`

INTRODUCTION ABOUT THE PROJECT

In today’s rapidly evolving economic landscape, financial literacy and personal wealth management have become critical skills for individuals seeking stability and growth. The rise of digital transactions has transformed how money is earned, spent, and invested, providing people with instant access to financial tools but often leading to fragmented financial data. Despite the abundance of generic budgeting apps and complex spreadsheet templates, many individuals struggle to find a centralized, user-friendly platform that offers both expense tracking and investment foresight. This is where the Personal Finance Management System comes in.

This project bridges the gap between daily expense management and long-term wealth creation by offering a comprehensive, feature-rich dashboard for individual users. Designed with a focus on data visualization, security, and usability, the platform provides a streamlined approach to monitoring financial health. Users can track daily expenses across custom categories, visualize spending trends through dynamic charts, and manage recurring payments via an interactive calendar. Unlike static trackers, this system integrates proactive tools like a Compound Interest Calculator and an Investment Advisor to help users plan for the future.

The goal of this project is to create a one-stop solution that empowers individuals to take control of their financial destiny. By leveraging data visualization libraries and robust backend logic, the platform not only provides a seamless expense logging experience but also ensures users have access to analytical insights that align with their savings goals and risk appetite.

As the necessity for personal financial planning continues to rise, this system is built to meet these growing needs. Its responsive design, intuitive dashboard, and real-time calculation capabilities make it a powerful tool for users ranging from students managing monthly allowances to professionals planning diverse investment portfolios.

From a technical perspective, the platform’s development is rooted in modern web development principles, utilizing HTML5, Jinja2 templates, Bootstrap, and Vanilla JavaScript to create a responsive and interactive frontend. The application is powered by a robust backend built with Python 3.x and the Flask framework, employing the Application Factory pattern to ensure scalability and modularity. The data layer is managed using MySQL with SQLAlchemy ORM, ensuring secure and efficient data handling. The project follows a modular development approach, utilizing Flask Blueprints to organize routing and logic, ensuring maintainability and ease of future enhancements.

By combining ease of use, performance, and analytical depth, the Personal Finance Management System seeks to stand out in the domain of personal utility software. Whether an individual is looking to cut unnecessary costs, plan for a major purchase, or understand the power of compounding, this platform offers the tools and resources needed to succeed in today’s complex financial environment.

`

OBJECTIVE AND SCOPE OF PROJECT

The Personal Finance Management System has the following objectives:

- To provide a comprehensive mechanism for tracking daily expenses across multiple customizable categories, ensuring users maintain an accurate record of their financial outflows.
- To offer visual insights into spending habits through dynamic charts and graphs, allowing users to analyze trends over time and identify areas for cost-cutting.
- To enable users to manage recurring financial obligations by setting up reminders via an interactive calendar, ensuring bills and payments are never missed.
- To empower users with financial forecasting tools, specifically a Compound Interest Calculator, to visualize the long-term growth potential of their savings.
- To provide a secure and user-friendly platform for users to create accounts, manage profiles, and store sensitive financial data with privacy ensured by authentication protocols.
- To enhance financial literacy by offering a curated "Investment Advisor" module that educates users on various investment options based on risk levels.

The scope for developing the Personal Finance Management System includes:

- Managing user authentication and session security, including registration, login, and role-based access control (User vs. Admin).
- Developing a responsive dashboard interface using Bootstrap that adapts seamlessly to desktops, tablets, and smartphones.
- Implementing complex data visualization using Chart.js to render expense

breakdowns, category distributions, and compound interest growth curves.

- Integrating client-side logic for real-time financial calculations and enabling data export functionality (e.g., downloading calculation results as CSV).
- Creating a robust backend architecture using Python (Flask) and MySQL to handle data persistence, relationship management between users and expenses, and efficient query processing.

`

INTRODUCTION OF HTML AND CSS

HTML (Hypertext Mark-up Language) and CSS (Cascading Style Sheets) are essential technologies that form the foundation of web development, enabling the creation of visually engaging and well-structured web pages. Much like Visual Basic in application development, HTML and CSS are integral in defining the structure and presentation of web content.

HTML (Hypertext Mark-up Language):

HTML serves as the fundamental framework of web pages, providing a structured mark-up language to define content and layout. Developers use HTML to create documents by utilizing tags that categorize and organize different elements on a webpage, such as headings, paragraphs, images, links, forms, and more.

CSS (Cascading Style Sheets):

CSS complements HTML by allowing developers to manage the visual styling of web pages. It introduces style rules that dictate how HTML elements should be displayed across various devices. By separating content from design, CSS streamlines the creation of consistent and visually appealing user interfaces.

Together, HTML and CSS provide the essential building blocks for crafting an engaging and user-friendly web experience. While HTML defines the structure, CSS enhances the design, offering a flexible and responsive layout. This synergy supports the principles of Rapid Application Development (RAD), simplifying the process of designing and styling web interfaces efficiently.

`

INTRODUCTION OF JAVASCRIPT

JavaScript is a versatile and powerful programming language primarily employed in web development. It is a key component of modern web browsers, empowering developers to create dynamic and interactive user interfaces. Often abbreviated as JS, JavaScript plays a pivotal role in enhancing web page functionality through client-side scripting.

Key Features of JavaScript:

Client-Side Scripting:

JavaScript is predominantly used for client-side scripting, meaning it executes directly within the user's web browser. This capability enables developers to craft dynamic content, validate forms, handle events, and manipulate the Document Object Model (DOM), allowing webpage content to be updated dynamically without the need for a page reload.

Object-Oriented:

JavaScript is an object-oriented language, supporting core concepts such as encapsulation, inheritance, and polymorphism. These features allow developers to organize and modularize their code effectively, facilitating better code management and scalability.

`

Interactivity and Dynamic Content:

JavaScript enhances web pages by enabling real-time responsiveness to user interactions. Developers can implement features like sliders, pop-ups, and form validations, while also facilitating dynamic content loading and updating, contributing to more engaging and user-friendly websites.

Event-Driven Programming:

JavaScript follows an event-driven programming model, wherein developers define functions that execute in response to specific events such as button clicks, mouseovers, or form submissions. This paradigm improves user experience by making web pages more interactive and responsive to user actions.

Cross-Browser Compatibility:

JavaScript is supported by all major web browsers, including Chrome, Firefox, Safari, and Edge, ensuring consistent functionality across diverse platforms and devices.

JavaScript is an indispensable technology in modern web development, frequently working alongside HTML and CSS to build interactive web applications. Over time, JavaScript has expanded beyond its initial use in web browsers, now also playing a significant role in server-side development (via Node.js), mobile app development, and other application domains.

`

INTRODUCTION OF PYTHON AND FLASK FRAMEWORK

Python is a high-level, interpreted, general-purpose programming language known for its emphasis on code readability and simplicity. Created by Guido van Rossum and first released in 1991, Python has grown to become one of the most popular programming languages in the world, powering everything from simple scripts to complex machine learning algorithms.

For this project, we utilize Flask, a micro web framework written in Python. Created by Armin Ronacher in 2010, Flask is classified as a "microframework" because it does not require particular tools or libraries to function. It has no default database abstraction layer, form validation, or other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Python code in a web context is executed via a WSGI (Web Server Gateway Interface) server. In our architecture, the Flask application factory (create_app) initializes the application, managing routes via Blueprints and handling database interactions through SQLAlchemy. Unlike older CGI scripts, this modern stack ensures that the application remains persistent and efficient in handling concurrent user requests.

While Python is often associated with data science and automation, its web capabilities are robust. Major platforms like Netflix, Uber, and Pinterest utilize Python for its scalability and ease of integration. The standard Python interpreter (CPython) is open-source and managed by the Python Software Foundation.

According to the TIOBE Index and GitHub Octoverse reports, Python consistently ranks as one of the top programming languages globally. Its ecosystem allows for the seamless

`

integration of mathematical calculations (essential for our Compound Interest module) and data processing, making it the ideal choice for a financial dashboard.

KEY FEATURES

- Microframework Architecture (Performance): Flask is lightweight and modular. Unlike monolithic frameworks that force a specific directory structure and dependencies, Flask allows the developer to keep the core application simple and extensible. This results in faster startup times and a codebase that is easier to navigate and maintain.
- Open Source: Both Python and Flask are open-source technologies with permissive licenses (BSD/MIT). This means the entire stack—from the interpreter to the web framework and ORM—is free to use, modify, and distribute, significantly reducing the development cost of the project.
- Readable Syntax: Python is often described as "executable pseudocode." Its syntax is clear, concise, and English-like, which reduces the cognitive load on developers. This readability ensures that the complex logic behind expense tracking and interest calculations remains understandable and maintainable.
- Jinja2 Templating (Embedded Logic): Flask utilizes the Jinja2 templating engine, which allows Python-like logic to be embedded within HTML files. This enables the server to render dynamic content—such as user-specific dashboards and expense tables—securely and efficiently before sending the final HTML to the client's browser.
- Platform Independence: Python is cross-platform by design. The Personal Finance Management System can be developed on Windows and deployed on Linux (e.g., Ubuntu with Nginx/Gunicorn) without requiring code changes. This portability

`

ensures flexibility in hosting and deployment options.

- Database Support (SQLAlchemy): This project uses Flask-SQLAlchemy, an Object Relational Mapper (ORM) that provides a high-level abstraction for database interactions. It supports a wide range of databases, including the MySQL database used in our production environment. The ORM allows us to interact with the database using Python classes and objects rather than raw SQL queries, improving security and maintainability.
- Robust Error Reporting: Flask comes with a built-in debugger and logger. During development, the Werkzeug debugger provides interactive tracebacks in the browser if a crash occurs, allowing for rapid identification and resolution of bugs within the expense logic or routing.
- Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. This allows for rapid prototyping and flexibility when handling various data inputs, such as different currency formats or user profile details, without the strict boilerplate code required by statically typed languages.
- Security: Security is a core component of the Flask ecosystem. Our project leverages:
- Werkzeug: For secure password hashing (PBKDF2/Bcrypt).
- Flask-Login: For managing user sessions securely.
- Jinja2 Auto-escaping: To automatically protect against Cross-Site Scripting (XSS) attacks by escaping untrusted input in templates.
- Extensibility (Control): Flask gives developers granular control over the application. Features are added only when needed. For instance, we explicitly added Flask-Migrate for database schema management and Flask-Login for authentication. This "opt-in" approach prevents bloat and ensures the application remains optimized for its specific financial tracking purpose.
- Thriving Community: The Python and Flask communities are vast and active.

Extensive documentation, third-party extensions (like Flask-WTF for forms), and community support ensure that any technical challenge encountered during the development of the Personal Finance Management System can be resolved quickly.

`

INTRODUCTION OF MYSQL

What is a Database?

A database is a specialized application designed to store, manage, and organize data in a structured manner. It provides distinct APIs that allow users to create, access, manage, search, and replicate the data it contains. While other types of data storage systems, such as file systems or large hash tables in memory, can be used, they generally do not offer the same speed and ease of data retrieval and manipulation as a dedicated database system.

In modern applications, relational database management systems (RDBMS) are commonly used to handle large volumes of data. The term "relational" refers to the way data is organized into tables, where relationships between the data are established using primary keys and foreign keys.

Relational Database Management System (RDBMS)

An RDBMS is software that facilitates the implementation and management of relational databases. It enables the creation of databases with tables, columns, and indexes while ensuring the following functions:

- **Table Structure:** Allows the creation and organization of tables to store data.
- **Referential Integrity:** Maintains the integrity of relationships between tables using primary and foreign keys.

Index Management: Automatically updates indexes to optimize data retrieval.

- **SQL Query Processing:** Interprets SQL queries and retrieves data by combining information from various tables.

RDBMS Terminology

Before exploring specific database systems like MySQL, it is important to understand some key terminology used in RDBMS:

- **Database:** A collection of related tables that store data.
- **Table:** A table is a collection of rows and columns, representing data in a structured format similar to a spreadsheet.
- **Column:** A column contains data of a specific type or category, such as customer names, addresses, or product prices.
- **Row:** A row (also known as a tuple, entry, or record) is a set of related data within a table, often representing an individual record.
- **Redundancy:** The practice of storing data multiple times to improve system performance. While redundancy can speed up access, it may introduce challenges in data consistency.
- **Primary Key:** A primary key is a unique identifier for each record in a table. It ensures that each row in a table is distinguishable from others, and no two rows can have the same primary key value.
- **Foreign Key:** A foreign key is used to create a relationship between two tables. It refers to the primary key of another table, ensuring referential integrity between the

`

two tables.

- **Compound Key (Composite Key):** A compound key is a primary key that consists of two or more columns, used when a single column cannot uniquely identify records in a table.

- **Index:** An index in a database is a data structure that allows for faster retrieval of data, similar to an index at the back of a book. It speeds up query execution by reducing the amount of data that needs to be scanned.

- **Referential Integrity:** Referential integrity ensures that foreign key values in a table always point to valid, existing rows in another table, preventing orphaned records and maintaining data consistency.

What is SQL?

SQL (Structured Query Language) is the standard language used for managing and manipulating data in a relational database management system (RDBMS). It provides an interface for accessing, updating, deleting, and managing database structures and data. SQL is the foundation of interacting with databases, enabling users to perform a wide range of

operations on the data they store.

SQL is divided into four main subsets:

1. **DDL (Data Definition Language):** DDL is used to define and manage the structure of a database. It allows the creation, alteration, and deletion of database objects such as tables, views, and schemas. Key commands include:

`

- **CREATE**: To create new database objects.
- **ALTER**: To modify existing database objects.
- **DROP**: To delete database objects.

2. **DML (Data Manipulation Language):** DML deals with the manipulation of data within the database. It allows for adding, updating, deleting, and querying data. Common DML commands include:
   - **SELECT**: To retrieve data from one or more tables.
   - **INSERT**: To add new data into a table.
   - **UPDATE**: To modify existing data.
   - **DELETE**: To remove data from a table.
3. **DCL (Data Control Language):** DCL is used to control access to data within the database. It helps manage user permissions, allowing for secure access and data management. Main DCL commands include:
   - **GRANT**: To give a user specific access rights to database objects.
   - **REVOKE**: To remove previously granted permissions.
4. **TCL (Transaction Control Language):** TCL is used to manage transactions in a database. A transaction is a sequence of operations performed as a single unit. TCL ensures that transactions are handled properly, preserving data integrity. Key TCL commands include:

`

- **COMMIT**: To save all changes made during the transaction.
- **ROLLBACK**: To undo changes made during the transaction if an error occurs.
- **SAVEPOINT**: To set a point within a transaction to which you can roll back.
- **SET TRANSACTION**: To configure transaction properties.

MySQL

MySQL is a widely used relational database management system (RDBMS) developed and maintained by MySQL AB, a Swedish company. It is known for being fast, reliable, and easy to use, making it a popular choice for both small businesses and large enterprises. MySQL has several advantages that contribute to its growing popularity:

- **Open Source:** MySQL is released under an open-source license, meaning it is free to use and modify.
- **Powerful:** Despite being free, MySQL provides a range of features that rival some of the most expensive commercial database management systems. It can handle substantial databases and a variety of data processing tasks.
- **Standardized SQL Support:** MySQL uses a standard form of SQL, making it compatible with most database applications.
- **Cross-Platform Compatibility:** MySQL can run on various operating systems,

including Windows, Linux, and macOS, and supports multiple programming languages like PHP, PERL, C, C++, and Java.

- **Speed:** MySQL is known for its quick performance, especially with large datasets. It is optimized for high-speed operations.

- **Scalability:** MySQL can handle large databases with ease. It supports up to 50 million rows per table, with the theoretical file size limit reaching 8 million terabytes.

- **Customization:** Being open source, MySQL can be customized and extended to fit specific needs. Developers can modify the software to suit their particular environment or requirements.

`

DEFINITION OF PROBLEM FOR DISHA FINANCE

-

The Personal Finance Management System faces several challenges related to its functionality, user experience, and backend operations. These problems span across multiple areas such as data visualization accuracy, calculation precision, security, and user retention, all of which impact the platform's overall effectiveness for personal wealth management. Below are the primary problems:

1. **User Experience and Interface:**
   - Ensuring intuitive navigation for users who may not be financially savvy.
   - Addressing cluttered or complex dashboard elements (charts/tables) that could hinder users from efficiently understanding their financial health.
2. **Data Visualization and Analysis:**
   - Ensuring accurate, real-time rendering of complex charts (e.g., expense trends, compound interest curves) based on user input.
   - Optimizing filtering mechanisms to allow users to easily refine expense history by date range, category, or amount.
3. **Data Entry and Integrity:**
   - Minimizing user error during manual data entry (e.g., incorrect dates or negative amounts) to avoid skewed financial reports.
   - Implementing a streamlined process for users to edit or delete erroneous entries without breaking related data constraints.
4. **Database Management and Performance:**
   -

- Ensuring that the database can handle a growing volume of expense records and recurring reminder data without compromising performance.
- Optimizing SQLAlchemy queries to avoid slowdowns and maintain quick dashboard load times as the user's history grows over years.

5. **Security and Privacy:**
    - Protecting highly sensitive data like income details, spending habits, and personal profile information.
    - Ensuring secure session management to prevent unauthorized access to a user’s financial dashboard.

6. **Calculation Accuracy:**
    - Addressing potential floating-point errors in financial calculations (e.g., compound interest projections) to ensure users receive precise figures.
    - Ensuring the "Investment Advisor" module provides logically sound recommendations based on the risk parameters defined in the system.
7. **Notification and Reminder Reliability:**
    - Streamlining the reminder system to ensure users are alerted about upcoming bills or maturity dates effectively.
    - Addressing issues where client-side reminders might be missed if the user does not log in frequently (since no background email worker is currently implemented).
8. **Mobile Compatibility:**
    - Ensuring a seamless mobile experience for users accessing the dashboard via smartphones to log expenses on the go.

    - Adapting complex data tables and charts to fit smaller screens without losing readability.

9. **System Scalability:**
    - Minimizing latency during concurrent operations, such as multiple users generating complex reports simultaneously.
    - Optimizing the Flask backend (using Gunicorn workers) to handle high traffic without blocking requests.
10. **User Education and Onboarding:**
    - Providing clear guidance on how to interpret financial metrics like "Compound Interest" or "Risk Levels" for beginners.
    - Reducing the learning curve to ensure users can utilize advanced features like the Investment Advisor immediately.

In sum, the Personal Finance Management System needs to address the above problems in a systematic and comprehensive manner to provide a smooth, secure, and effective financial planning platform. These challenges require a balanced approach to enhance user experience, streamline operations, and ensure the scalability of the platform.

`

SYSTEM ANALYSIS

System analysis for the Personal Finance Management System involves understanding the requirements of individual users seeking financial stability. The system must provide a seamless user experience while ensuring data security and efficient management of financial records. The system analysis phase includes gathering requirements through analyzing common financial pitfalls, studying existing budgeting tools, and identifying key functionalities that will make this project stand out. Here is the System Analysis for the Personal Finance Management System in points:

1. **Problem Identification:**
    - Modern individuals often lack a centralized view of their finances, leading to overspending and missed savings opportunities.
    - Static spreadsheets fail to provide real-time visual feedback or long-term growth projections (compound interest), leaving users reactive rather than proactive.
2. **User Requirements:**
    - **End Users:** Need a platform that provides quick expense logging, visual breakdown of spending habits, tools to calculate future wealth, and a secure environment for their data.
    - **Administrators:** Require features for managing global content (like Investment Options), monitoring user growth, and ensuring system health.
3. **System Requirements:**
    - **Functional:**
        - User authentication (Registration/Login) and profile management.
        - Expense CRUD (Create, Read, Update, Delete) functionalities with categorization.
        - Interactive Dashboard with Charts (Bar, Doughnut, Line) for data visualization.
        - Compound Interest Calculator and Investment Advisor modules.

        - Reminder system for tracking upcoming payments.

    - **Non-Functional:**
        - **Scalability:** Ability to handle increasing amounts of transaction data per user.
        - **Security:** High security to protect sensitive financial data using hashing and session protections.
        - **Performance:** Optimization to ensure instant feedback when adding expenses or calculating interest.

4. **System Architecture:**
    - The system is based on the **Model-View-Template (MVT)** architecture (standard for Flask), separating the business logic from the user interface for better maintainability.
    - The **Frontend** is developed using **HTML5, Jinja2 Templates, Bootstrap, and Vanilla JavaScript**, providing a responsive and interactive experience.
    - The **Backend** is developed using **Python (Flask) and MySQL (via SQLAlchemy)**, enabling secure data storage, complex querying, and reliable logic execution.
5. **Data Flow:**
    - The system manages user input (expense entries, calculator parameters) through secure forms and AJAX requests.
    - Flask routes process this data, validate it, and interact with the **MySQL database** via **SQLAlchemy models**.
    - JSON responses are sent back to the frontend to update charts dynamically without full page reloads.
6. **User Interface and Experience:**
    - The system emphasizes ease of use, with a dashboard-centric design, sidebar navigation, and minimal steps to log a transaction.
    - **AJAX** is used extensively to update charts and tables asynchronously, providing a smoother application-like experience.

`

7. **Security Considerations:**
   - The system protects user data with **Werkzeug** security helpers for password hashing.
   - **Flask-Login** manages user sessions to ensure authenticated access only.
   - Input validation is implemented at both the client-side (HTML5) and server-side (Python) to prevent data corruption.

`

SYSTEM REQUIREMENTS

Minimum Hardware Requirements for Development

- **CPU:** Intel Core i3 (10th Gen) or AMD Ryzen 3 equivalent (Modern multi-core processors are recommended for running the Python interpreter and database services simultaneously).
- **Memory (RAM):** 8 GB (4 GB is the absolute minimum, but 8 GB is recommended to run Visual Studio Code, a MySQL server, and a web browser smoothly).
- **Hard Disk:** 256 GB SSD (Solid State Drive is highly recommended over HDD for faster project indexing and local server startup).
- **System Type:** 64-bit Operating System (Required for modern Python versions and efficient memory management).

Minimum Software Requirements for Development

- **Operating System:**
  - **Windows:** Windows 10 or 11 (64-bit).
  - **Linux:** Ubuntu 20.04 LTS or later (Preferred for deployment simulation).
  - **macOS:** macOS Catalina or later.
- **Database:** MySQL 8.0 Community Server (or MariaDB equivalent).
- **Web Server:**
  - **Development:** Werkzeug (Built-in Flask development server).
  - **Production Readiness:** Gunicorn (WSGI Server) behind Nginx.
- **Programming Language:** Python 3.10 or later.
- **Frontend Technologies:**

`

  - HTML5, CSS3 (Bootstrap 5 Framework).
  - JavaScript (ES6 Modules, Chart.js for visualization, FullCalendar.js).
- **IDE / Text Editor:** Visual Studio Code (with Python Extension) or PyCharm Community Edition.
- **Version Control:** Git (for tracking changes).
- **Browser:** Google Chrome or Mozilla Firefox (latest versions with Developer Tools).

Minimum Hardware and Software Requirements for Running the Website

On PCs (Desktops and Laptops)

- **CPU:** Intel Core i3 / AMD Ryzen 3 or equivalent (Sufficient for processing client-side JavaScript charts).
- **Memory (RAM):** 4 GB or higher (Required to handle the DOM manipulation for the dashboard and charts).
- **Hard Disk:** Not applicable (Client-side storage relies on browser cache/cookies; the application is hosted on the server).
- **Operating System:**
  - Windows: Windows 10 or later.
  - macOS: macOS Big Sur or later.
  - Linux: Any modern distribution with a GUI.
- **Web Browser:** Modern browsers supporting ES6 JavaScript and Canvas API (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge, Safari). **Internet Explorer is not supported.**

On Mobile Devices (Phones and Tablets)

- **CPU:** Quad-core processor (e.g., Snapdragon 600 series, Apple A12, or equivalent) for smooth chart rendering.

`

- **Memory (RAM):** 3 GB or higher (To ensure the dashboard remains responsive without reloading tabs).
- **Operating System:**
  - **Android:** Android 10.0 or later.
  - **iOS:** iOS 14 or later.
- **Browser:**
  - **Android:** Google Chrome (latest version).
  - **iOS:** Safari (latest version).
- **Screen Resolution:** 720x1280 pixels or higher (The Bootstrap grid system will adapt the layout, but HD resolution is recommended for reading financial tables).
- **Internet Connection:** 4G/5G or stable Wi-Fi (Required for fetching real-time data via AJAX and syncing expenses to the cloud database).

`

SYSTEM DESIGN AND CODING

INTRODUCTION: Software design is a critical phase in the software engineering process, laying the foundation for all subsequent development activities. It transcends development paradigms and application areas, serving as a common thread in the creation of every engineered system. For the **Personal Finance Management System**, the design phase was initiated after the financial tracking requirements were specified and analyzed. This step formed the core of the development phase, bridging the gap between the initial concept of "wealth management" and the final Python/Flask implementation.

The central focus of this design is **data integrity and usability**. Through the design process, the quality of the final dashboard is ensured by translating user requirements—such as accurate interest calculations and secure expense logging—into a structured framework. A robust design minimizes the risk of building unstable systems, ensuring that critical financial data is handled securely and that the system remains maintainable for future feature additions like API integrations.

In this phase, detailed representations of the MySQL database schema, Flask Blueprint structures, and frontend interactive flows were refined, reviewed, and documented. From both technical and project management perspectives, system design plays a pivotal role in guiding the engineering process toward a successful, bug-free launch.

INPUT DESIGN: Input design is the process of converting user-oriented input into a computer-readable format. As a key component of the overall system design, input design for a financial application must be executed with meticulous attention to detail. This is because the collection of input data—specifically monetary values and dates—represents the most resource-intensive aspect of the system. Effective input design focuses on optimizing the methods of input collection while ensuring the highest possible level of accuracy to prevent financial discrepancies.

The primary objectives of input design for this project are as follows:

`

1. **Cost-Effectiveness:** Create an input process that is efficient, using intuitive forms that require minimal clicks to log an expense.
2. **Accuracy:** Achieve the highest possible accuracy in input data (e.g., preventing negative values for prices) to minimize errors in financial reports.
3. **User-Friendliness:** Ensure that the input process is intuitive, utilizing date pickers and dropdowns to facilitate seamless interaction.

Input Data Considerations: When designing input data for the Personal Finance Management System, the goal is to create a process that is easy, logical, and error-free. The data entry forms (managed via HTML5 and Jinja2) are designed so that users are clear on the required fields. For example, the "Add Expense" form explicitly labels fields for Amount, Category, and Date to avoid ambiguity.

The input process involves the following stages in our context:

- **Data Recording:** Capturing raw financial data (e.g., "500" for "Groceries") from users via web forms.
- **Data Validation:** Checking the accuracy of input data against predefined criteria (e.g., ensuring the email format is valid during registration or that interest rates are numeric).
- **Data Conversion:** Converting string inputs from forms into Python storage formats (e.g., `decimal` for money, `datetime` for dates).
- **Data Transmission:** Securely transmitting data from the client browser to the Flask backend using POST requests.

Input Types in the Project: A fundamental aspect of our input design is selecting data capture methods that reduce errors.

- **External Input:** Data sourced directly from the user, such as User Registration details, Expense Entries, and Savings Goals.
- **Internal Input:** Data generated by the system, such as `created_at` timestamps for new records.

`

- **Interactive Input:** Input provided by users in real-time, such as adjusting the "Years to Grow" slider in the Compound Interest Calculator to see immediate results.

OUTPUT DESIGN: Output design refers to the process of determining how the results of processing will be communicated to the users. In a financial application, outputs play a crucial role in conveying meaningful insights, transforming raw numbers into visual trends for decision-making. Designing effective output involves organizing the presentation of information in a manner that is clear, actionable, and accessible.

The outputs of the Personal Finance Management System are categorized as follows:

1. **External Outputs:** Information presented to the user on the dashboard, such as the "Total Monthly Spend" or "Upcoming Reminders" list.
2. **Internal Outputs:** Logs generated within the system for debugging, such as database connection errors or failed login attempts.
3. **Interactive Outputs:** Dynamic charts (via Chart.js) that respond to user queries, such as filtering expenses by "Food" or "Transport" category.
4. **Turnaround Outputs:** Outputs used for further processing, such as the "Maturity Amount" calculated by the system, which helps users decide on future investment inputs.

The key to designing effective outputs is ensuring that the right financial metrics are presented in the right format. For instance, expense distribution is best shown as a **Doughnut Chart**, while spending history is best shown as a **Line Graph**. Outputs are structured to provide clear communication that facilitates financial planning.

The design of screens and interfaces plays a critical role. We utilize a **Responsive Dashboard Layout** (Sidebar + Main Content) to make outputs not only informative but also interactive. Users can navigate through tabs, request specific date ranges, and fulfill their needs by querying their financial history effectively.

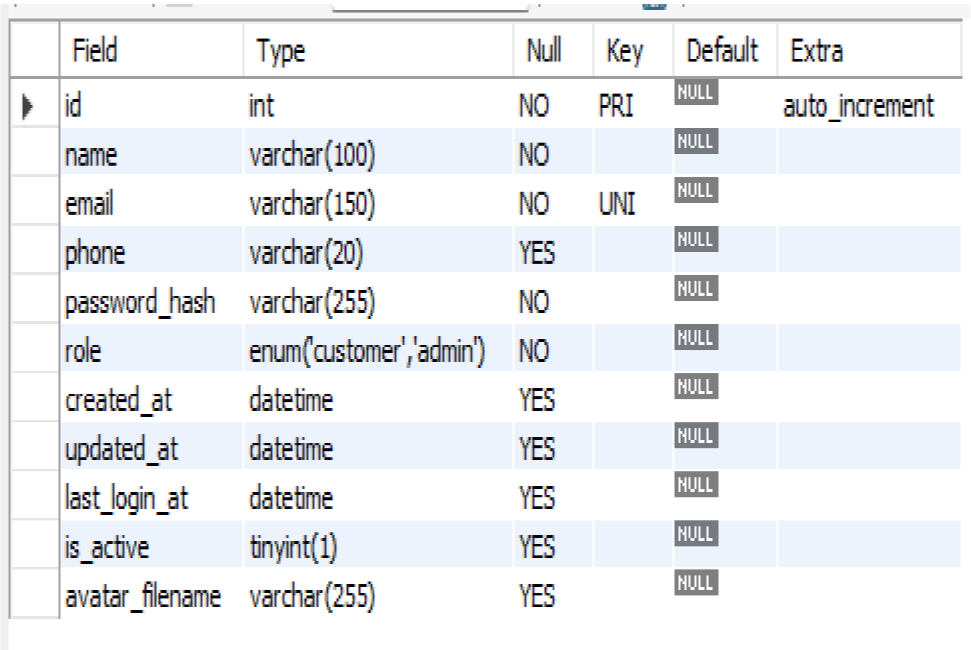
CONCLUSION: In the software engineering process, system design—including both input and output design—is foundational for building quality, reliable, and user-friendly financial tools. Input design ensures that sensitive financial data is captured accurately and efficiently, while output design guarantees that users receive the analytical insights they need in a well-structured, visual manner.

These steps are critical for the success of the Personal Finance Management System, as they directly impact both the functionality (calculation accuracy) and usability (ease of tracking) of the final product. By focusing on these design principles, we ensure that the system meets user needs, operates efficiently on the Flask/MySQL stack, and remains scalable for future enhancements.

`

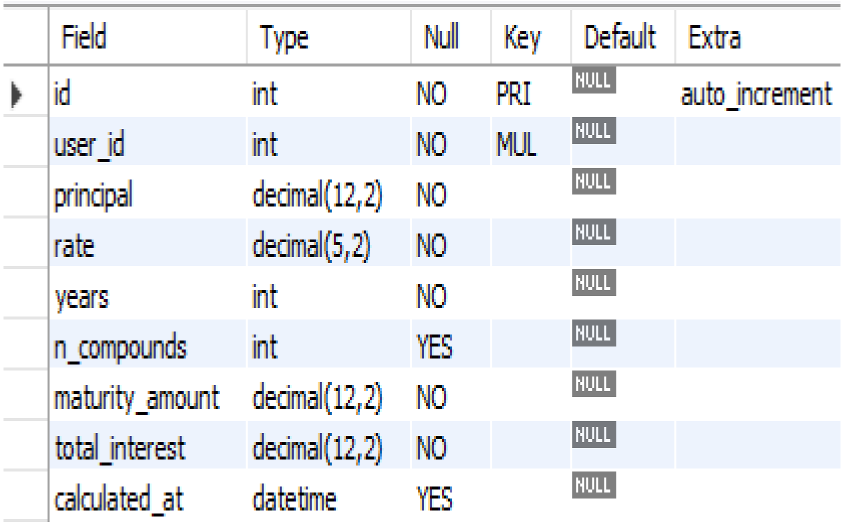
DATABASE DESIGN

TABLE 1: Users



Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(100)	NO		NULL	
email	varchar(150)	NO	UNI	NULL	
phone	varchar(20)	YES		NULL	
password_hash	varchar(255)	NO		NULL	
role	enum('customer','admin')	NO		NULL	
created_at	datetime	YES		NULL	
updated_at	datetime	YES		NULL	
last_login_at	datetime	YES		NULL	
is_active	tinyint(1)	YES		NULL	
avatar_filename	varchar(255)	YES		NULL	

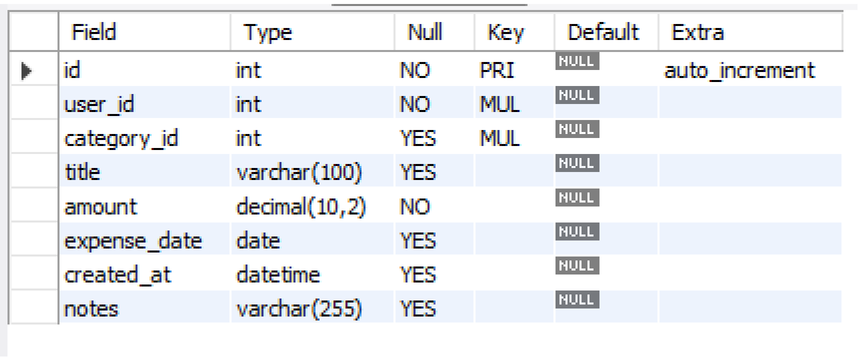
TABLE 2: Ci_Calcualtion:



Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
user_id	int	NO	MUL	NULL	
principal	decimal(12,2)	NO		NULL	
rate	decimal(5,2)	NO		NULL	
years	int	NO		NULL	
n_compounds	int	YES		NULL	
maturity_amount	decimal(12,2)	NO		NULL	
total_interest	decimal(12,2)	NO		NULL	
calculated_at	datetime	YES		NULL	

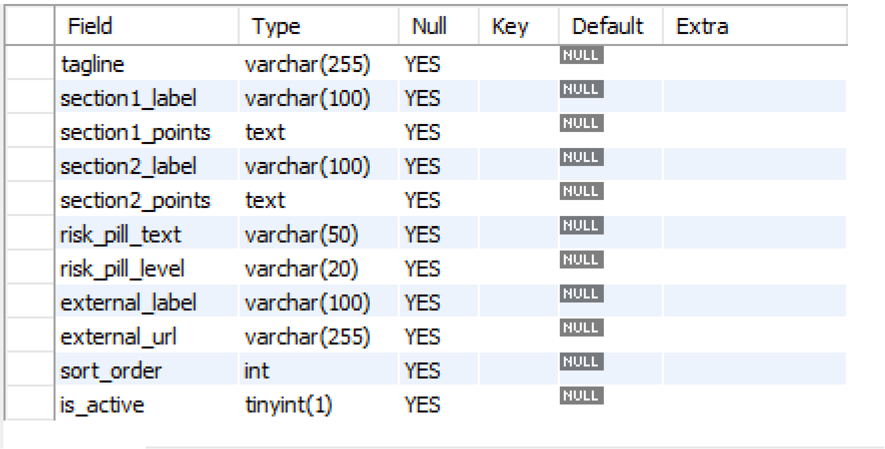
`

TABLE 3: Expenses



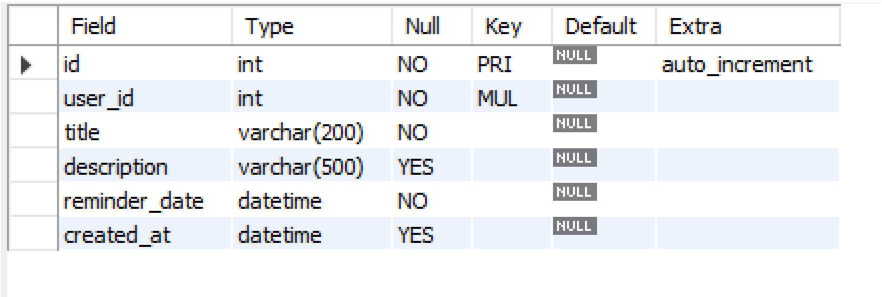
	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	user_id	int	NO	MUL	NULL	
	category_id	int	YES	MUL	NULL	
	title	varchar(100)	YES		NULL	
	amount	decimal(10,2)	NO		NULL	
	expense_date	date	YES		NULL	
	created_at	datetime	YES		NULL	
	notes	varchar(255)	YES		NULL	

TABLE 4: Investment Options:



	Field	Type	Null	Key	Default	Extra
	tagline	varchar(255)	YES		NULL	
	section1_label	varchar(100)	YES		NULL	
	section1_points	text	YES		NULL	
	section2_label	varchar(100)	YES		NULL	
	section2_points	text	YES		NULL	
	risk_pill_text	varchar(50)	YES		NULL	
	risk_pill_level	varchar(20)	YES		NULL	
	external_label	varchar(100)	YES		NULL	
	external_url	varchar(255)	YES		NULL	
	sort_order	int	YES		NULL	
	is_active	tinyint(1)	YES		NULL	

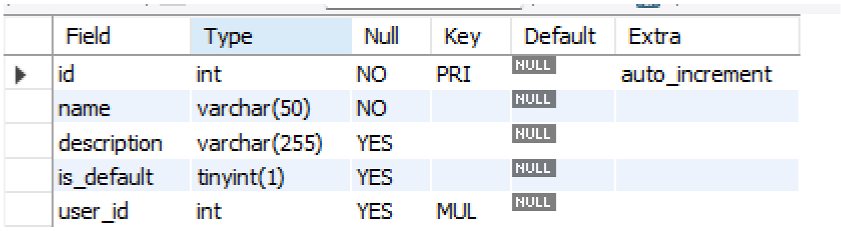
TABLE 5: Reminders



	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	user_id	int	NO	MUL	NULL	
	title	varchar(200)	NO		NULL	
	description	varchar(500)	YES		NULL	
	reminder_date	datetime	NO		NULL	
	created_at	datetime	YES		NULL	

`

TABLE 6: Expense Categories:



	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	name	varchar(50)	NO		NULL	
	description	varchar(255)	YES		NULL	
	is_default	tinyint(1)	YES		NULL	
	user_id	int	YES	MUL	NULL	

`

DATA FLOW DIAGRAM

It is a graphical tool used to describe and analyze the flow of data through a system. It focuses on the data flowing into the system, between processes, and in and out of data stores.

DFDs are of two types:

1. **Physical DFD:** The physical DFD is a model of the current system and is used to ensure that the current system has been clearly understood. It depicts *how* the system will be implemented (e.g., hardware, software, people).
2. **Logical DFD:** Logical DFDs are the model of the proposed system. They clearly show the requirements on which the new system should be built, focusing on *what* the system does rather than *how* it does it.

Notation Used In DFD:

There are four simple notations used to complete DFDs:

- **Dataflow:**

  Description: An arrow that shows the direction of data movement from one point to another.

- **External Entity:**

  Description: A square or rectangle that represents a source or destination of data outside the system (e.g., The User).

`

- **Process:**

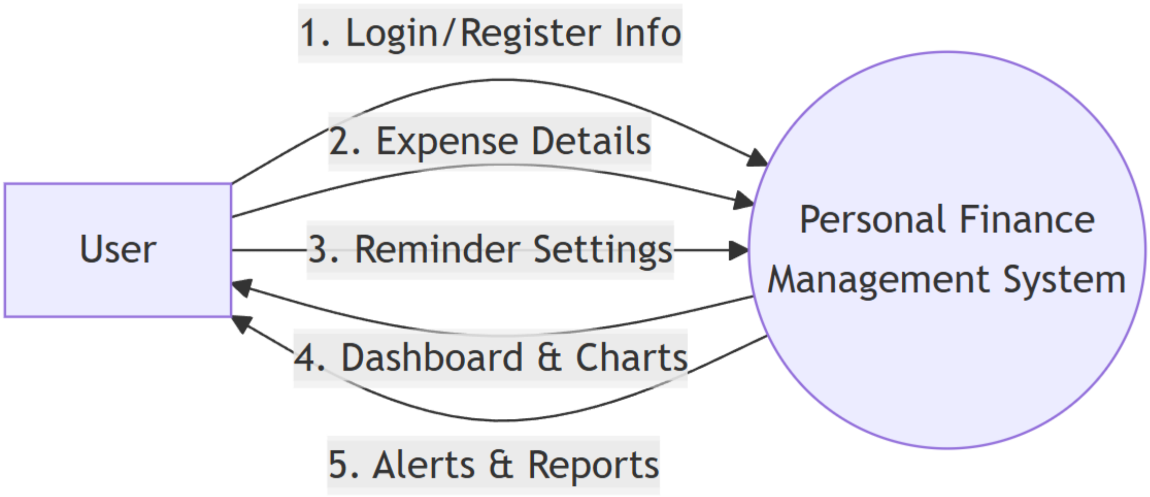
  Description: A circle or rounded rectangle that represents a transformation or manipulation of data (e.g., "Calculate Interest" or "Verify Login").

- **Data store:**

  Description: An open-ended rectangle representing a repository where data is stored for later use (e.g., The MySQL Database).

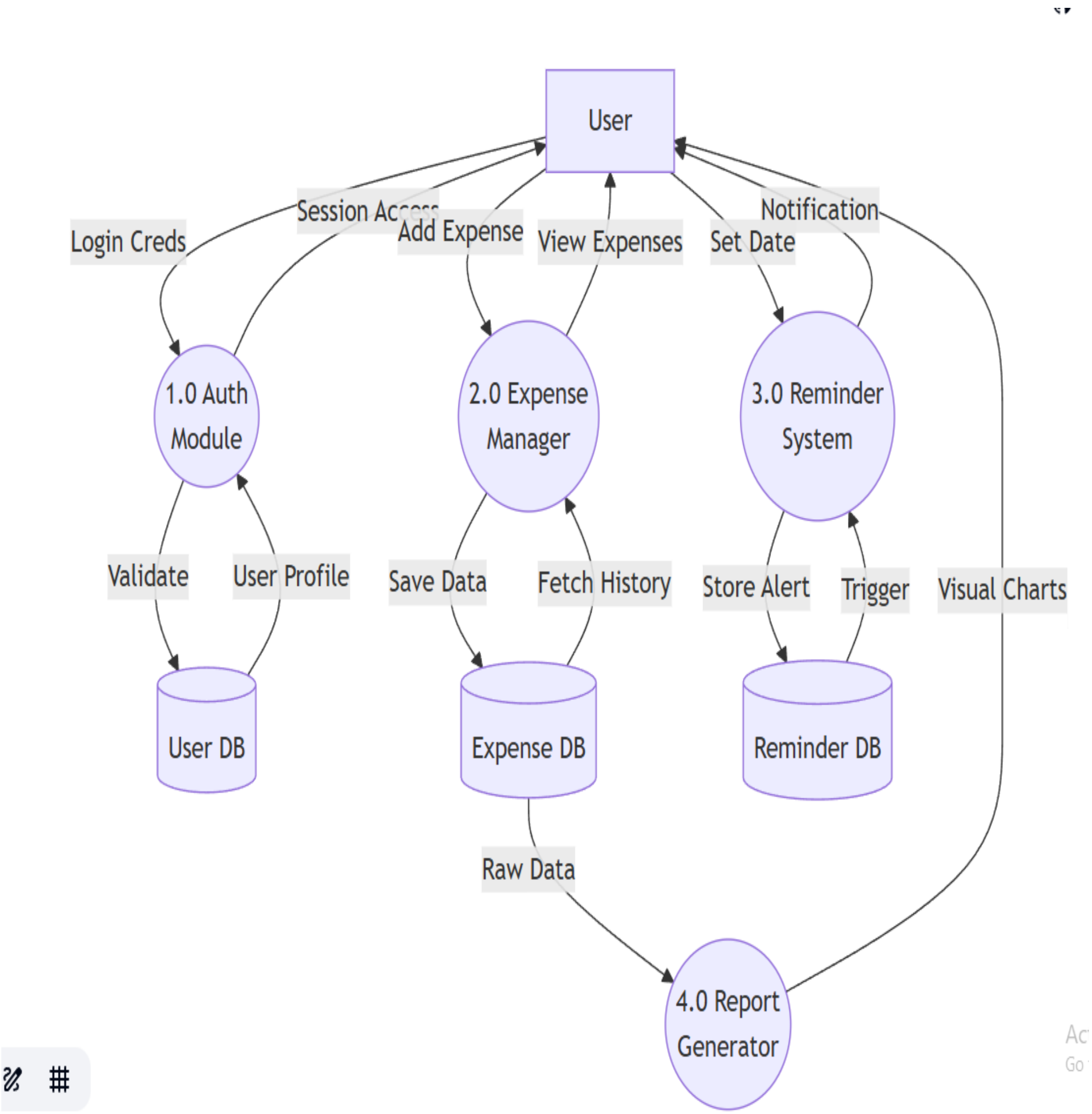
DFD FOR THE SYSTEM

Level 0 (Context Diagram):



`

Level 1 (Detailed Diagram):



`

ENTITY RELATIONSHIP DIAGRAM

Entity Relationship Diagram (ERD) can express the overall logical structure of a database graphically. It shows the relationships between different data sets.

The components of E-R Diagram are:

- **Entity:**

  Entity is a thing or object in a real world that is distinguishable from all other objects (e.g., *User*, *Expense*, *Category*).

- **Relationship:**

  It is an association among several entities (e.g., *User* **logs** *Expense*).

- **Weak Entity:**

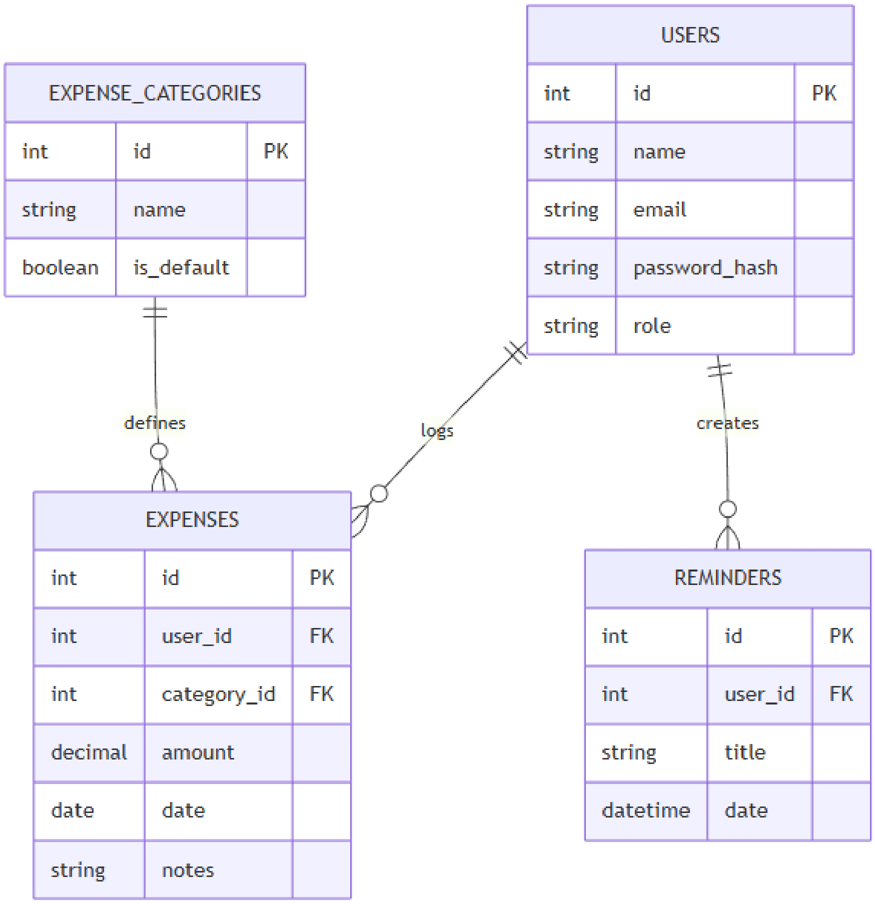
  A Weak entity is an entity that does not have any primary key of its own and depends on the existence of a strong entity (e.g., *Expense Category* might be weak if it relies entirely on a specific *User* context).

- **Attributes:**

  A property or characteristic of an entity. Often shown as an oval or circle (e.g., *Email*, *Amount, Date*).

`

ENTITY RELATIONSHIP DIAGRAM (Visual)



EXPENSE_CATEGORIES		
int	id	PK
string	name	
boolean	is_default	

USERS		
int	id	PK
string	name	
string	email	
string	password_hash	
string	role	

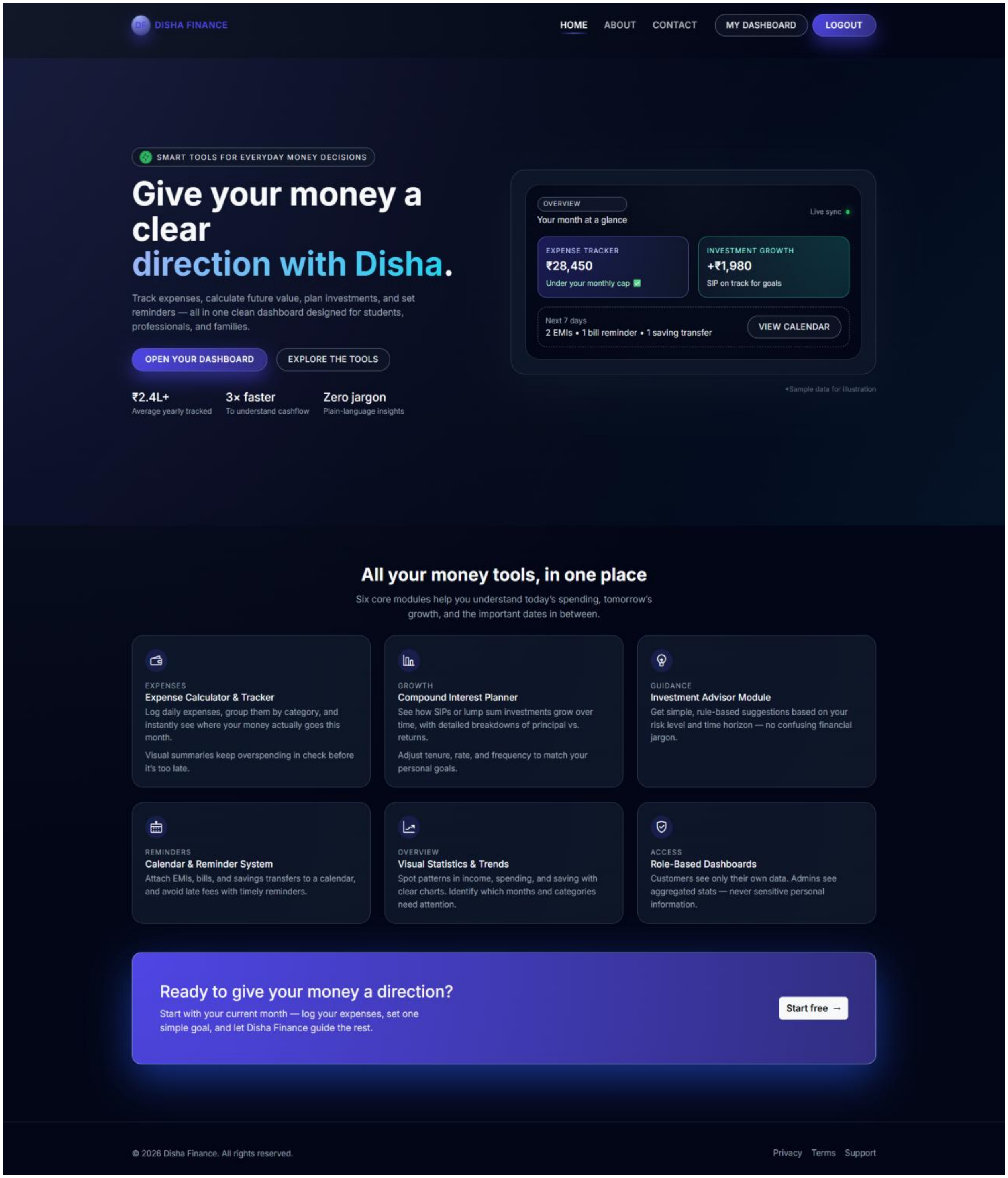
EXPENSES		
int	id	PK
int	user_id	FK
int	category_id	FK
decimal	amount	
date	date	
string	notes	

REMINDERS		
int	id	PK
int	user_id	FK
string	title	
datetime	date	

`

INPUT FORMS, PAGES AND CODING

Home Page:



SMART TOOLS FOR EVERYDAY MONEY DECISIONS

Give your money a clear direction with Disha.

Track expenses, calculate future value, plan investments, and set reminders — all in one clean dashboard designed for students, professionals, and families.

₹2.4L+
Average yearly tracked

3× faster
To understand cashflow

Zero jargon
Plain-language insights

OVERVIEW
Your month at a glance
Live sync

EXPENSE TRACKER
₹28,450
Under your monthly cap ✅

INVESTMENT GROWTH
+₹1,980
SIP on track for goals

Next 7 days
2 EMIs • 1 bill reminder • 1 saving transfer

*Sample data for illustration

All your money tools, in one place

Six core modules help you understand today's spending, tomorrow's growth, and the important dates in between.

EXPENSES
Expense Calculator & Tracker
Log daily expenses, group them by category, and instantly see where your money actually goes this month.

Visual summaries keep overspending in check before it's too late.

GROWTH
Compound Interest Planner
See how SIPs or lump sum investments grow over time, with detailed breakdowns of principal vs. returns.

Adjust tenure, rate, and frequency to match your personal goals.

GUIDANCE
Investment Advisor Module
Get simple, rule-based suggestions based on your risk level and time horizon — no confusing financial jargon.

REMINDERS
Calendar & Reminder System
Attach EMIs, bills, and savings transfers to a calendar, and avoid late fees with timely reminders.

OVERVIEW
Visual Statistics & Trends
Spot patterns in income, spending, and saving with clear charts. Identify which months and categories need attention.

ACCESS
Role-Based Dashboards
Customers see only their own data. Admins see aggregated stats — never sensitive personal information.

Ready to give your money a direction?

Start with your current month — log your expenses, set one simple goal, and let Disha Finance guide the rest.

© 2026 Disha Finance. All rights reserved.

`

Code

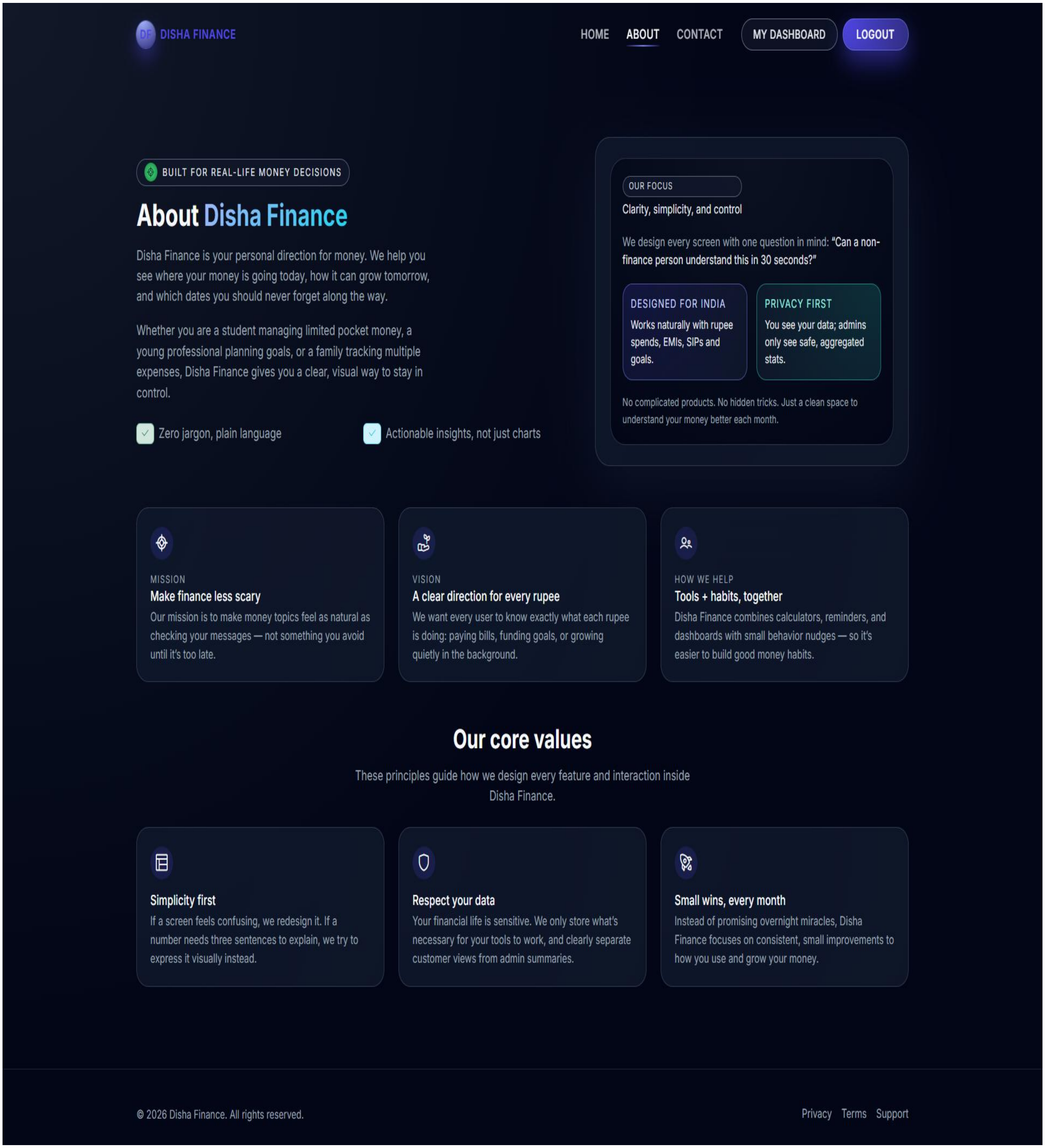
```
 {% extends "base.html" %}
{% block title %}Disha Finance – Home{% endblock %}

{% block content %}

<section class="hero-gradient">
  <div class="container position-relative">
    <div class="row align-items-center">

      <div class="col-lg-6">
        <div class="chip">
          <span><i class="lni lni-bolt"></i></span> Smart finance tools
        </div>
        <h1 class="hero-title">Give your money a clear direction.</h1>
        <p class="hero-sub">
          Track expenses, calculate future value, and set reminders —
          all in one clean dashboard.
        </p>
       <div class="d-flex gap-2 mb-4">
          <a href="{{ url_for('auth.register') }}" class="btn df-cta-btn">
            Open Dashboard
          </a>
          <a href="#features" class="btn df-cta-outline">Explore Tools</a>
        </div>
      </div>
      <div class="col-lg-6">
        <div class="hero-card">
          <div class="preview-card">
            <div class="preview-header">Overview</div>
            <div class="row g-3">
              <div class="col-6">
                <div class="widget-box">
                  <div>Expense Tracker</div>
                  <div class="amount">₹28,450</div>
                </div>
              </div>
              <div class="col-6">
                <div class="widget-box">
                  <div>Investment Growth</div>
                  <div class="amount">+₹1,980</div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>

    </div>
  </div>
</section>
```

`

```
<section class="df-section" id="features">
  <div class="container">
    <div class="text-center mb-4">
      <h2 class="df-section-title">Core Modules</h2>
    </div>

    <div class="row g-4">
      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-wallet"></i></div>
          <h5 class="feature-title">Expense Tracker</h5>
          <p>Log daily expenses and visualize categorical spending.</p>
        </div>
      </div>
      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-bar-chart"></i></div>
          <h5 class="feature-title">Compound Interest</h5>
          <p>Plan SIPs and visualize long-term wealth generation.</p>
        </div>
      </div>
      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-calendar"></i></div>
          <h5 class="feature-title">Reminders & Calendar</h5>
          <p>Never miss a bill payment or EMI date again.</p>
        </div>
      </div>
    </div>
  </div>
</section>

<section class="df-section pt-0">
  <div class="container">
    <div class="cta-card p-5 rounded-4">
      <div class="row align-items-center">
        <div class="col-md-8">
          <h3>Ready to start?</h3>
          <p class="text-light">Start with your current month and let Disha guide the rest.</p>
        </div>
        <div class="col-md-4 text-end">
          <a href="{{ url_for('auth.register') }}" class="btn btn-light">Start free →</a>
        </div>
      </div>
    </div>
  </div>
</section>

{% endblock %}:
```

`

About Page:



BUILT FOR REAL-LIFE MONEY DECISIONS

About Disha Finance

Disha Finance is your personal direction for money. We help you see where your money is going today, how it can grow tomorrow, and which dates you should never forget along the way.

Whether you are a student managing limited pocket money, a young professional planning goals, or a family tracking multiple expenses, Disha Finance gives you a clear, visual way to stay in control.

✓ Zero jargon, plain language

✓ Actionable insights, not just charts

OUR FOCUS

Clarity, simplicity, and control

We design every screen with one question in mind: **"Can a non-finance person understand this in 30 seconds?"**

DESIGNED FOR INDIA

Works naturally with rupee spends, EMIs, SIPs and goals.

PRIVACY FIRST

You see your data; admins only see safe, aggregated stats.

No complicated products. No hidden tricks. Just a clean space to understand your money better each month.

MISSION

Make finance less scary

Our mission is to make money topics feel as natural as checking your messages — not something you avoid until it's too late.

VISION

A clear direction for every rupee

We want every user to know exactly what each rupee is doing: paying bills, funding goals, or growing quietly in the background.

HOW WE HELP

Tools + habits, together

Disha Finance combines calculators, reminders, and dashboards with small behavior nudges — so it's easier to build good money habits.

Our core values

These principles guide how we design every feature and interaction inside Disha Finance.

Simplicity first

If a screen feels confusing, we redesign it. If a number needs three sentences to explain, we try to express it visually instead.

Respect your data

Your financial life is sensitive. We only store what's necessary for your tools to work, and clearly separate customer views from admin summaries.

Small wins, every month

Instead of promising overnight miracles, Disha Finance focuses on consistent, small improvements to how you use and grow your money.

© 2026 Disha Finance. All rights reserved.

`

Code:

```
{% extends "base.html" %}
{% block title %}About – Disha Finance{% endblock %}

{% block content %}

<section class="df-section">
  <div class="container">
    <div class="row align-items-center gy-4">

      <div class="col-lg-7">
        <div class="mb-3">
          <div class="chip">
            <span><i class="lni lni-target"></i></span> Built for real life
          </div>
        </div>

        <h1 class="hero-title mb-3">About <span>Disha Finance</span></h1>

        <p class="hero-sub mb-3">
          Disha Finance is your personal direction for money. We help you see
          where your money is going today and how it can grow tomorrow.
        </p>

        <div class="row g-3">
          <div class="col-sm-6">
            <span class="badge bg-success-subtle text-success">
              <i class="lni lni-checkmark"></i> Zero jargon
            </span>
          </div>
          <div class="col-sm-6">
            <span class="badge bg-info-subtle text-info">
              <i class="lni lni-checkmark"></i> Actionable insights
            </span>
          </div>
        </div>
      </div>

      <div class="col-lg-5">
        <div class="hero-card">
          <div class="preview-card">
            <div class="preview-header">
              <div class="tag-pill mb-1">Our focus</div>
              <div>Clarity, simplicity, and control</div>
            </div>

            <div class="row g-3 mt-2">
              <div class="col-6">
                <div class="widget-box">
                  <div class="text-uppercase">Designed for India</div>
```

`

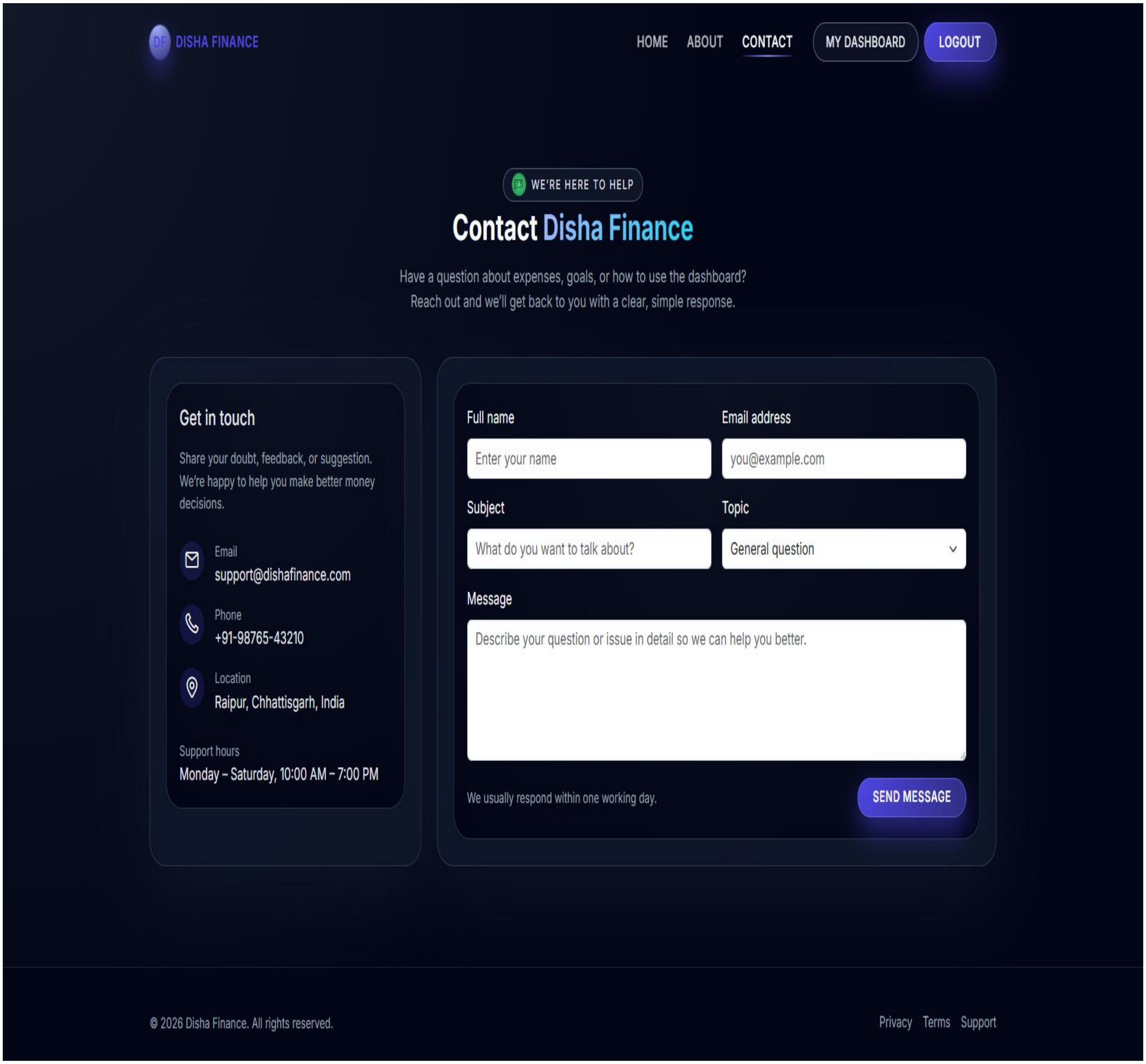
```
                  <small>Works with rupees, EMIs, and SIPs.</small>
                </div>
              </div>
              <div class="col-6">
                <div class="widget-box">
                  <div class="text-uppercase">Privacy first</div>
                  <small>Admins only see aggregated stats.</small>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>

    </div>
  </div>
</section>

<section class="df-section pt-0">
  <div class="container">
    <div class="row g-4">

      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-target"></i></div>
          <h5 class="feature-title">Mission</h5>
          <p>To make money topics feel as natural as checking your messages.</p>
        </div>
      </div>

      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-grow"></i></div>
          <h5 class="feature-title">Vision</h5>
          <p>We want every user to know exactly what each rupee is doing.</p>
        </div>
      </div>

      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-users"></i></div>
          <h5 class="feature-title">How we help</h5>
          <p>Combining calculators and dashboards with behavior nudges.</p>
        </div>
      </div>

    </div>
  </div>
</section>

<section class="df-section pt-0 pb-5">
```

`

```
  <div class="container">
    <div class="text-center mb-4">
      <h2 class="df-section-title">Our core values</h2>
    </div>

    <div class="row g-3">
      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-layout"></i></div>
          <h5 class="feature-title">Simplicity first</h5>
        </div>
      </div>
      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-shield"></i></div>
          <h5 class="feature-title">Respect your data</h5>
        </div>
      </div>
      <div class="col-md-4">
        <div class="feature-card">
          <div class="feature-icon"><i class="lni lni-rocket"></i></div>
          <h5 class="feature-title">Small wins</h5>
        </div>
      </div>
    </div>
  </div>
</section>

{% endblock %}
```

`

Contact Page:



WE'RE HERE TO HELP

Contact Disha Finance

Have a question about expenses, goals, or how to use the dashboard?
Reach out and we'll get back to you with a clear, simple response.

Get in touch

Share your doubt, feedback, or suggestion. We're happy to help you make better money decisions.

Email
support@dishafinance.com

Phone
+91-98765-43210

Location
Raipur, Chhattisgarh, India

Support hours
Monday – Saturday, 10:00 AM – 7:00 PM

Full name: Enter your name

Email address: you@example.com

Subject: What do you want to talk about?

Topic: General question

Message: Describe your question or issue in detail so we can help you better.

We usually respond within one working day.

© 2026 Disha Finance. All rights reserved.

Code:

```
{% extends "base.html" %}
{% block title %}Contact – Disha Finance{% endblock %}

{% block content %}

<section class="df-section">
  <div class="container">
```

`

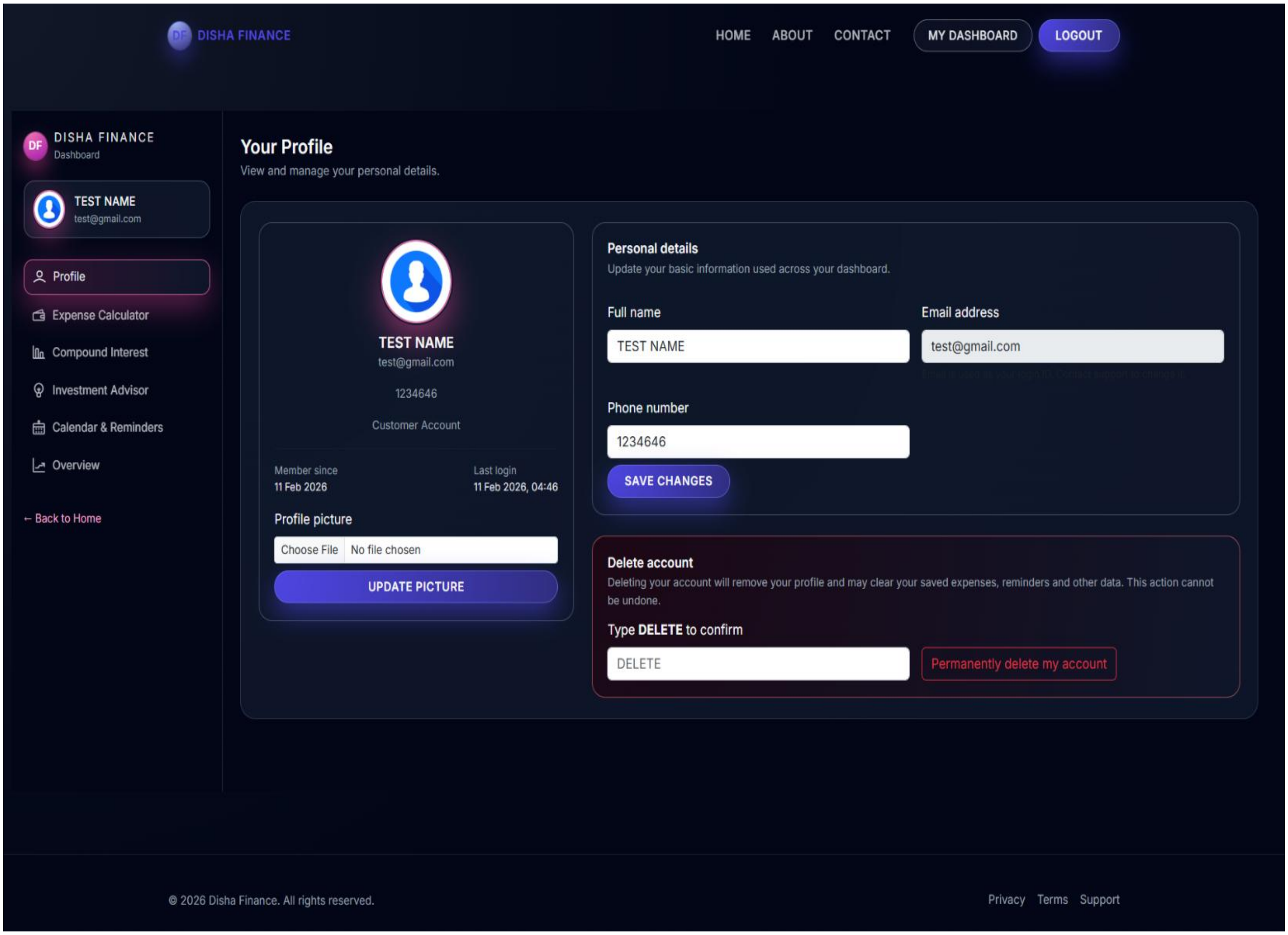
```
<div class="row justify-content-center mb-4">
  <div class="col-lg-8 text-center">
    <h1 class="hero-title">Contact Disha Finance</h1>
    <p class="df-section-sub">Have questions? Reach out to us.</p>
  </div>
</div>

<div class="row g-4">

  <div class="col-lg-4">
    <div class="hero-card h-100">
      <div class="preview-card">
        <h5 class="mb-3">Get in touch</h5>

        <ul class="list-unstyled">
          <li class="mb-3">
            <i class="lni lni-envelope"></i> support@dishafinance.com
          </li>
          <li class="mb-3">
            <i class="lni lni-phone"></i> +91-98765-43210
          </li>
          <li class="mb-3">
            <i class="lni lni-map-marker"></i> Raipur, Chhattisgarh, India
          </li>
        </ul>

        <div class="mt-4">
          <small>Hours: Mon – Sat, 10:00 AM – 7:00 PM</small>
        </div>
      </div>
    </div>
  </div>

  <div class="col-lg-8">
    <div class="hero-card h-100">
      <div class="preview-card">

        {% with messages = get_flashed_messages(with_categories=true) %}
          {% if messages %}
            {% for category, msg in messages %}
              <div class="alert alert-{{ category }}">{{ msg }}</div>
            {% endfor %}
          {% endif %}
        {% endwith %}

        <form action="{{ url_for('public.contact') }}" method="POST" class="row g-3">

          <div class="col-md-6">
            <label for="name" class="form-label">Full Name</label>
            <input type="text" name="name" class="form-control" required>
```

`

```
       </div>

       <div class="col-md-6">
        <label for="email" class="form-label">Email</label>
        <input type="email" name="email" class="form-control" required>
       </div>

       <div class="col-md-6">
        <label for="subject" class="form-label">Subject</label>
        <input type="text" name="subject" class="form-control" required>
       </div>

       <div class="col-md-6">
        <label for="topic" class="form-label">Topic</label>
        <select name="topic" class="form-select">
         <option value="general">General question</option>
         <option value="expenses">Expense tracking</option>
         <option value="bug">Report a Bug</option>
        </select>
       </div>

       <div class="col-12">
        <label for="message" class="form-label">Message</label>
        <textarea name="message" rows="5" class="form-control" required></textarea>
       </div>

       <div class="col-12 text-end">
        <button type="submit" class="btn df-cta-btn">Send Message</button>
       </div>

      </form>
      </div>
    </div>
   </div>

  </div>
 </div>
</section>

{% endblock %}
```

`

User Profile Tab:



TEST NAME
test@gmail.com

Your Profile

View and manage your personal details.

TEST NAME
test@gmail.com
1234646
Customer Account

Member since
11 Feb 2026

Last login
11 Feb 2026, 04:46

Profile picture
No file chosen

Personal details
Update your basic information used across your dashboard.

Full name: TEST NAME

Email address: test@gmail.com

Email is used as your login ID. Contact support to change it.

Phone number: 1234646

Delete account
Deleting your account will remove your profile and may clear your saved expenses, reminders and other data. This action cannot be undone.

Type **DELETE** to confirm
DELETE

© 2026 Disha Finance. All rights reserved.

Code:

```
{% extends "dashboard/layout.html" %}

{% block dashboard_title %}Your Profile{% endblock %}

{% block dashboard_content %}
<div class="row g-4">

  <div class="col-lg-4">
    <div class="card">
      <div class="card-body text-center">
        <div class="mb-3">
          <img
            src="{{ url_for('static', filename='uploads/avatars/' ~ (current_user.avatar_filename or
'default.jpg')) }}"
```

`

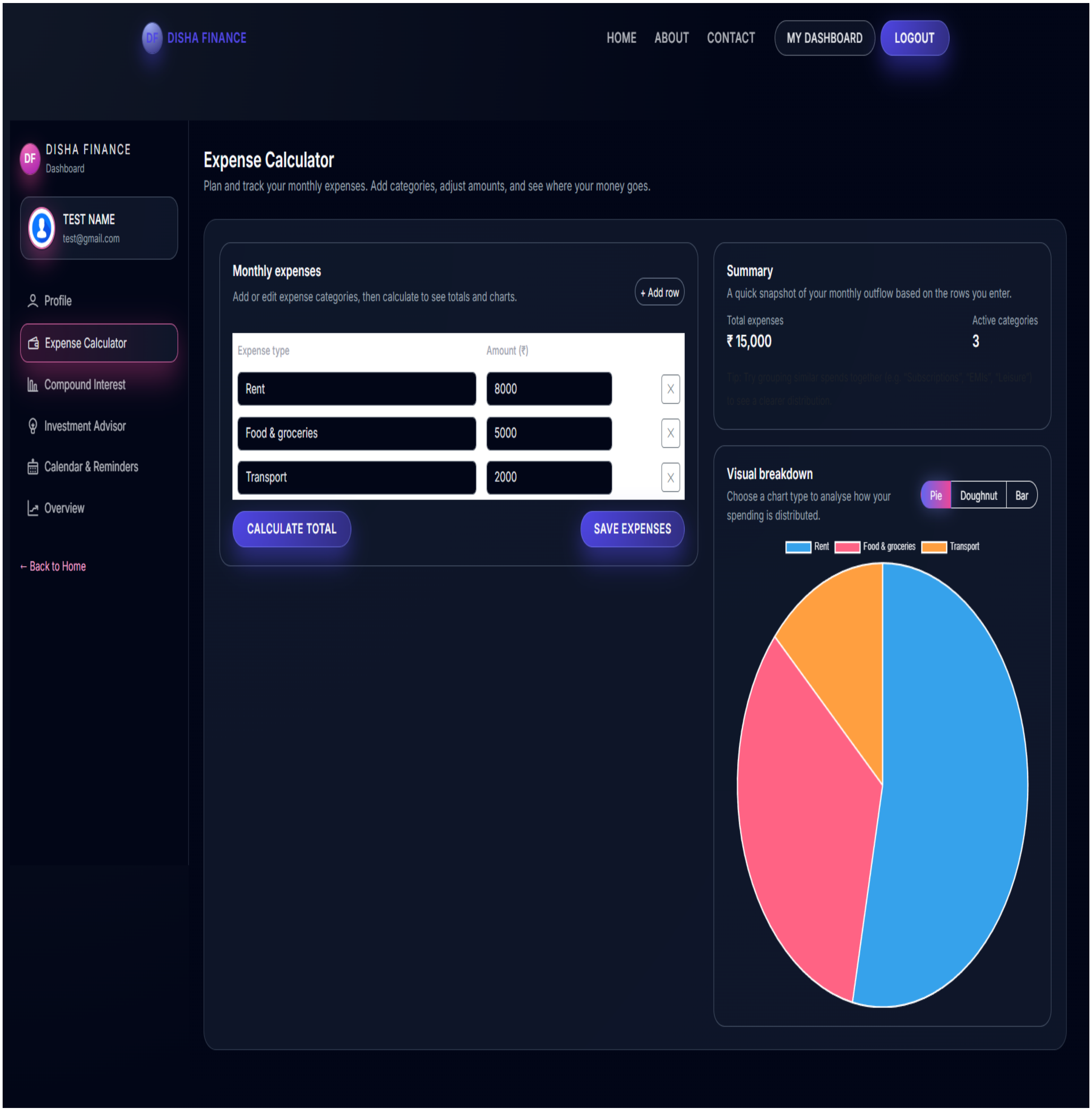
```
            alt="Profile Picture" class="rounded-circle img-fluid" style="width: 120px;"
          >
        </div>

        <h3>{{ current_user.name or 'Your Name' }}</h3>
        <p class="text-muted">{{ current_user.email }}</p>
        <span class="badge bg-secondary">{{ current_user.role|capitalize }} Account</span>

        <hr>

        <div class="text-start small">
          <div><strong>Joined:</strong> {{ current_user.created_at.strftime("%d %b %Y") }}</div>
          <div><strong>Last Login:</strong> {{ current_user.last_login_at.strftime("%d %b %Y") }}</div>
        </div>

        <form action="{{ url_for('dashboard.upload_avatar') }}" method="POST" enctype="multipart/form-data" class="mt-3">
          <label class="form-label text-start d-block">Change Picture</label>
          <input type="file" name="avatar" class="form-control form-control-sm mb-2" accept="image/*">
          <button type="submit" class="btn btn-sm df-cta-btn w-100">Update Avatar</button>
        </form>
      </div>
    </div>
  </div>

  <div class="col-lg-8">

    <div class="card mb-4">
      <div class="card-header">Personal Details</div>
      <div class="card-body">
        <form action="{{ url_for('dashboard.update_profile') }}" method="POST" class="row g-3">

          <div class="col-md-6">
            <label class="form-label">Full Name</label>
            <input type="text" name="name" class="form-control" value="{{ current_user.name }}" required>
          </div>

          <div class="col-md-6">
            <label class="form-label">Email (Read-only)</label>
            <input type="email" class="form-control" value="{{ current_user.email }}" disabled>
          </div>

          <div class="col-md-6">
            <label class="form-label">Phone</label>
            <input type="text" name="phone" class="form-control" value="{{ current_user.phone }}">
          </div>
```

`

```
          <div class="col-12">
            <button type="submit" class="btn df-cta-btn">Save Changes</button>
          </div>

        </form>
      </div>
    </div>

    <div class="card border-danger">
      <div class="card-header text-danger">Delete Account</div>
      <div class="card-body">
        <p class="small text-muted">This action is permanent and removes all your data.</p>

        <form action="{{ url_for('dashboard.delete_account') }}" method="POST" class="row
align-items-end">
          <div class="col-md-8">
            <label class="form-label">Type <strong>DELETE</strong> to confirm</label>
            <input type="text" name="confirm" class="form-control" placeholder="DELETE">
          </div>
          <div class="col-md-4">
            <button type="submit" class="btn btn-outline-danger w-100">Delete
Account</button>
          </div>
        </form>
      </div>
    </div>

  </div>
</div>
{% endblock %}
```

`

Expense Calculator Tab:



TEST NAME
test@gmail.com

Expense Calculator

Plan and track your monthly expenses. Add categories, adjust amounts, and see where your money goes.

Monthly expenses

Add or edit expense categories, then calculate to see totals and charts.

Expense type	Amount (₹)
Rent	8000
Food & groceries	5000
Transport	2000

Summary

A quick snapshot of your monthly outflow based on the rows you enter.

Total expenses
₹ 15,000

Active categories
3

Tip: Try grouping similar spends together (e.g. "Subscriptions", "EMIs", "Leisure") to see a clearer distribution.

Visual breakdown

Choose a chart type to analyse how your spending is distributed.

Code:

```
{% extends "dashboard/layout.html" %}
```

`

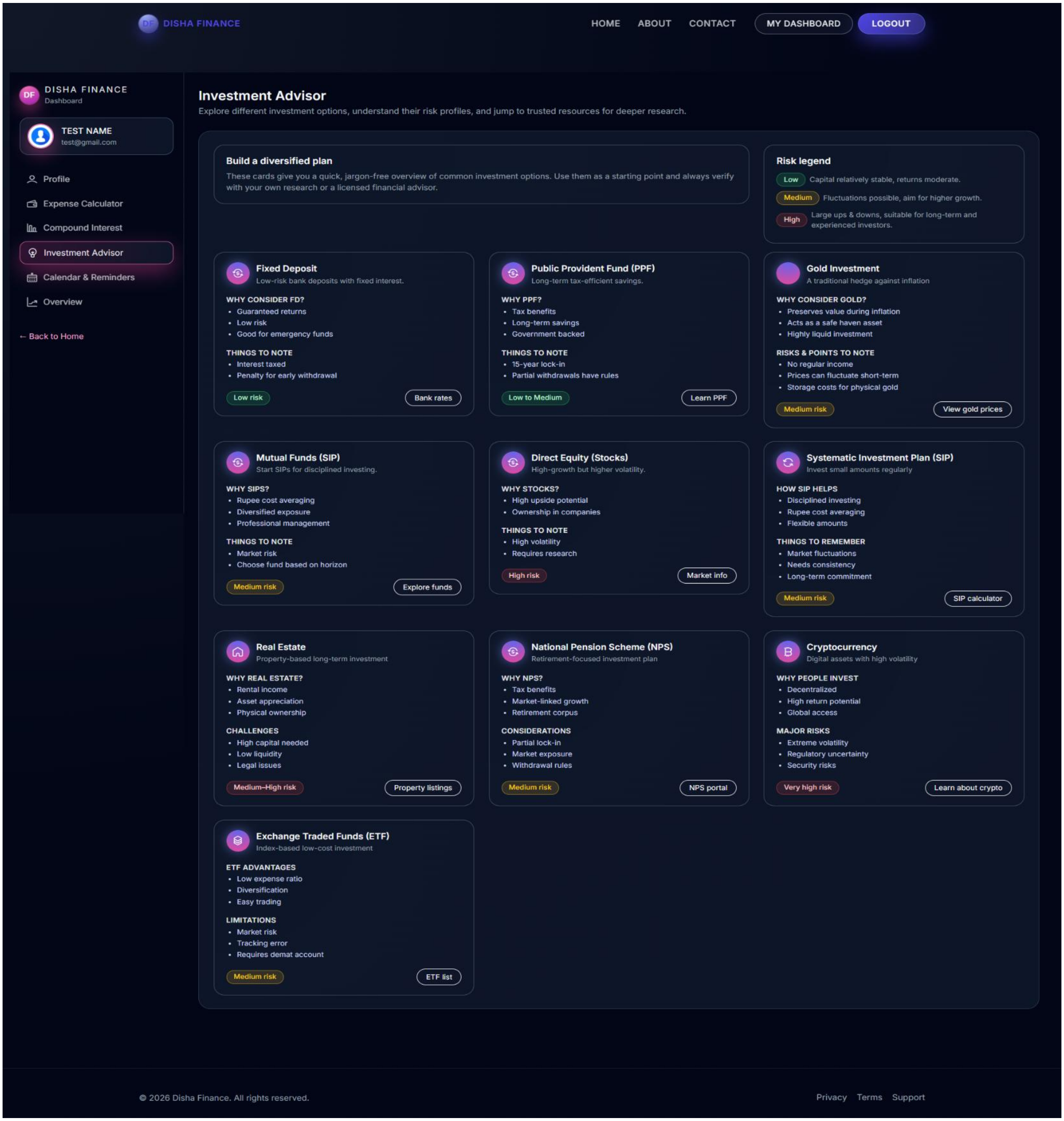
```
{% block dashboard_title %}Expense Calculator{% endblock %}
{% block dashboard_subtitle %}Plan and track your monthly expenses.{% endblock %}
{% block dashboard_content %}
<div class="row g-4">  <div lass="col-lg-7">
  <div class="card h-100">
   <div class="card-header d-flex justify-content-between align-items-center">
    <h5 class="mb-0">Monthly Expenses</h5>
    <button type="button" class="btn btn-sm btn-outline-primary df-exp-add-row">
     + Add Row
    </button>
   </div>
   <form id="df-exp-form" method="POST" action="{{ url_for('dashboard.save_expenses')
}}">
    <div class="card-body p-0">
     <table class="table table-borderless align-middle mb-0">
      <thead>
       <tr>
        <th width="55%">Category</th>
        <th width="30%">Amount (₹)</th>
        <th width="15%"></th>
       </tr>
      </thead>
      <tbody id="df-exp-rows">
       <tr class="df-exp-row">
        <td>
         <input type="text" class="form-control" name="types[]" value="Rent">
        </td>
        <td>
         <input type="number" class="form-control df-exp-amount" name="amounts[]"
value="8000">
        </td>
        <td class="text-end">
         <button type="button" class="btn btn-sm btn-outline-danger df-exp-remove-
row">✕</button>
        </td>
       </tr>
       </tbody>
     </table>
    </div>
    <div class="card-footer text-end">
     <button type="button" class="btn btn-secondary df-exp-calc">Calculate Total</button>
     <button type="submit" class="btn df-cta-btn">Save Expenses</button>
    </div>
   </form>
  </div>
 </div>
 <div class="col-lg-5">
  <div class="card mb-3">
   <div class="card-body">
    <h5 class="card-title">Summary</h5>
    <div class="d-flex justify-content-between mt-3">
```

`

```
          <div>
            <small class="text-muted">Total Expenses</small>
            <div class="h4" id="df-exp-total">₹ 0</div>
          </div>
          <div>
            <small class="text-muted">Categories</small>
            <div class="h4" id="df-exp-categories">0</div>
          </div>
        </div>
      </div>
    </div>
    <div class="card">
      <div class="card-header d-flex justify-content-between">
        <span>Visual Breakdown</span>
        <div class="btn-group btn-group-sm" role="group">
          <button type="button" class="btn btn-outline-secondary" data-chart-type="pie">Pie</button>
          <button type="button" class="btn btn-outline-secondary" data-chart-type="bar">Bar</button>
        </div>
      </div>
      <div class="card-body">
        <canvas id="expensesChartMain" height="200"></canvas>
      </div>
    </div>
  </div>
</div>
{% if history %}
<div class="card mt-4">
  <div class="card-header">Saved History</div>
  <div class="card-body p-0">
    <ul class="list-group list-group-flush">
      {% for item in history %}
        <li class="list-group-item d-flex justify-content-between align-items-center">
          <div>
            <strong>{{ item.date.strftime("%b %Y") }}</strong>
            <small class="text-muted ms-2">Saved on {{ item.date.strftime("%d %b") }}</small>
          </div>
          <div class="text-end">
            <span class="fw-bold d-block">₹ {{ "%.2f"|format(item.total) }}</span>
            <a href="{{ url_for('dashboard.expenses_export_csv', period=item.period) }}" class="small text-decoration-none">Export CSV</a>
          </div>
        </li>
      {% endfor %}
    </ul>
  </div>
</div>
{% endif %}

{% endblock %}
```

`

Investment Adsvisor Tab:



Investment Advisor

Explore different investment options, understand their risk profiles, and jump to trusted resources for deeper research.

Build a diversified plan

These cards give you a quick, jargon-free overview of common investment options. Use them as a starting point and always verify with your own research or a licensed financial advisor.

Risk legend

Low Capital relatively stable, returns moderate.

Medium Fluctuations possible, aim for higher growth.

High Large ups & downs, suitable for long-term and experienced investors.

Fixed Deposit

Low-risk bank deposits with fixed interest.

WHY CONSIDER FD?
- Guaranteed returns
- Low risk
- Good for emergency funds

THINGS TO NOTE
- Interest taxed
- Penalty for early withdrawal

Low risk | Bank rates

Public Provident Fund (PPF)

Long-term tax-efficient savings.

WHY PPF?
- Tax benefits
- Long-term savings
- Government backed

THINGS TO NOTE
- 15-year lock-in
- Partial withdrawals have rules

Low to Medium | Learn PPF

Gold Investment

A traditional hedge against inflation

WHY CONSIDER GOLD?
- Preserves value during inflation
- Acts as a safe haven asset
- Highly liquid investment

RISKS & POINTS TO NOTE
- No regular income
- Prices can fluctuate short-term
- Storage costs for physical gold

Medium risk | View gold prices

Mutual Funds (SIP)

Start SIPs for disciplined investing.

WHY SIPS?
- Rupee cost averaging
- Diversified exposure
- Professional management

THINGS TO NOTE
- Market risk
- Choose fund based on horizon

Medium risk | Explore funds

Direct Equity (Stocks)

High-growth but higher volatility.

WHY STOCKS?
- High upside potential
- Ownership in companies

THINGS TO NOTE
- High volatility
- Requires research

High risk | Market info

Systematic Investment Plan (SIP)

Invest small amounts regularly

HOW SIP HELPS
- Disciplined investing
- Rupee cost averaging
- Flexible amounts

THINGS TO REMEMBER
- Market fluctuations
- Needs consistency
- Long-term commitment

Medium risk | SIP calculator

Real Estate

Property-based long-term investment

WHY REAL ESTATE?
- Rental income
- Asset appreciation
- Physical ownership

CHALLENGES
- High capital needed
- Low liquidity
- Legal issues

Medium–High risk | Property listings

National Pension Scheme (NPS)

Retirement-focused investment plan

WHY NPS?
- Tax benefits
- Market-linked growth
- Retirement corpus

CONSIDERATIONS
- Partial lock-in
- Market exposure
- Withdrawal rules

Medium risk | NPS portal

Cryptocurrency

Digital assets with high volatility

WHY PEOPLE INVEST
- Decentralized
- High return potential
- Global access

MAJOR RISKS
- Extreme volatility
- Regulatory uncertainty
- Security risks

Very high risk | Learn about crypto

Exchange Traded Funds (ETF)

Index-based low-cost investment

ETF ADVANTAGES
- Low expense ratio
- Diversification
- Easy trading

LIMITATIONS
- Market risk
- Tracking error
- Requires demat account

Medium risk | ETF list

© 2026 Disha Finance. All rights reserved.

Code:

```
{% extends "dashboard/layout.html" %}
```

`

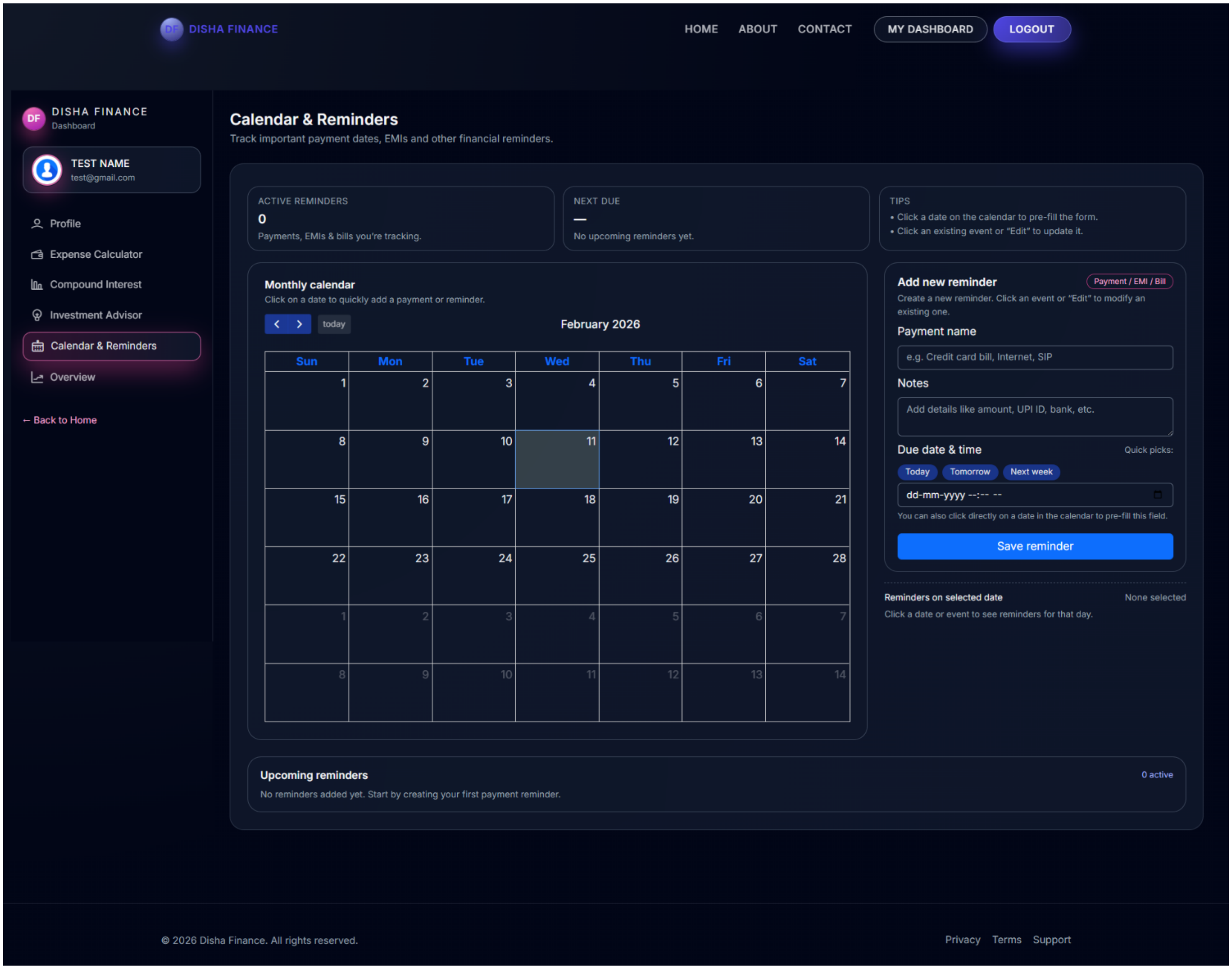
```
{% block dashboard_title %}Investment Advisor{% endblock %}
{% block dashboard_subtitle %}Explore investment options and understand risk profiles.{%
endblock %}

{% block dashboard_content %}
<div class="df-adv-layout">

  <div class="row g-4 mb-3">
    <div class="col-lg-8">
      <div class="card h-100">
        <div class="card-body">
          <h5 class="card-title">Build a Diversified Plan</h5>
          <p class="card-text">
            Use these cards as a starting point. Always verify with your own research.
          </p>
        </div>
      </div>
    </div>

    <div class="col-lg-4">
      <div class="card h-100">
        <div class="card-body">
          <h5 class="card-title">Risk Legend</h5>
          <ul class="list-unstyled small">
            <li><span class="badge bg-success">Low</span> Stable capital, moderate
returns.</li>
            <li><span class="badge bg-warning text-dark">Medium</span> Balanced growth and
risk.</li>
            <li><span class="badge bg-danger">High</span> High volatility, long-term
growth.</li>
          </ul>
        </div>
      </div>
    </div>
  </div>

  <div class="row g-4">
    {% if options %}
      {% for opt in options %}
        <div class="col-md-6 col-xl-4">
          <div class="card h-100">
            <div class="card-header d-flex align-items-center gap-2">
              <i class="{{ opt.icon_class or 'lni lni-investment' }}"></i>
              <h5 class="mb-0">{{ opt.name }}</h5>
            </div>

            <div class="card-body">
              {% if opt.tagline %}
                <p class="text-muted small">{{ opt.tagline }}</p>
              {% endif %}
```

`

```
              {% if opt.section1_points %}
                <h6>{{ opt.section1_label or "Details" }}</h6>
                <ul>
                  {% for line in opt.section1_points.splitlines() if line %}
                    <li>{{ line }}</li>
                  {% endfor %}
                </ul>
              {% endif %}

              {% if opt.section2_points %}
                <h6>{{ opt.section2_label or "Notes" }}</h6>
                <ul>
                  {% for line in opt.section2_points.splitlines() if line %}
                    <li>{{ line }}</li>
                  {% endfor %}
                </ul>
              {% endif %}
            </div>

            <div class="card-footer d-flex justify-content-between align-items-center">
              <span class="badge bg-{{ 'danger' if opt.risk_pill_level == 'high' else 'success' }}">
                {{ opt.risk_pill_text or 'Medium Risk' }}
              </span>

              {% if opt.external_url %}
                <a href="{{ opt.external_url }}" target="_blank" class="btn btn-sm btn-outline-primary">
                  Learn More
                </a>
              {% endif %}
            </div>
          </div>
        </div>
      {% endfor %}

    {% else %}
      <div class="col-12">
        <div class="alert alert-info">
          No investment options configured. Please contact admin.
        </div>
      </div>
    {% endif %}
  </div>

</div>
{% endblock %}
```

`

Calender & Reminders Tab:



DF DISHA FINANCE
Dashboard

TEST NAME
test@gmail.com

Calendar & Reminders
Track important payment dates, EMIs and other financial reminders.

ACTIVE REMINDERS
0
Payments, EMIs & bills you're tracking.

NEXT DUE
—
No upcoming reminders yet.

TIPS
- Click a date on the calendar to pre-fill the form.
- Click an existing event or "Edit" to update it.

Monthly calendar
Click on a date to quickly add a payment or reminder.

February 2026

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
1	2	3	4	5	6	7
8	9	10	11	12	13	14

Add new reminder
Payment / EMI / Bill
Create a new reminder. Click an event or "Edit" to modify an existing one.

Payment name
e.g. Credit card bill, Internet, SIP

Notes
Add details like amount, UPI ID, bank, etc.

Due date & time
Quick picks:

dd-mm-yyyy --:-- --
You can also click directly on a date in the calendar to pre-fill this field.

Reminders on selected date
None selected
Click a date or event to see reminders for that day.

Upcoming reminders
0 active
No reminders added yet. Start by creating your first payment reminder.

© 2026 Disha Finance. All rights reserved.

\Code:

```
{% extends "dashboard/layout.html" %}

{% block dashboard_title %}Calendar & Reminders{% endblock %}
{% block dashboard_subtitle %}Track payments, EMIs and bills.{% endblock %}

{% block extra_css %}
  {{ super() }}
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/fullcalendar@6.1.15/main.min.css">
{% endblock %}

{% block extra_js %}
```

`

```
  {{ super() }}
  <script src="https://cdn.jsdelivr.net/npm/fullcalendar@6.1.15/index.global.min.js"></script>
  <script src="{{ url_for('static', filename='js/reminders.js') }}"></script>
{% endblock %}

{% block dashboard_content %}
<div class="df-calendar-wrap">

  <div class="row mb-3 g-3">
    <div class="col-md-4">
      <div class="card p-3">
        <h5>Active Reminders</h5>
        <h3>{{ reminders|length }}</h3>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card p-3">
        <h5>Next Due</h5>
        <h3>
          {% if reminders %}
            {{ reminders[0].reminder_date.strftime("%d %b") }}
          {% else %}
            —
          {% endif %}
        </h3>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card p-3 bg-light">
        <small><strong>Tip:</strong> Click on a calendar date to quickly add a reminder for that
day.</small>
      </div>
    </div>
  </div>

  <div class="row g-4">

    <div class="col-lg-8">
      <div class="card h-100">
        <div class="card-body">
          <div id="df-calendar" data-events-url="{{ url_for('dashboard.calendar_events')
}}"></div>
        </div>
      </div>
    </div>

    <div class="col-lg-4">
      <div class="card">
        <div class="card-header">Add / Edit Reminder</div>
        <div class="card-body">
```

`

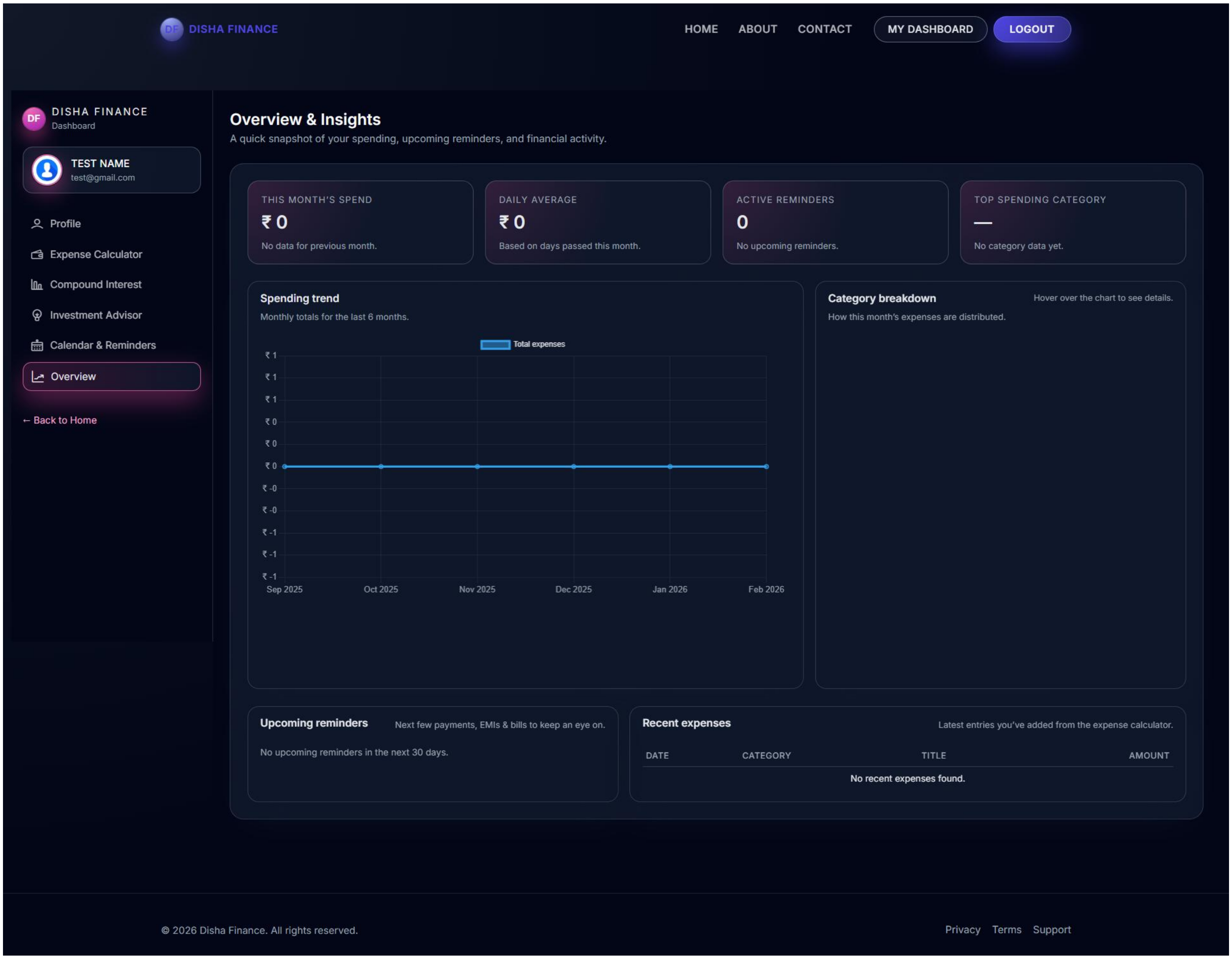
```
          <form id="df-add-rem-form" data-add-url="{{ url_for('dashboard.add_reminder') }}">
            <input type="hidden" id="df-reminder-id" name="reminder_id">

            <div class="mb-3">
              <label class="form-label">Title</label>
              <input type="text" name="title" class="form-control" placeholder="e.g. Credit Card
Bill" required>
            </div>

            <div class="mb-3">
              <label class="form-label">Notes</label>
              <textarea name="description" class="form-control" rows="2"></textarea>
            </div>

            <div class="mb-3">
              <label class="form-label">Date & Time</label>
              <input type="datetime-local" name="date" class="form-control" required>
            </div>

            <div class="d-grid gap-2">
              <button type="submit" class="btn df-cta-btn">Save Reminder</button>
              <button type="button" class="btn btn-secondary d-none" id="df-cal-cancel-
edit">Cancel</button>
            </div>
          </form>

        </div>
      </div>

      <div class="card mt-3">
        <div class="card-header">
          Events on <span id="df-daylist-date">Selected Date</span>
        </div>
        <ul class="list-group list-group-flush" id="df-daylist-items">
          <li class="list-group-item text-muted small">Click a date to see details.</li>
        </ul>
      </div>
    </div>

  </div>

  <div class="card mt-4">
    <div class="card-header">All Upcoming Reminders</div>
    <div class="card-body">
      {% if reminders %}
        <ul class="list-group">
          {% for r in reminders %}
            <li class="list-group-item d-flex justify-content-between align-items-center">
              <div>
                <strong>{{ r.title }}</strong> <br>
                <small class="text-muted">{{ r.reminder_date.strftime("%d %b %Y, %I:%M %p")
```

`

```
}}</small>
        {% if r.description %}<p class="mb-0 small">{{ r.description }}</p>{% endif %}
       </div>
       <div>
        <button type="button" class="btn btn-sm btn-outline-info df-edit-rem"
         data-id="{{ r.id }}" data-title="{{ r.title }}" data-date="{{
r.reminder_date.isoformat() }}">
         Edit
        </button>
        <button type="button" class="btn btn-sm btn-outline-danger df-del-rem" data-id="{{
r.id }}">
         Delete
        </button>
       </div>
      </li>
     {% endfor %}
    </ul>
   {% else %}
    <p class="text-muted">No reminders found.</p>
   {% endif %}
  </div>
 </div>

</div>
{% endblock %}
```

`

<u>OverView Tab:</u>



Overview & Insights

A quick snapshot of your spending, upcoming reminders, and financial activity.

THIS MONTH'S SPEND	DAILY AVERAGE	ACTIVE REMINDERS	TOP SPENDING CATEGORY
₹ 0	₹ 0	0	—
No data for previous month.	Based on days passed this month.	No upcoming reminders.	No category data yet.

Spending trend

Monthly totals for the last 6 months.

Category breakdown

Hover over the chart to see details.

How this month's expenses are distributed.

Upcoming reminders

Next few payments, EMIs & bills to keep an eye on.

No upcoming reminders in the next 30 days.

Recent expenses

Latest entries you've added from the expense calculator.

DATE	CATEGORY	TITLE	AMOUNT
No recent expenses found.			

© 2026 Disha Finance. All rights reserved.

Code:

```
{% extends "dashboard/layout.html" %}
{% block dashboard_title %}Overview & Insights{% endblock %}
{% block dashboard_subtitle %}A quick snapshot of your financial activity.{% endblock %}

{% block dashboard_content %}
<div class="df-ov-wrapper">

  <div class="row g-4 mb-4">
    <div class="col-md-3">
      <div class="card p-3">
        <small class="text-muted">This Month's Spend</small>
```

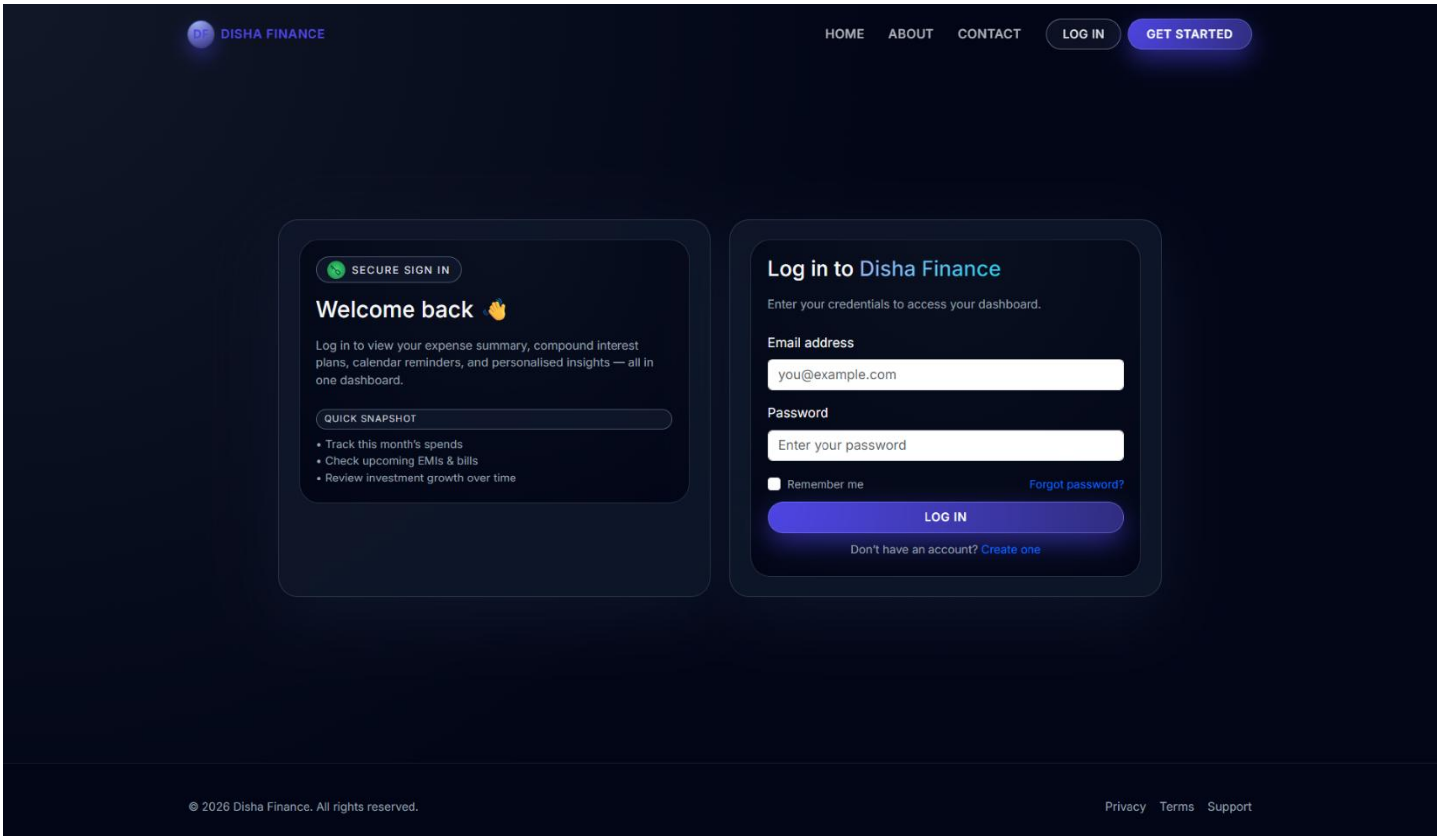
`

```
      <h3 id="df-ov-total-month">₹ 0</h3>
      <small id="df-ov-month-change" class="text-success">—</small>
    </div>
  </div>

  <div class="col-md-3">
    <div class="card p-3">
      <small class="text-muted">Daily Average</small>
      <h3 id="df-ov-daily-avg">₹ 0</h3>
    </div>
  </div>

  <div class="col-md-3">
    <div class="card p-3">
      <small class="text-muted">Active Reminders</small>
      <h3 id="df-ov-reminders-count">0</h3>
      <small id="df-ov-next-reminder">No upcoming reminders.</small>
    </div>
  </div>

  <div class="col-md-3">
    <div class="card p-3">
      <small class="text-muted">Top Category</small>
      <h3 id="df-ov-top-category">—</h3>
      <small id="df-ov-top-category-amount">—</small>
    </div>
  </div>
</div>

<div class="row g-4 mb-4">
  <div class="col-lg-8">
    <div class="card h-100">
      <div class="card-header">Spending Trend (Last 6 Months)</div>
      <div class="card-body">
        <canvas id="df-ov-expense-trend"></canvas>
      </div>
    </div>
  </div>

  <div class="col-lg-4">
    <div class="card h-100">
      <div class="card-header">Category Breakdown</div>
      <div class="card-body">
        <canvas id="df-ov-category-chart"></canvas>
      </div>
    </div>
  </div>
</div>

<div class="row g-4">
  <div class="col-md-6">
```

```
`
      <div class="card h-100">
        <div class="card-header">Upcoming Reminders</div>
        <div class="card-body">
          <ul class="list-group list-group-flush" id="df-ov-rem-list">
            <li class="list-group-item text-muted">Loading reminders...</li>
          </ul>
        </div>
      </div>
    </div>

    <div class="col-md-6">
      <div class="card h-100">
        <div class="card-header">Recent Expenses</div>
        <div class="card-body p-0">
          <table class="table table-striped mb-0">
            <thead>
              <tr>
                <th>Date</th>
                <th>Category</th>
                <th class="text-end">Amount</th>
              </tr>
            </thead>
            <tbody id="df-ov-expense-rows">
              <tr><td colspan="3" class="text-center text-muted">Loading expenses...</td></tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>

</div>
{% endblock %}
```

`

<u>Login Page:</u>



SECURE SIGN IN

Welcome back 👋

Log in to view your expense summary, compound interest plans, calendar reminders, and personalised insights — all in one dashboard.

QUICK SNAPSHOT

- Track this month's spends
- Check upcoming EMIs & bills
- Review investment growth over time

Log in to Disha Finance

Enter your credentials to access your dashboard.

Email address

you@example.com

Password

Enter your password

□ Remember me Forgot password?

LOG IN

Don't have an account? Create one

© 2026 Disha Finance. All rights reserved.

Privacy Terms Support

Code:

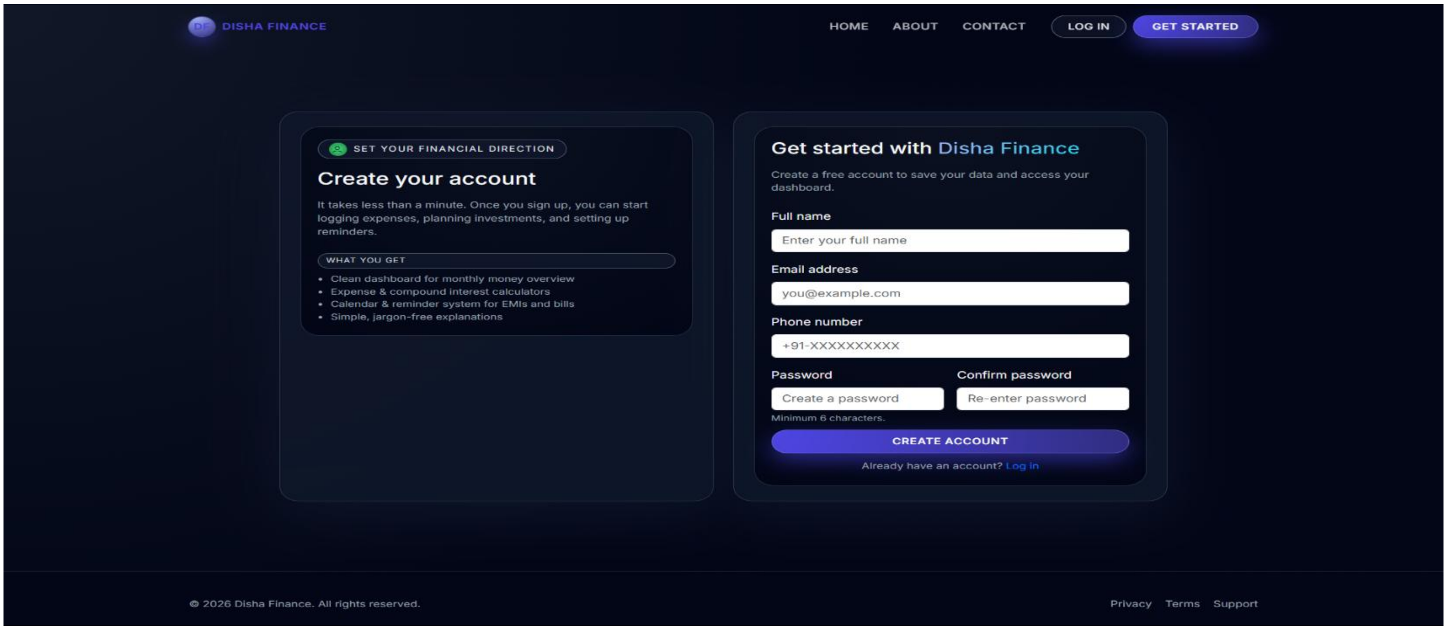
```
{% extends "base.html" %}
{% block title %}Login – Disha Finance{% endblock %}

{% block content %}
<section class="d-flex align-items-center" style="min-height: 80vh;">
  <div class="container">
    <div class="row justify-content-center">

      <div class="col-lg-5 d-none d-lg-block">
        <div class="card h-100 bg-light">
          <div class="card-body">
            <h3>Welcome Back 👋</h3>
            <p>Log in to access your financial dashboard.</p>
            <ul>
              <li>Track monthly spends</li>
              <li>Check upcoming bills</li>
              <li>Review investment growth</li>
            </ul>
          </div>
```

`

```
        </div>
      </div>
      <div class="col-lg-5 col-md-8">
        <div class="card shadow-sm">
          <div class="card-body p-4">
            <h2 class="mb-3">Log In</h2>

            {% with messages = get_flashed_messages(with_categories=true) %}
              {% if messages %}
                {% for category, msg in messages %}
                  <div class="alert alert-{{ category }}">{{ msg }}</div>
                {% endfor %}
              {% endif %}
            {% endwith %}

            <form action="{{ url_for('auth.login') }}" method="POST">

              <div class="mb-3">
                <label for="email" class="form-label">Email Address</label>
                <input type="email" name="email" class="form-control" required>
              </div>
              <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" name="password" class="form-control" required>
              </div>

              <div class="d-flex justify-content-between mb-3">
                <div class="form-check">
                  <input type="checkbox" class="form-check-input" name="remember"
id="remember">
                  <label class="form-check-label" for="remember">Remember me</label>
                </div>
                <a href="#" class="small">Forgot Password?</a>
              </div>

              <button type="submit" class="btn df-cta-btn w-100 mb-3">Log In</button>

              <div class="text-center">
                <small>Don't have an account? <a href="{{ url_for('auth.register') }}">Create
one</a></small>
              </div>

            </form>
          </div>
        </div>
      </div>

    </div>
  </div>
</section>
{% endblock %}
```

`

Register Page:



SET YOUR FINANCIAL DIRECTION

Create your account

It takes less than a minute. Once you sign up, you can start logging expenses, planning investments, and setting up reminders.

WHAT YOU GET

- Clean dashboard for monthly money overview
- Expense & compound interest calculators
- Calendar & reminder system for EMIs and bills
- Simple, jargon-free explanations

Get started with Disha Finance

Create a free account to save your data and access your dashboard.

Full name: Enter your full name

Email address: you@example.com

Phone number: +91-XXXXXXXXXX

Password: Create a password

Confirm password: Re-enter password

Minimum 6 characters.

Already have an account? Log in

© 2026 Disha Finance. All rights reserved.

Code:

```
{% extends "base.html" %}
{% block title %}Register – Disha Finance{% endblock %}
{% block content %}
<section class="d-flex align-items-center" style="min-height: 80vh;">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-lg-5 d-none d-lg-block">
        <div class="card h-100 bg-light">
          <div class="card-body">
            <h3>Create your account</h3>
            <p>Join Disha Finance to start taking control of your money.</p>
            <ul>
              <li>Expense & CI Calculators</li>
              <li>Calendar & Reminder System</li>
              <li>Secure Data Storage</li>
            </ul>
          </div>
        </div>
      </div>
      <div class="col-lg-5 col-md-8">
        <div class="card shadow-sm">
          <div class="card-body p-4">
            <h2 class="mb-3">Get Started</h2>
            {% with messages = get_flashed_messages(with_categories=true) %}
              {% if messages %}
```

```
`
                {% for category, msg in messages %}
                  <div class="alert alert-{{ category }}">{{ msg }}</div>
                {% endfor %}
              {% endif %}
            {% endwith %}
            <form action="{{ url_for('auth.register') }}" method="POST" class="row g-3">

              <div class="col-12">
                <label for="name" class="form-label">Full Name</label>
                <input type="text" name="name" class="form-control" required>
              </div>
              <div class="col-12">
                <label for="email" class="form-label">Email Address</label>
                <input type="email" name="email" class="form-control" required>
              </div>
              <div class="col-12">
                <label for="phone" class="form-label">Phone Number</label>
                <input type="text" name="phone" class="form-control" placeholder="+91..."
required>
              </div>
              <div class="col-md-6">
                <label for="password" class="form-label">Password</label>
                <input type="password" name="password" class="form-control" required>
                <small class="text-muted">Min 6 chars.</small>
              </div>

              <div class="col-md-6">
                <label for="confirm_password" class="form-label">Confirm</label>
                <input type="password" name="confirm_password" class="form-control" required>
              </div>

              <div class="col-12 mt-4">
                <button type="submit" class="btn df-cta-btn w-100">Create Account</button>
              </div>
              <div class="col-12 text-center">
                <small>Already have an account? <a href="{{ url_for('auth.login') }}">Log
in</a></small>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>

{% endblock %}
```

`

LIMITATIONS

1. **Lack of Bank API Integration (Manual Entry):** The current system relies entirely on manual data entry for logging expenses and income. Users must physically type in every transaction, which can be tedious and prone to human error. Unlike commercial apps (like Mint or YNAB), this system does not currently connect to real-world bank accounts for automatic transaction fetching, which may reduce long-term user retention.
2. **No Multi-Currency Support:** The platform currently operates with a single currency unit (implied based on user input). It does not support real-time currency conversion or multi-currency wallets. This limits the usability for users who travel frequently, manage foreign investments, or deal with international transactions, as they cannot view a consolidated net worth in a single base currency.
3. **Basic "Static" Investment Advice:** The "Investment Advisor" module provides recommendations based on static, pre-defined content managed by the admin. It lacks a real-time connection to stock market APIs or live crypto prices. Consequently, the investment advice is educational rather than actionable in real-time, and users cannot track the live performance of their actual investment portfolios within the dashboard.

`

FUTURE SCOPE OF THE PROJECT

1. **AI-Driven Financial Insights:** The system can integrate artificial intelligence (AI) and machine learning algorithms to analyze spending patterns over time. AI-driven insights could automatically detect "anomalous spending" (e.g., a subscription price hike) or suggest personalized budget cuts based on the user's historical data and savings goals.
2. **OCR Receipt Scanning:** Implementing Optical Character Recognition (OCR) technology would allow users to upload images of physical receipts. The system could then automatically parse the date, merchant, and total amount, significantly reducing the manual effort required to log expenses and improving data accuracy.
3. **Integration with Banking APIs (Open Banking):** Collaborating with Open Banking APIs (like Plaid or Yodlee) would enable the secure, read-only fetching of transactions directly from user bank accounts. This would automate the expense tracking process, ensuring that every coffee or utility bill is captured without user intervention.
4. **Gamification and Social Challenges:** The platform can extend its services to include "Savings Challenges" (e.g., "No Spend November" or "52-Week Save"). Gamifying the savings process with badges and leaderboards (among friends) could increase user engagement and motivation to stick to financial goals.
5. **Mobile Application Development:** While the current web dashboard is responsive, developing a dedicated mobile application (iOS/Android) using React Native or Flutter would provide a better native experience. Features like push notifications for bill reminders and offline data entry would significantly enhance usability.

`

CONCLUSION

The **Personal Finance Management System** project successfully addresses the key objectives of providing a comprehensive and user-friendly platform for individual wealth management. By offering specific tools for expense tracking, visualized reporting, and future financial planning (via the Compound Interest Calculator), the platform empowers users to take control of their economic health.

The scope of the project includes creating a secure system that isolates user data, with a focus on **Flask-based authentication**, **SQLAlchemy ORM** for data integrity, and a **dynamic JavaScript frontend**. The **Application Factory pattern** ensures that the system is modular and maintainable, while the **MySQL database** allows for the efficient storage of thousands of transaction records. The project also adheres to security best practices, such as password hashing and CSRF protection, to safeguard sensitive financial information.

In conclusion, this project provides an efficient, scalable, and secure solution for replacing manual spreadsheets with a smart digital dashboard. It meets the platform’s goals by ensuring ease of use, instant visual feedback, and streamlined workflows for logging and analyzing money. With future enhancements like API integration and mobile support, this system is poised to become a valuable tool for personal financial literacy.

`

BIBLIOGRAPHY

The Bibliography contains references to all the documents and resources that were referred to for the creation and successful completion of the project. It contains the names of the referred software engineering documents, Python/Flask documentation, and frontend library standards.

1. **Book:** *Flask Web Development: Developing Web Applications with Python* by Miguel Grinberg.
2. **Book:** *Python Crash Course* by Eric Matthes (Section on Web Apps).
3. **Documentation:** Flask Official Documentation (https://flask.palletsprojects.com/).
4. **Documentation:** Jinja2 Template Designer Documentation.
5. **Resource:** https://getbootstrap.com (Bootstrap 5 Documentation).
6. **Resource:** https://www.chartjs.org (Chart.js Documentation for Data Visualization).
7. **Resource:** https://fullcalendar.io (FullCalendar JavaScript Library).
8. **Resource:** https://sqlalchemy.org (SQLAlchemy ORM Tutorial).
9. **Tool:** https://fontawesome.com (Icons for Dashboard).
10. **Tool:** https://mermaid.live (For System Diagrams).
11. **Tutorials:** "Corey Schafer - Python Flask Tutorials" on YouTube.