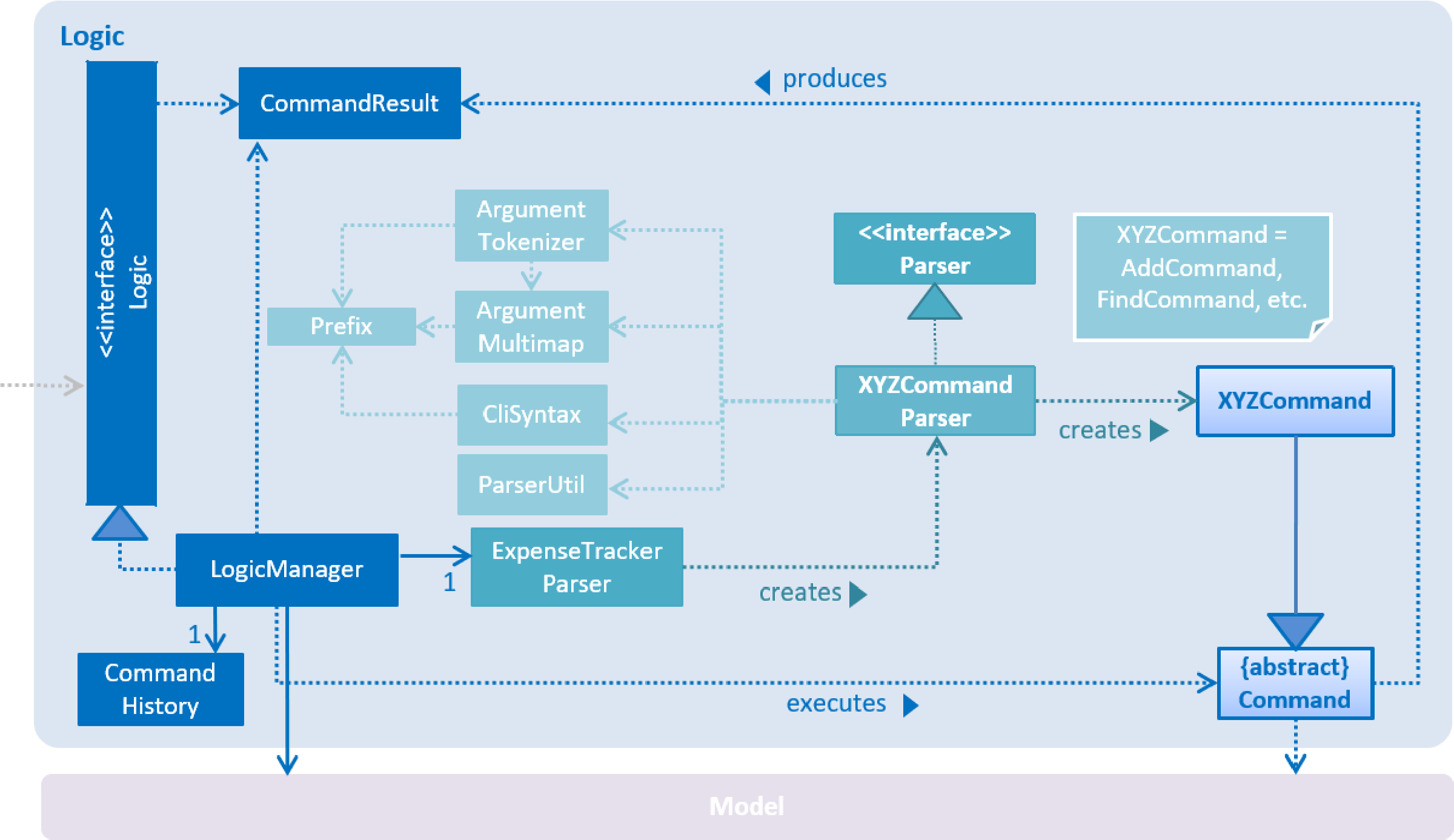
Activity Diagram



ER diagram



system design



4.1.Module

Users can:

* **Add Income** – Enter income details like amount, source, and date.
* **Add Expenses** – Record spending with categories such as Food, Travel, Utilities, etc.
* **View Summary** – Get an overview of income vs. expenses and remaining budget.
* **Generate Reports** – View monthly/weekly reports, pie charts, and graphs.
* **Set Budgets** – Define spending limits for various categories.
* **Search Transactions** – Filter income/expense records using keywords or date range.
* **Edit/Delete Entries** – Modify or remove existing income and expense records.

**Admin Module**

Admins have full control over the system and can:

* **Secure Login System** – Access the admin panel via authentication.
* **Dashboard Management** – Monitor system statistics, recent activity, and user behavior.
* **User Management** – View, block, or remove users and reset passwords.
* **Category Control** – Add, edit, or remove predefined categories and subcategories for expenses.
* **Sub-Admin Management** – Create sub-admin accounts with role-based permissions.
* **Data Backup** – Perform system-wide data export or backups of transactions and users.
* **Audit Logs** – Monitor login attempts, transaction history, and changes made to data.

**Sub-Admin Module**

Sub-Admins can:

* **Manage Transactions** – Add, edit, or delete income/expense entries within assigned user groups or categories.
* **Category Oversight** – Maintain specific categories assigned by the Admin.
* **Generate Insights** – View and export reports and analytics for assigned data.
* **Moderate Entries** – Approve or flag suspicious or incorrect entries by users.
* **Limited Access Control** – Cannot create sub-admins or change system settings but can view assigned sections.

DATA STRUCTURE OF ALL MODULES:

**User Module**

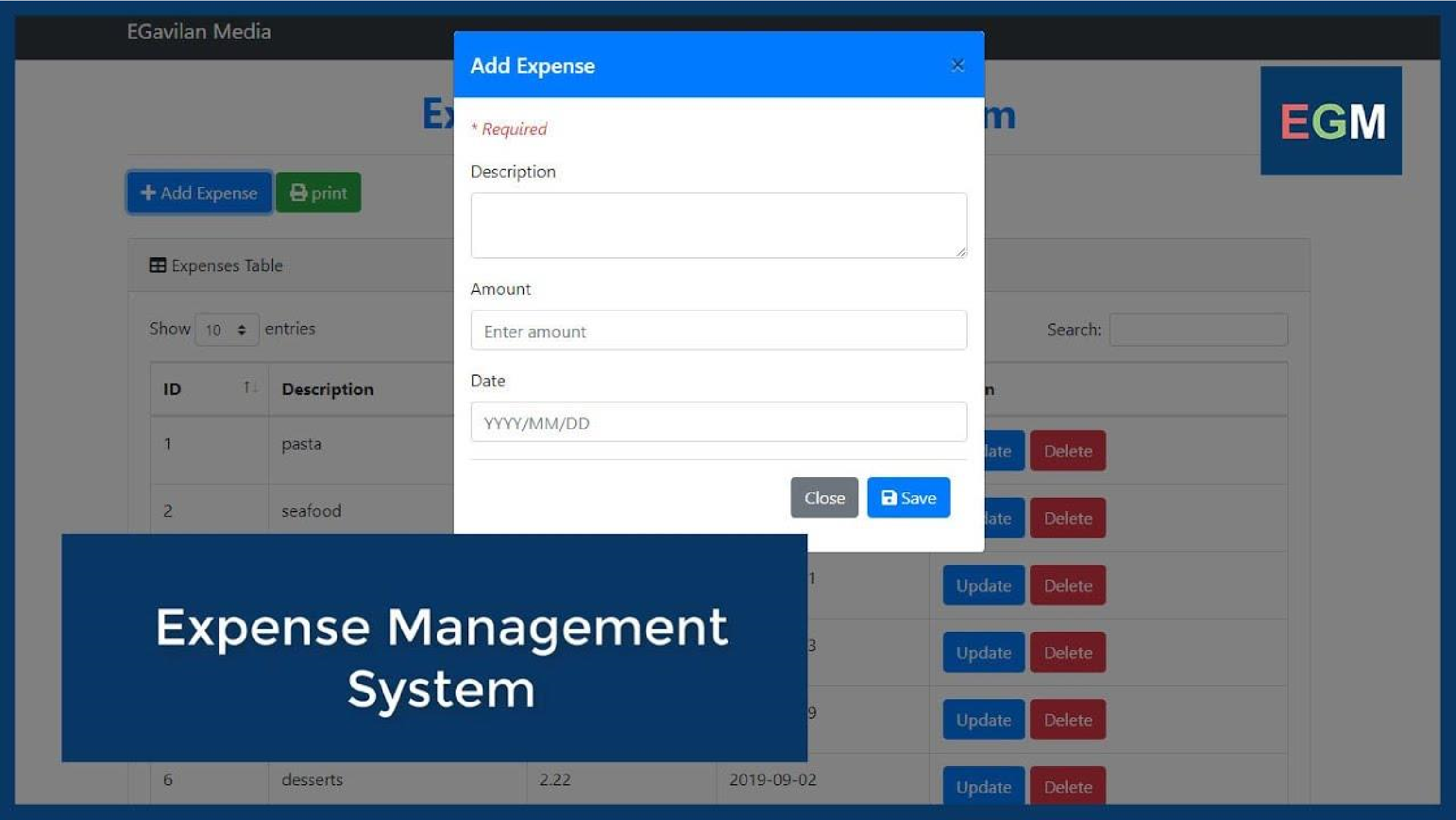
Users can:

* Add income and expense entries.
* Set budgets.
* View summaries and reports.
* Search transactions.
* Edit or delete entries.

**Data Structure:**

1. **User** 
   * + user\_id: Unique user identifier
     + name: Full name
     + email: Email address (used for login)
     + password\_hash: Encrypted password
     + role: 'user', 'admin', or 'sub\_admin'
     + created\_at: Account creation timestamp
     + is\_active: Status of the user (active/inactive)
2. **Income** 
   * + income\_id: Unique income entry ID
     + user\_id: Associated user ID
     + amount: Income amount
     + source: Source of income (e.g., salary)
     + date: Date of transaction
     + notes: Optional notes
     + category\_id: (optional) Category for income
3. **Expense** 
   * + expense\_id: Unique expense entry ID
     + user\_id: Associated user ID
     + amount: Expense amount
     + category\_id: Expense category ID
     + date: Date of transaction
     + description: Description or note
4. **Category** 
   * + category\_id: Unique ID
     + name: Name of category (e.g., Food, Rent)
     + type: 'income' or 'expense'
     + created\_by: Admin or sub-admin who added it
5. **Budget** 
   * + budget\_id: Unique budget entry user\_id: Associated user
     + category\_id: Budget category
     + amount\_limit: Budget limit
     + start\_date: Start date of budget perio

4.2 PROCEDURAL DESIGN:



Start

|

|--> [Dashboard]

|

|--> [Add Expense]

| |

| --> Input Form --> Validate --> Save to DB --> Update Balance

|

|--> [Add Income]

| |

| --> Input Form --> Validate --> Save to DB --> Update Balance

| --> [View Transactions]

|--> [Manage Categories]

|

--> Add/Edit/Delete Category --> Save Changes

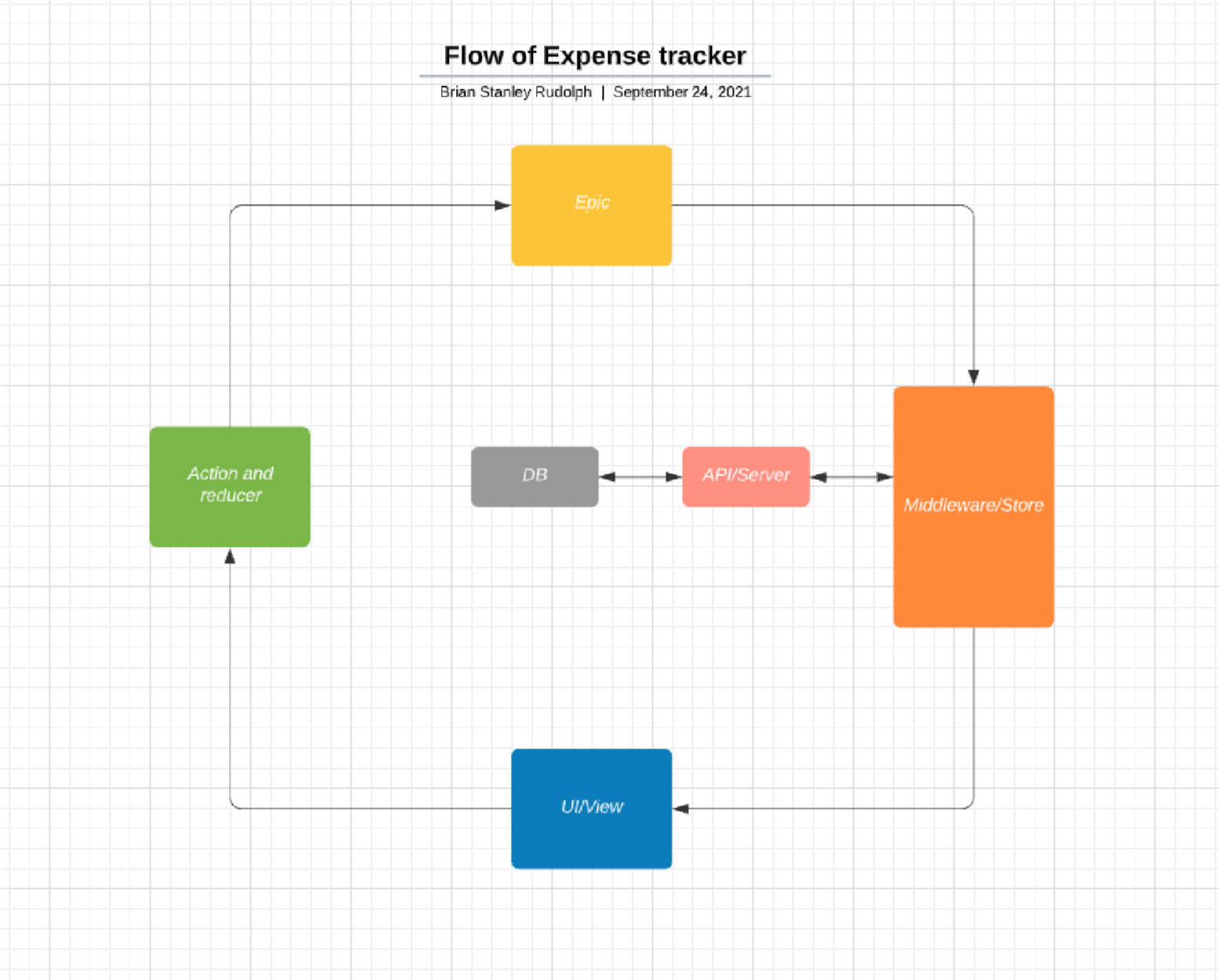
# 4.2.1 Admin Panel Design

In the current version of the **Expense Tracker System**, the Admin Panel is accessible without a user authentication mechanism. This means that any user with access to the system interface can directly open the admin panel and perform administrative functions. While this version does not include a login page or credential verification, the Admin Panel is still designed to centralize all administrative tasks in a single interface.

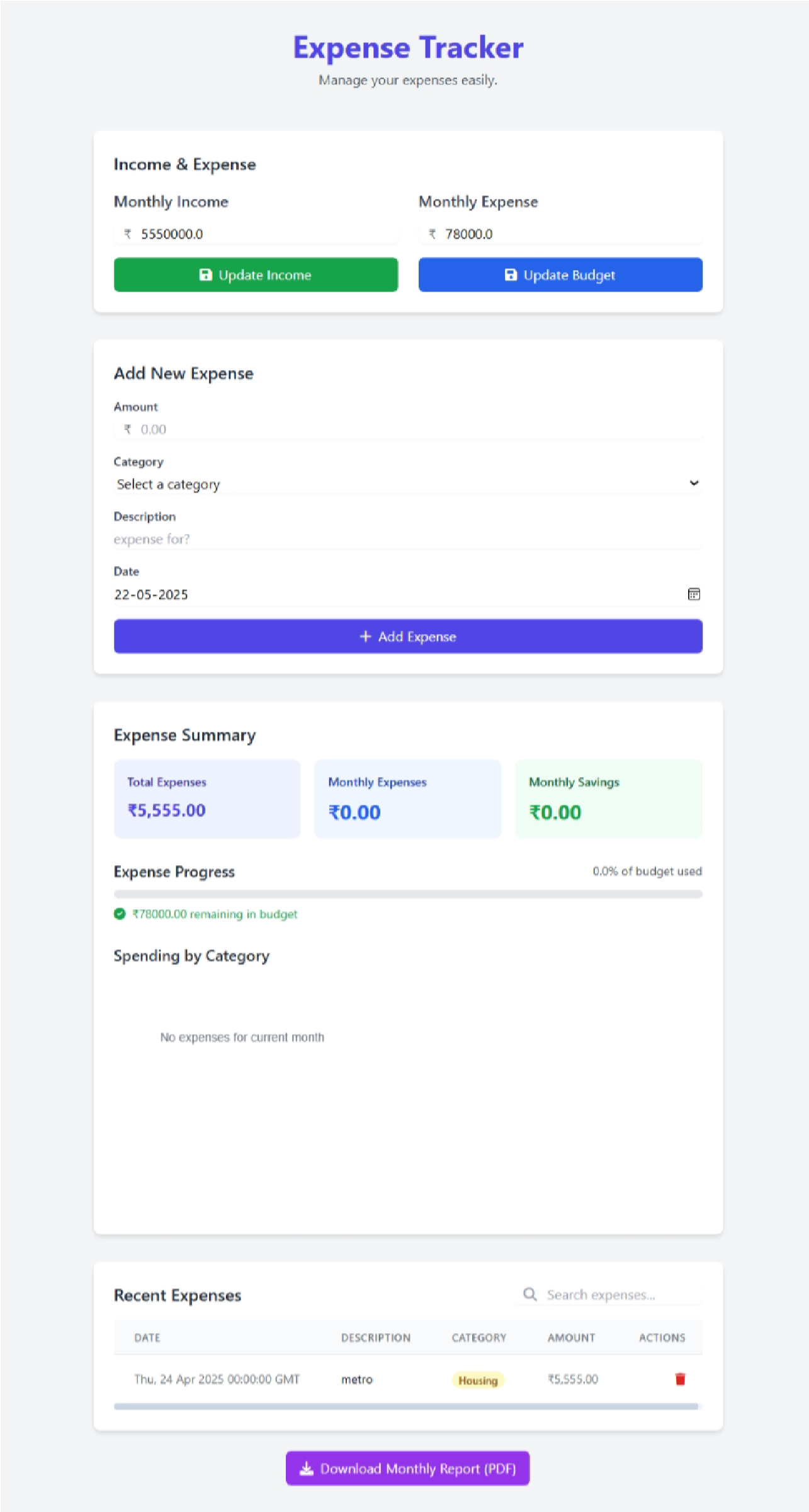
From the Admin Panel, the administrator can manage various components of the system, including income and expense categories, user transaction data (if multi-user), and budget settings. The admin also has the ability to view overall system reports such as category-wise expenditure, monthly summaries, and budget utilization graphs.

The panel interface includes clearly labeled options for adding, editing, or deleting categories, updating budget limits, and viewing detailed financial reports. All changes made in the admin panel reflect immediately in the system, ensuring that data remains current.

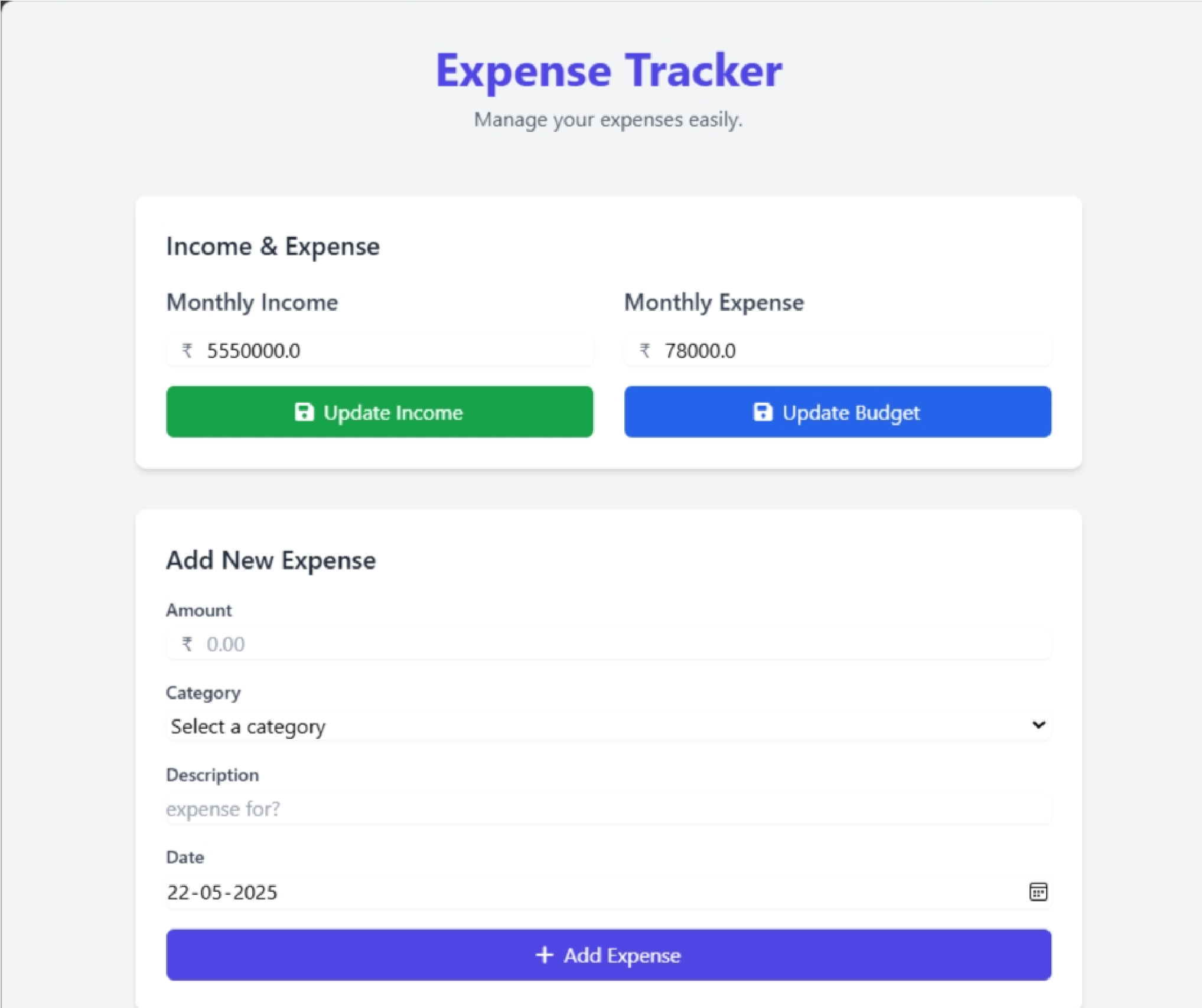
**Fig. 4.2: Login Flow Chart.**

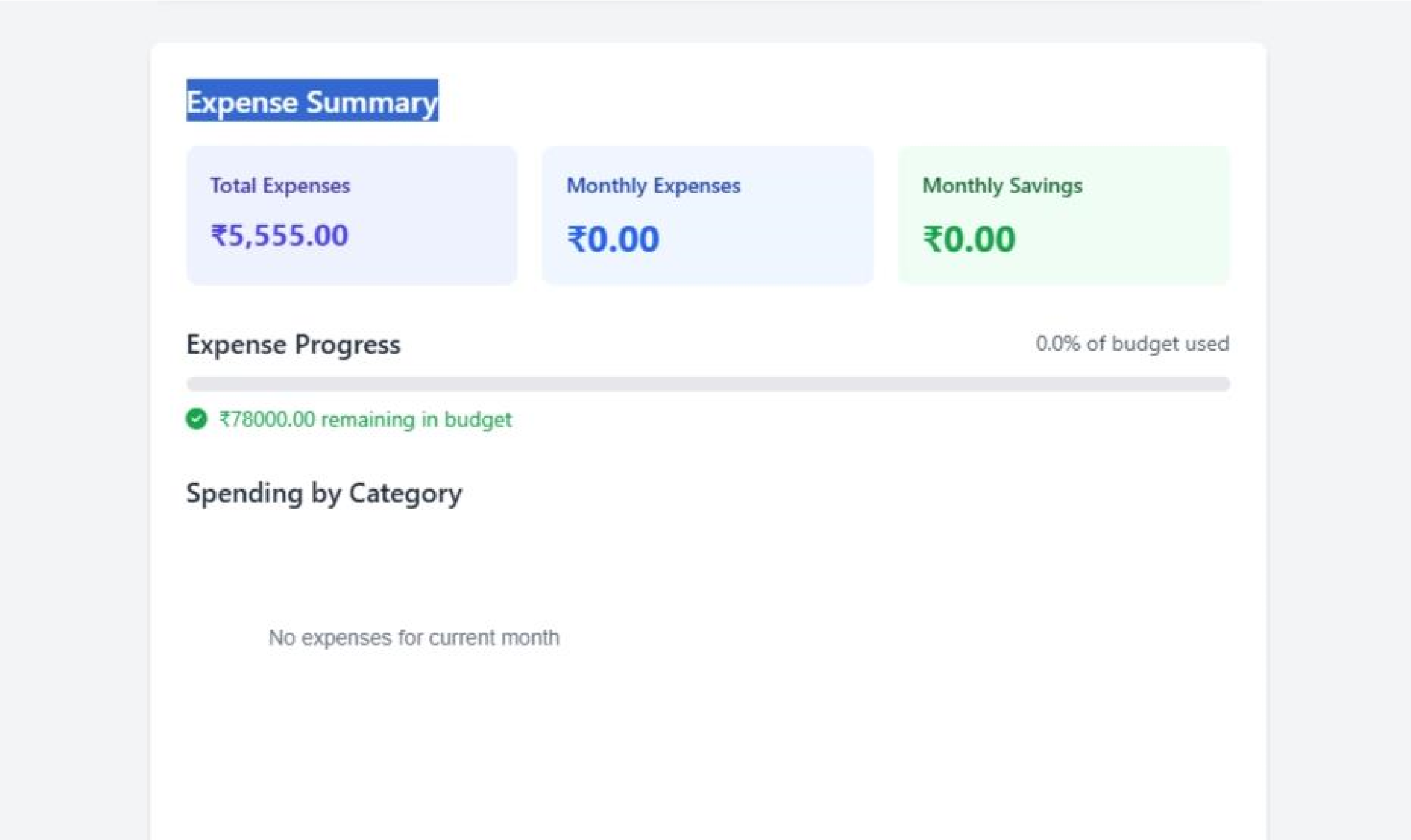


**Dashboard**

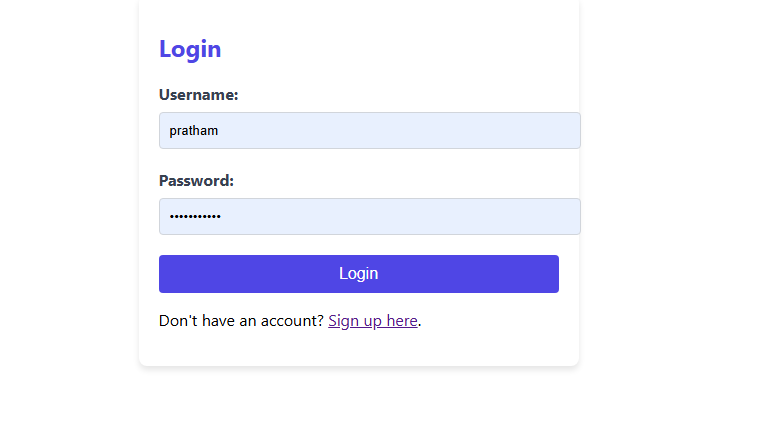


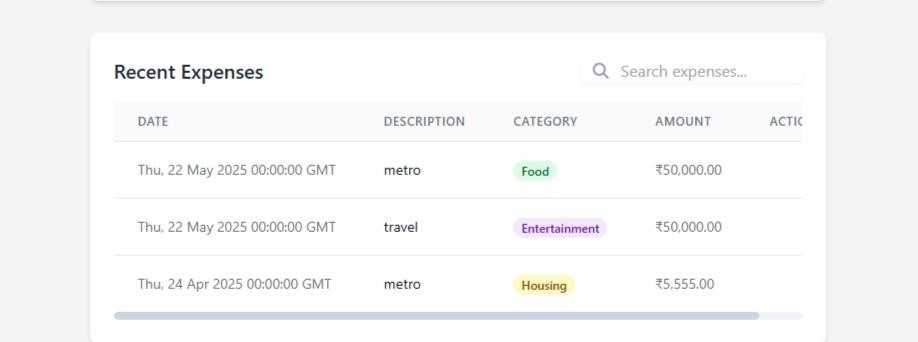
**Data Entry**

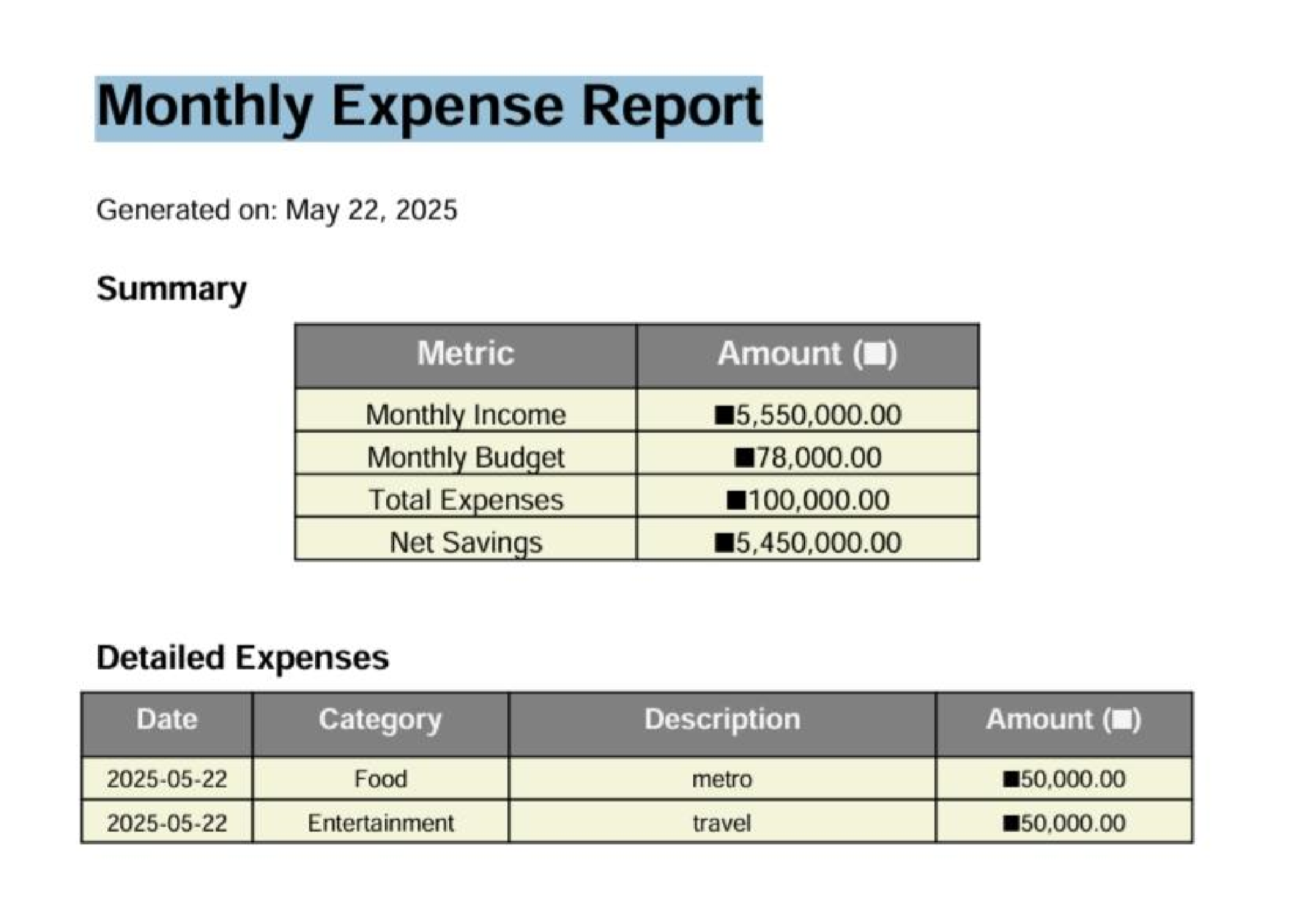




**Login Dashboard**

****





Code

Run.py

|  |
| --- |
| import os import sys import webbrowser from threading import Timer from app import app from config import Config    def open\_browser():  """Open the browser to the application URL after a short delay""" webbrowser.open('http://127.0.0.1:5000/')    if \_\_name\_\_ == "\_\_main\_\_": # Check if Flask is installed try:  from flask import Flask except ImportError:  print("Flask is not installed. Installing Flask...") os.system(f"{sys.executable} -m pip install flask")  # Set Flask environment variables os.environ['FLASK\_APP'] = 'app.py' os.environ['FLASK\_ENV'] = 'development'    # Open browser after a short delay  Timer(1.5, open\_browser).start()    # Start the Flask application print("Starting Expense Tracker application...") print(f"Access the application at: http://{Config.HOST}:{Config.PORT}/") print("Press CTRL+C to stop the server.")    # Run the Flask application app.run( host=Config.HOST, port=Config.PORT, debug=Config.DEBUG  ) |

Requirements

Flask==3.0.2 python-dotenv==1.0.1 mysql-connector-python==8.3.0 pandas==2.2.1

|  |
| --- |
| openpyxl==3.1.2 reportlab==4.1.0 |

Init\_db.py

|  |
| --- |
| import mysql.connector from config import Config    def init\_database(): try:  # Connect to MySQL server conn = mysql.connector.connect( host=Config.MYSQL\_HOST, user=Config.MYSQL\_USER, password=Config.MYSQL\_PASSWORD, auth\_plugin='mysql\_native\_password'  )  cursor = conn.cursor()    # Create database if it doesn't exist cursor.execute(f"CREATE DATABASE IF NOT EXISTS {Config.MYSQL\_DB}") cursor.execute(f"USE {Config.MYSQL\_DB}")    # Drop existing tables if they exist cursor.execute("DROP TABLE IF EXISTS expenses") cursor.execute("DROP TABLE IF EXISTS settings")    # Create expenses table cursor.execute('''  CREATE TABLE expenses ( id INT AUTO\_INCREMENT PRIMARY KEY, amount DECIMAL(10,2) NOT NULL, category VARCHAR(50) NOT NULL, description TEXT, date DATE NOT NULL  )  ''')    # Create settings table cursor.execute('''  CREATE TABLE settings ( setting\_key VARCHAR(50) PRIMARY KEY, value TEXT NOT NULL  ) |

''')

# Insert default settings cursor.execute('''

|  |
| --- |
| INSERT INTO settings (setting\_key, value)  VALUES  ('monthly\_income', '0'),  ('monthly\_expense', '0')  ''')    conn.commit() print("Database initialized successfully!")  except mysql.connector.Error as err:  print(f"Error initializing database: {err}") finally: if 'conn' in locals():  cursor.close() conn.close()    if \_\_name\_\_ == '\_\_main\_\_':  init\_database() |

Html

table.py

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>  <title>Product Table</title>  <link rel="stylesheet" href="styles.css"/>  </head>  <body>  <div class="table-container">  <h1>Product List</h1>  <table>  <thead>  <tr>  <th>#</th>  <th>Product Name</th>  <th>Category</th>  <th>Price ($)</th>  <th>Stock</th>  <th>Description</th>  </tr> |

|  |
| --- |
| ergonomic design</td></tr>  <tr><td>2</td><td>Bluetooth  Speaker</td><td>Audio</td><td>45</td><td>60</td><td>Portable with powerful bass</td></tr>  <tr><td>3</td><td>Smart  Watch</td><td>Wearables</td><td>150</td><td>80</td><td>Fitness and health tracking</td></tr>  <tr><td>4</td><td>USB-C  Hub</td><td>Accessories</td><td>30</td><td>50</td><td>Multiport connectivity</td></tr>  <tr><td>5</td><td>Gaming  Keyboard</td><td>Electronics</td><td>70</td><td>40</td><td>RGB backlight, mechanical keys</td></tr>  <tr><td>6</td><td>Laptop  Stand</td><td>Office</td><td>35</td><td>100</td><td>Adjustable aluminum stand</td></tr>  <tr><td>7</td><td>External Hard  Drive</td><td>Storage</td><td>85</td><td>75</td><td>1TB USB 3.0 drive</td></tr>  <tr><td>8</td><td>LED  Monitor</td><td>Electronics</td><td>180</td><td>30</td><td>24-inch full HD display</td></tr>  <tr><td>9</td><td>Noise Cancelling  Headphones</td><td>Audio</td><td>120</td><td>25</td><td>Wireless over-ear design</td></tr>  <tr><td>10</td><td>Portable  Charger</td><td>Power</td><td>40</td><td>150</td><td>10000mAh fast charging</td></tr>  </tbody>  </table>  </div>  </body>  </html> |

Report

The **Expense Tracker System** includes a powerful reporting module that helps users gain a clear understanding of their financial habits. Through various types of reports, the system provides both summary and detailed insights into income and spending patterns. Users can view a **monthly summary report**, which displays the total income, total expenses, and the remaining balance or savings for the selected month. This allows them to quickly evaluate their financial health at a glance.

A **category-wise expense report** breaks down spending into specific categories such as Food, Travel, Utilities, and more. This makes it easier for users to identify where most of their money is going and adjust their spending accordingly. Additionally, the **budget vs. actual report** compares what the user planned to spend in each category with what they actually spent. If any category exceeds the set limit, the system notifies the user, helping them stay within their financial goals.

Future scope

The **Expense Tracker System** is designed to offer a wide range of features that make personal finance management simple, efficient, and user-friendly. At its core, the system allows users to add, view, edit, and delete their income and expense records. Each transaction can be categorized to help users better understand where their money is coming from and where it is going. Users can set monthly budgets for different categories and receive alerts when their spending exceeds the defined limits.

**Conclusion**

The Expense Tracker System has been successfully developed to help users manage and monitor their daily financial activities efficiently. This system provides a user-friendly interface that allows individuals to log, categorize, and analyze their expenses and income. By automating the tracking process, it reduces manual effort and improves accuracy in personal budgeting.

The integration of essential modules such as user registration, daily expense tracking, and data visualization makes the system comprehensive and easy to use. Both admin and user functionalities have been implemented to ensure proper control, data management, and accessibility.

Overall, the project has achieved its objectives of promoting financial awareness, enhancing user convenience, and demonstrating key software development practices. Future enhancements may include mobile app integration, multi-currency support, and AI-driven insights for smarter budgeting decisions.

**Bibliography**

 Pressman, R. S. (2014). *Software Engineering: A Practitioner’s Approach* (8th ed.). McGraw-Hill Education.

 Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

 Welling, L., & Thomson, L. (2009). *PHP and MySQL Web Development* (4th ed.). Addison-Wesley.

 Duckett, J. (2011). *HTML and CSS: Design and Build Websites*. Wiley.

 Stack Overflow. (n.d.). *Various posts and discussions on bug fixes and implementation*. Retrieved from <https://stackoverflow.com>

 Mozilla Developer Network (MDN). (n.d.). *Web development documentation*. Retrieved from <https://developer.mozilla.org/>

 Bootstrap Team. (n.d.). *Bootstrap Documentation*. Retrieved from <https://getbootstrap.com/>

 W3Schools. (n.d.). *HTML, CSS, JavaScript, PHP Tutorials*. Retrieved from <https://www.w3schools.com/>

 GeeksforGeeks. (n.d.). *Expense Tracker Project Tutorials and Source Code*. Retrieved from <https://www.geeksforgeeks.org/>

 GitHub. (n.d.). *Open-source Expense Tracker Repositories*. Retrieved from <https://github.com/>