BudgetMate Portal

Project Report

By
Sheetal Nilawar – AF04956455
Dimpal Dewasi – AF04956456

Index

Acknowledgement

The project **"BudgetMate Portal"** is the Project work carried out by

Name	Enrollment No
Sheetal Nilawar	**AF04956455**
Dimpal Dewasi	**AF04956456**

We are thankful to my project guide for guiding me to complete the Project.

Their suggestions and valuable information regarding the formation of the Project Report have provided me a lot of help in completing the Project and its related topics.

We are also thankful to my family member and friends who were always there to provide support and moral boost up.

Abstract

Managing personal finances effectively is a critical aspect of modern life, yet many individuals struggle with tracking expenses, setting savings goals, and maintaining financial discipline. ***BudgetMate*** is a full-stack personal finance management system designed to address these challenges by providing users with an intuitive and interactive platform. The application enables users to record income and expenses, categorize transactions, set financial goals, and monitor their progress through insightful visualizations such as pie charts and reports. A dedicated savings feature allows users to allocate funds toward specific goals, ensuring structured financial growth. The system emphasizes simplicity, security, and usability by integrating authentication, data visualization, and dynamic reporting. By combining expense tracking, goal management, and financial insights in one platform, BudgetMate empowers users to make informed financial decisions, develop better saving habits, and achieve long-term financial stability.

1. Introduction

Managing personal finances is a crucial aspect of modern life. In an era where expenses are diverse and income is often limited; individuals struggle to balance their financial activities while ensuring adequate savings for the future. Poor financial planning can result in debt accumulation, low savings, and the inability to achieve long-term goals such as education, travel, or retirement planning. To overcome these challenges, technology can play a vital role in helping individuals track, plan, and optimize their financial behavior.

BudgetMate is a comprehensive personal finance management system developed to simplify the process of financial planning. It enables users to record, track, and visualize their expenses, set financial goals, and monitor progress in real-time. The application serves as a digital companion that not only records income and expenses but also encourages disciplined savings habits and goal-oriented planning.

1. Background of the Study

In today's digital era, financial literacy and management are essential skills. While many individuals earn a stable monthly income, most lack the discipline or tools to manage it effectively. Traditional methods such as handwritten notes or spreadsheets are time-consuming, error-prone, and lack visualization capabilities.

The demand for personalized, digital financial tools has grown significantly in recent years. Mobile and web-based applications provide accessibility, automation, and real-time tracking, making financial planning more convenient. BudgetMate has been developed with this vision to empower individuals by providing an all-in-one solution for financial management.

2. Motivation

The development of BudgetMate is motivated by several key challenges observed in everyday life:

Lack of Expense Awareness – Most individuals are unaware of where their income is spent, resulting in unnecessary expenses.

Poor Savings Discipline – Without clear goals, savings are often neglected, leading to financial instability during emergencies.

Goal Neglect – People aspire to save for travel, education, or purchases but lack a structured system to track progress.

Need for Visualization – Understanding finances is easier with visual representation (charts, graphs, progress bars) than with raw numbers.

Ease of Use – A user-friendly platform motivates consistent usage, unlike complex financial tools or manual spreadsheets.

BudgetMate addresses these issues by combining expense tracking, savings allocation, and goal management into one platform.

3. Significance of the Study

This project is significant for several reasons:

For Individuals: Helps them take control of their finances, build savings discipline, and achieve short- and long-term goals.

For Students: Assists in tracking educational expenses, pocket money usage, and savings for future needs.

For Professionals: Provides a structured overview of salary, monthly expenses, and savings growth.

For Society: Encourages financial literacy and stability, reducing reliance on debt and promoting savings culture.

4. Features of BudgetMate

The application offers several modules that make financial management seamless:

Transaction Management

a) Record daily income and expenses.
b) Categorize transactions into predefined categories (Food, Travel, Rent, Bills, Savings, Donations, etc.).
c) View transaction history in tabular format with sorting and filtering.
d)

Goal Management

a) Create financial goals with name, target amount, description, and deadline.
b) Allocate money toward goals progressively.
c) View progress using dynamic progress bars.
d) Status updates: Active, Achieved, Expired.

Savings Tracking

a) Track total money saved from monthly income.
b) Automatic reflection of goal-based savings in dashboards and charts.

Visualization Tools

a) Interactive pie charts for expense vs. savings distribution.
b) Excel report generation for record-keeping.
c) Clean, theme-based UI for readability.

Authentication & Security

a) User login and signup system for secure access.
b) Token-based authentication for protected routes.

5. Expected Outcomes

The implementation of BudgetMate aims to achieve the following outcomes:

- Increased financial awareness among users.
- Better management of salary and expenses.
- Stronger savings habits and disciplined goal achievement.
- Clear visualization of financial activities.
- Exportable financial reports for personal record or professional use.

6. Organization of the Report

This project report is structured into the following chapters:

1. Abstract – A brief summary of the project.
2. Introduction – Background, motivation, and significance of the system.
3. Problem Statement & Objectives – Defining the financial challenges and project aims.
4. System Design – Architecture, ER Diagrams, and database schema.
5. Implementation – Description of frontend, backend, and APIs.
6. Results & Discussion – Screenshots, charts, and system performance.
7. Conclusion & Future Scope – Final remarks and scope for enhancement.

2. System Analysis

System analysis is a crucial stage in the software development life cycle (SDLC). It focuses on understanding the existing problems, analysing requirements, and determining the feasibility of the proposed system. For *BudgetMate*, system analysis plays an essential role in identifying user needs for personal finance management, studying existing alternatives, and justifying the development of a new, improved solution.

1.Problem Definition

Managing finances is a significant challenge faced by individuals in today's society. Traditional methods such as manual accounting, maintaining notebooks, or using spreadsheets have the following limitations:

Lack of Real-time Tracking: Users cannot efficiently monitor income and expenses instantly.
Data Complexity: Manual records are often cluttered and hard to interpret.

Absence of Visualization: Understanding spending patterns requires graphs and charts, which are not easily available in traditional methods.

Goal Tracking Issues: Saving for a target (e.g., travel, gadgets, education) is difficult without structured monitoring.

Error Prone: Manual calculations increase the risk of financial mismanagement.

Thus, there is a need for a digital, user-friendly, and automated solution that simplifies financial management and provides visualization of savings and expenses.

2. Objectives of the System

The proposed *BudgetMate* system aims to:

- Provide a centralized platform to record, manage, and analyse financial transactions.
- Enable goal-based savings management for better financial discipline.
- Offer data visualization tools (pie charts, progress bars) for better understanding of spending and saving patterns.
- Enhance user accessibility through a web-based platform.
- Ensure data accuracy and security through authentication and database integration.
- Provide report generation features for exporting financial records.

3. Feasibility Study

Feasibility analysis evaluates the practicality of developing the proposed system. The following aspects are considered:

a. Technical Feasibility

The system is built using React (frontend) and Node.js + Express (backend).
Database: MySQL ensures structured storage of transactions, goals, and salary data.
Tools such as Recharts (for charts) and ExcelJS (for reports) are readily available.
Conclusion: The required technologies are available, reliable, and widely used, making the project technically feasible.

b. Economic Feasibility

The system is cost-effective since it relies on open-source technologies.
No licensing fees are required for core development tools.
Long-term benefits (better financial management, savings awareness) outweigh the minimal development cost.
Conclusion: Economically feasible.

c. Operational Feasibility

Users can easily adapt to the system due to its simple interface.
The system replaces manual effort, increasing efficiency.
Security measures (authentication, authorization) ensure trust.
Conclusion: Operationally feasible.

d. Time Feasibility

The project can be developed within the given timeline using agile methodology.
Features are modular (transactions, goals, reports) and can be built iteratively.
Conclusion: Time feasibility is achieved.

4. System Requirements

a. Functional Requirements:

- User Authentication (Signup, Login).
- Record income and expenses.
- Categorize transactions (Food, Rent, Travel, Savings, etc.).
- Create, update, and delete financial goals.
- Save money into specific goals.
- Visualize financial data (pie charts, progress bars).
- Export reports in Excel format.

b. Non-Functional Requirements:

- Usability: User-friendly interface with intuitive navigation.
- Reliability: System should handle transactions without data loss.
- Scalability: Should support multiple users simultaneously.
- Security: Data protection via token-based authentication.
- Performance: Transactions and reports should load quickly.

c. Hardware Requirements:

- Processor: Intel i5 or higher.
- RAM: 4 GB minimum (8 GB recommended).
- Storage: 500 MB for database and reports.

d. Software Requirements

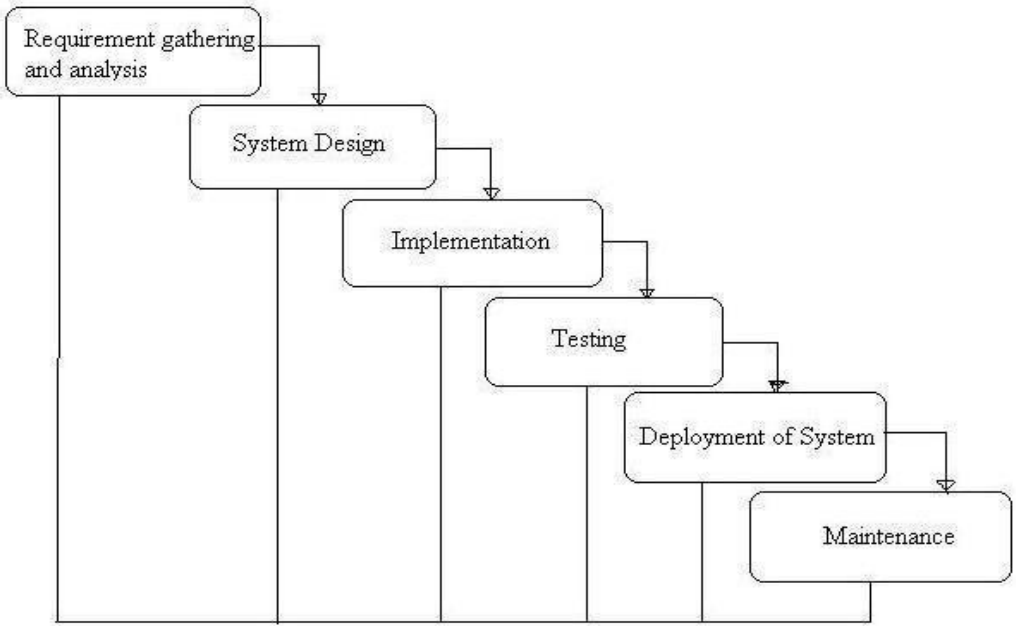
- Operating System: Windows/Linux/MacOS.
- Development Tools: Node.js, MySQL, React.
- Browser: Google Chrome / Mozilla Firefox.

5. Proposed System

The proposed *BudgetMate* system provides:

Expense Tracking: Users can log daily expenses with categories.
Goal Management: Set targets, allocate funds, and track progress.
Savings Integration: Savings automatically update in dashboards and reports.
Data Visualization: Interactive charts for better financial understanding.
Reports: Exportable Excel files for offline analysis.

Unlike traditional methods, this system provides automation, accuracy, visualization, and accessibility, making financial management easier and more effectively.



Phases of Development

1. **Requirement Analysis & System Study**
    - Identifying project goals, challenges, and functional specifications.
    - Gathering stakeholder requirements and defining core functionalities.
2. **System Design**
    - Structuring the **database, modules, and architecture**.
    - Designing **user interfaces** for optimal accessibility.
3. **Implementation (Coding)**
    - Backend development using **Python (Django)**.
    - Frontend design using **HTML, CSS, Bootstrap**.
    - Database integration with **MySQL**.
    - Sentiment analysis integration with **TextBlob**.
4. **Testing & Debugging**
    - Unit testing, integration testing, and usability checks.
    - Debugging for performance improvements.
5. **Deployment & Maintenance**
    - Hosting on a scalable environment.
    - Continuous updates for feature enhancements.

Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

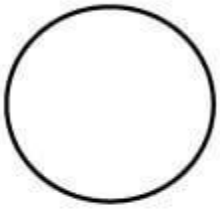
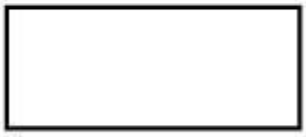
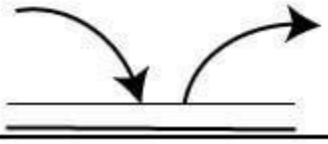
It shows how data enter and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:

Symbol	Name	Function
	Data flow	Used to Connect Processes to each , other , to sources or Sinks; te arrow head indicates direction of data flow.
	Process	Perfroms Some transformation of Input data to yield output data.
	Source of Sink (External Entity)	A Source of System inputs or Sink of System outputs.
	Data Store	A repository of data; the arrow heads indicate net inputs and net outputs to store.

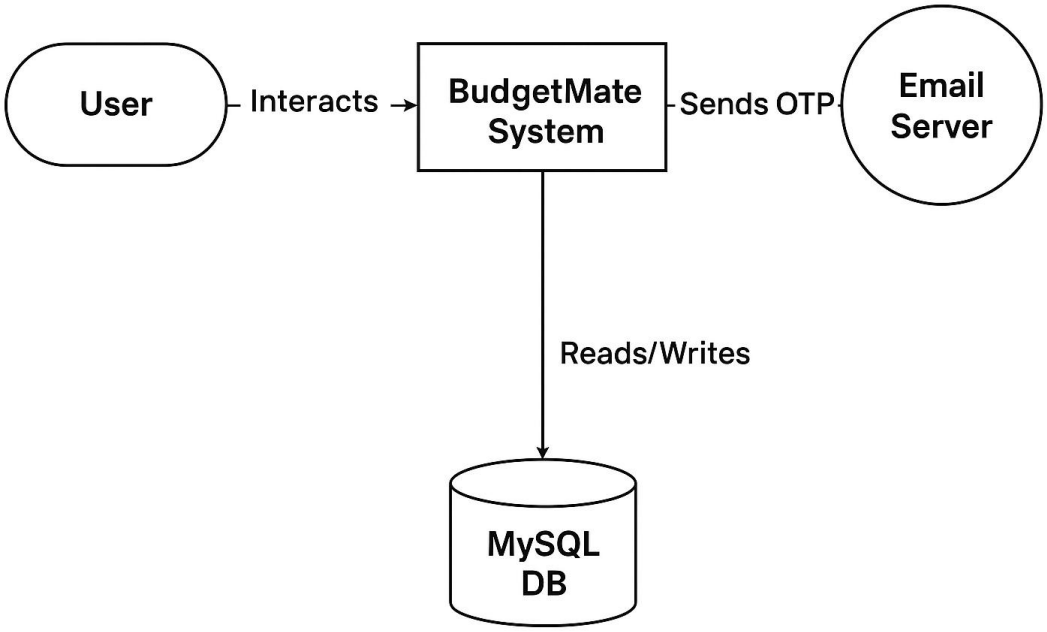
Symbols for Data Flow Diagrams

Circle: A circle (bubble) shows a process that transforms data inputs into data outputs.

Data Flow: A curved line shows the flow of data into or out of a process or data store.

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

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



Zero-Level DFD (Context Diagram) of BudgetMate

The **Zero-Level DFD**, also called the **Context Diagram**, represents the entire **BudgetMate Application** as a single process. It shows the interaction between the system and external entities, along with the flow of data between them.

Entities and Data Flows

1. **User (External Entity)**
    - The primary actor who interacts with the system.
    - **Inputs Provided by User:**
        - Registration details (first name, last name, email, password).
        - Login credentials (email, password).
        - Financial data (transactions, savings, goals, salary).
        - Requests (view reports, check balance, update goals).
    - **Outputs Received by User:**
        - OTPs for authentication.
        - Notifications (success, errors, reminders).
        - Dashboard visualizations (pie charts, summaries).
        - Downloadable reports (Excel/CSV).

2. **Email Server (External Entity)**
    - Works as a medium for **sending OTPs** and notifications to users during login or account creation.
    - Ensures security by verifying identity before access is granted.
3. **Database (External Entity)**
    - Central storage where all system data is kept.
    - **Data Stored:**
        - User credentials and profile information.
        - Transaction records (income, expenses, savings).
        - Goals and progress updates.
        - Reports and summaries.
    - **Data Retrieved:**
        - User's transaction history.
        - Saved progress toward financial goals.
        - Summary calculations (total expenses, salary, current balance).

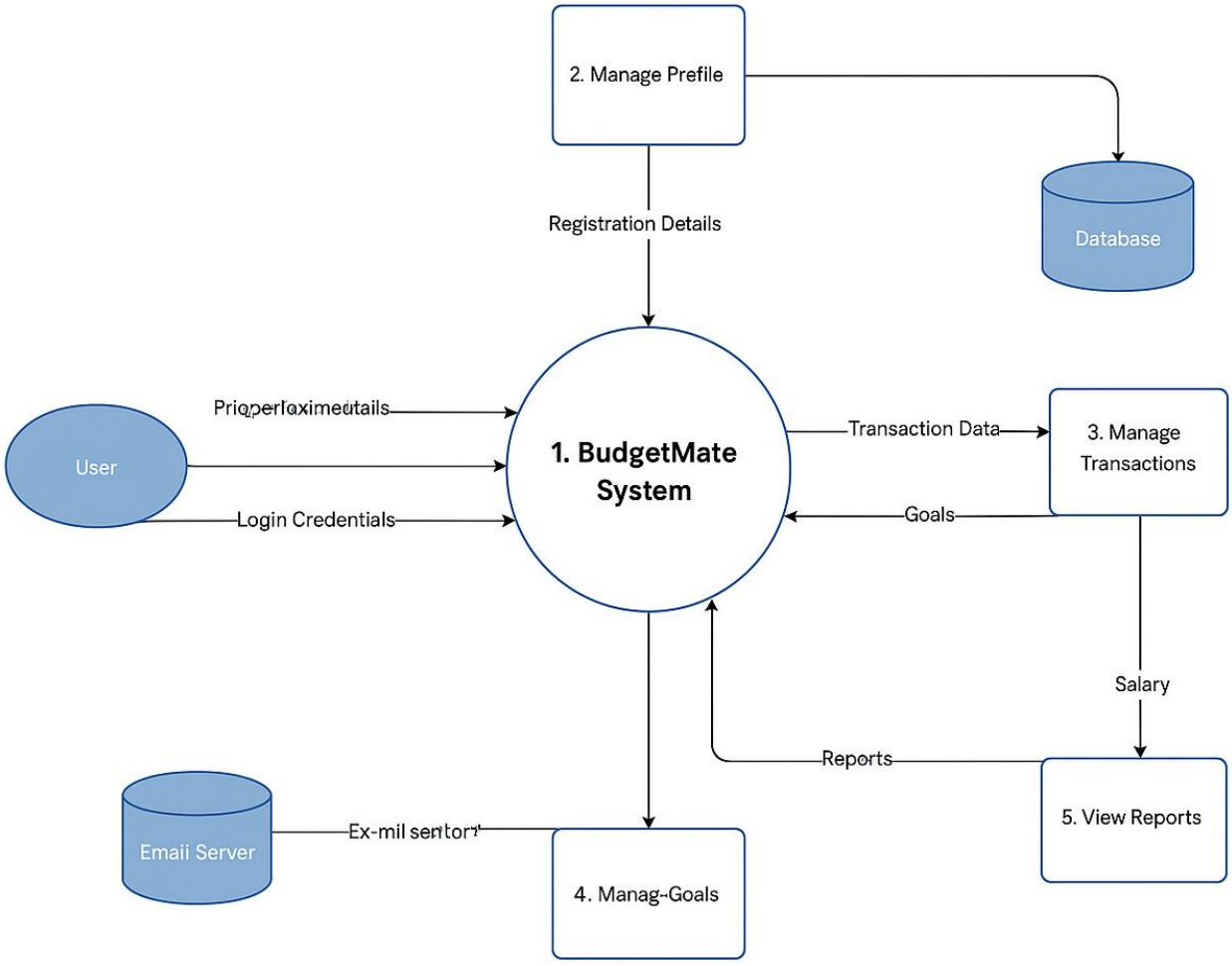
Central Process: BudgetMate System

- The **BudgetMate System** acts as the main process in the zero-level DFD.
- It takes **inputs from the User**, validates them, stores data in the **Database**, and provides processed outputs back to the user.
- It also interacts with the **Email Server** to handle OTP-based authentication.

Overall Flow

1. The **User** interacts with the **BudgetMate System** by entering details (e.g., login, transactions, goals).
2. The system validates credentials and interacts with the **Email Server** for OTP verification.
3. The system stores/retrieves data from the **Database** for transactions, salary, and goals.
4. The system processes data and returns results such as **dashboard insights, savings visualization, and reports** back to the **User**.

First-Level Data Flow Diagram



The **First-Level Data Flow Diagram (DFD)** breaks down the overall system into its major functional components to show how different processes interact with users, the database, and external services. Here's the detailed explanation:

1. User Management

- **Inputs:** User provides registration details, login credentials, or requests OTP verification.
-
- **Processes:**
  - Registration (collect first name, last name, email, password).
  - Login (authenticate using stored credentials).
  - OTP verification (via email for security).

- **Data Flows:**
  - Registration/Login data is stored in the **Database**.
  - OTP is sent through the **Email Server** and verified.
- **Outputs:** Successful login/registration confirmation to the User.

2. Transaction Management

- **Inputs:** User enters expense, income, or savings transaction.

- **Processes:**
  - Adding transactions (amount, category, notes).
  - Categorizing them into income, expense, or savings.
  - Viewing and managing past transactions.

- **Data Flows:**
  - Transactions are stored in the **Database**.
  - Retrieved transactions are displayed back to the User.

- **Outputs:** Updated transaction history and category summaries.

3. Goal Management

- **Inputs:** User sets financial goals (e.g., savings target, end date).

- **Processes:**
  - Adding new goals.
  - Tracking savings progress (via linked transactions).
  - Adding "Save Money" amounts to existing goals.

- **Data Flows:**
  - Goals and their progress are stored in the **Database**.
  - User retrieves updated status (active, achieved, expired)
    .
- **Outputs:** Progress reports, updated goal status.

4. Report Generation

- **Inputs:** User requests summary or export report.

- **Processes:**
  - Generating expense vs savings summaries.
  - Exporting reports in Excel format.

- **Data Flows:**
  - Data pulled from **Database**.
  - Reports generated and sent to the User.

- **Outputs:** Downloadable Excel reports, dashboards with pie charts.

5. Salary Management

- **Inputs:** User provides or updates monthly salary.

- **Processes:**
  - Storing or modifying salary records.
  - Calculating current balance (salary – expenses + savings).

- **Data Flows:**
  - Salary details stored in the **Database**.
  - Reflected in dashboards and reports.

- **Outputs:** Updated salary and balance information.

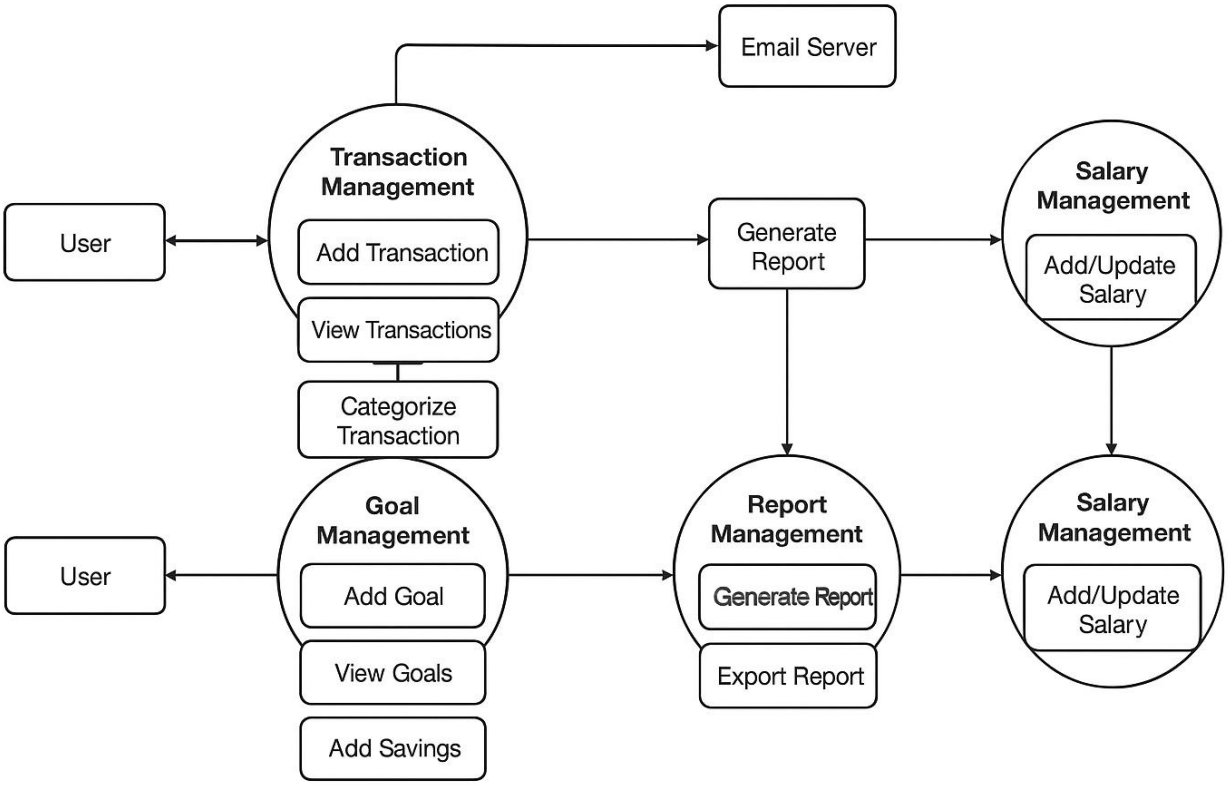
External Entities

- **User:** Main actor interacting with all processes.

- **Email Server:** Used for OTP verification during signup/login.

- **Database:** Central storage for user info, transactions, goals, and salary data.

Second-Level Data Flow Diagram



Second-Level Data Flow Diagram (DFD)

The second-level DFD provides a more detailed breakdown of the **first-level processes**, showing the sub-processes, specific data stores, and interactions between system components.

1. User Management

- **Inputs:** User credentials (signup, login), profile updates.
- **Processes:**
  - Registration validation.
  - Authentication using stored credentials.
  - Profile management (first name, last name, etc.).
- **Outputs:** Authenticated user session, updated user details.
- **Data Stores:** Users Table.

2. Transaction Management

- **Inputs:** Transaction details (amount, category, date, notes, goal association).
- **Processes:**
  - Adding new transactions (expenses, income, savings).
  - Categorizing transactions (food, rent, transport, savings, etc.).

    - Linking savings to specific goals.
- **Outputs:** Updated transaction records, categorized expense/saving data.
- **Data Stores:** Transactions Table.

3. Goals Management

- **Inputs:** Goal details (name, target amount, end date, description).
- **Processes:**
    - Create/Edit/Delete financial goals.
    - Track savings linked to a goal.
    - Update goal status (Active, Achieved, Expired).
- **Outputs:** Goal progress, remaining savings amount.
- **Data Stores:** Goals Table.

4. Report Generation

- **Inputs:** Transactions and savings data from database.
- **Processes:**
    - Aggregating expenses by category.
    - Summarizing monthly salary, expenses, and savings.
    - Generating reports (Excel export).
- **Outputs:** Pie charts, downloadable Excel reports.

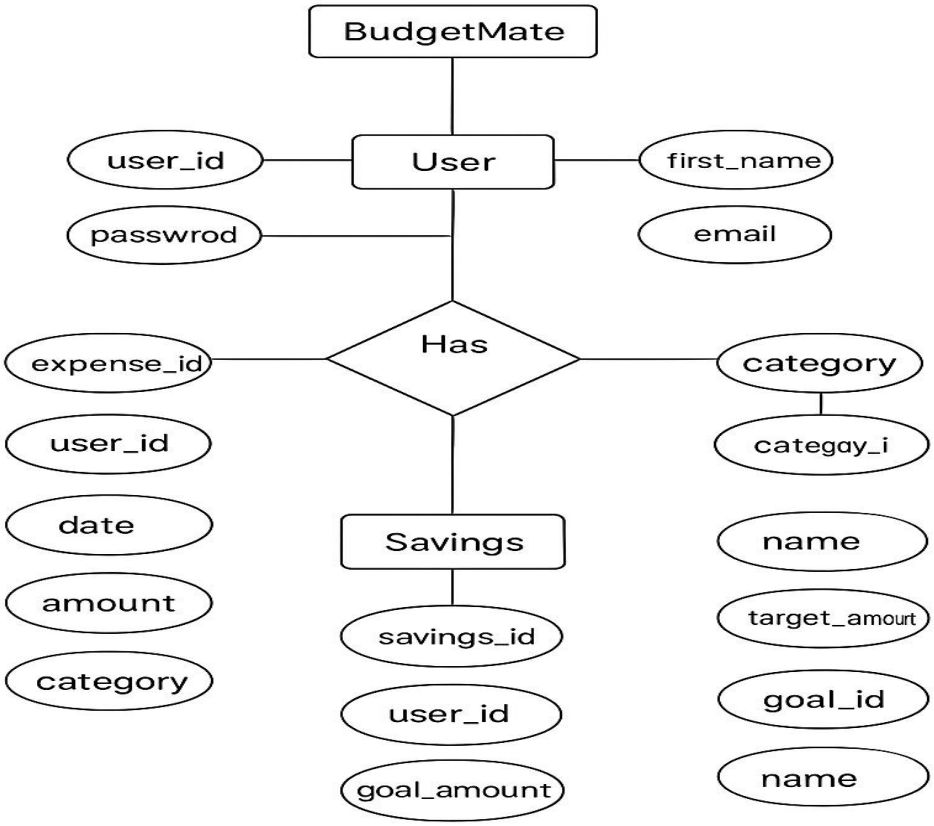
5. Dashboard & Visualization

- **Inputs:** Summarized salary, expenses, and savings data.
- **Processes:**
    - Displaying balance, total expenses, and salary.
    - Rendering pie chart (Expenses vs. Savings).
    - Showing real-time progress of goals.
- **Outputs:** Interactive UI dashboard for users.

Data Stores in Second-Level DFD

- **Users** (user_id, first_name, last_name, email, password, salary).
- **Transactions** (transaction_id, user_id, category, amount, date, notes, goal_id).
- **Goals** (goal_id, user_id, goal_name, goal_amount, saved_amount, status, end_date).

ER diagram:



The ER diagram for **BudgetMate** illustrates the logical structure of the database, showing entities, attributes, and their relationships. Here's a detailed description:

1. Entities and Their Attributes

User

- Attributes: user_id (PK), name, email, password, created_at
- Represents the registered users of the system.

Goal

- Attributes: goal_id (PK), goal_name, goal_amount, description, end_date, achieved, user_id (FK)
- Defines savings goals set by users.

Transaction

- Attributes: transaction_id (PK), amount, category, type (income/expense/savings), date, user_id (FK), goal_id (FK)

- Captures financial activities of the user, including income, expenses, and savings.

Category

- Attributes: category_id (PK), category_name, description
- Represents classifications for expenses and income (e.g., food, rent, salary).

Salary

- Attributes: salary_id (PK), monthly_salary, date_set, user_id (FK)
- Stores users' salary details for monthly budgeting.

2. Relationships

- **User → Goal**: One user can create many goals (1 : M).
- **User → Transaction**: One user can have multiple transactions (1 : M).
- **Goal → Transaction**: A goal can be linked to many transactions (savings), but each transaction belongs to one goal (1 : M).
- **Category → Transaction**: Each transaction is assigned to one category, but a category can have many transactions (1 : M).
- **User → Salary**: A user can have multiple salary records over time (1 : M).

3. ER Diagram Summary

- Central entity is **User**, connected with **Goals, Transactions, and Salary**.
- **Transactions** act as a bridge entity linking **User, Goals, and Categories**.
- This structure ensures normalization, avoids redundancy, and allows detailed financial tracking.

3. System design

A) Modules

1. User Module

Users can:

- Register/Login securely.
- Add income (monthly salary).
- Add, view, filter, and export transactions.
- Create savings goals with target amount, description, and end date.
- Add money towards specific goals (savings).
- Track progress of goals through progress bars.
- View dashboard summaries (salary, total expenses, balance, pie chart of expenses vs savings).
- Download transaction reports in Excel format.

2. Admin Module

Admins have full control over the platform and can:

- Secure Login System – Access the admin dashboard with authentication.
- User Management – View and manage registered users.
- Transaction Oversight – Monitor transaction logs for fraud or misuse.
- Goal Oversight – Ensure goals are properly linked with transactions.
- Report Generation – View and download reports of all users' financial activity.
- System Settings – Manage platform-wide configurations (like categories).

3. System Module (Backend Services)

This module handles the core operations of BudgetMate:

- Authentication (JWT-based).
- CRUD operations for transactions, goals, and salary.
- Linking savings transactions to goals.
- Generating summary analytics (salary vs expenses vs savings).
- Generating Excel reports for download.

B) Data Structure of All Modules

We have organized one database BudgetMate for system design. It is based on MySQL and consists of the following customized tables:

1. Customized Tables
   - Users Table (users)
   - Stores user details: id, first_name, last_name, email, password.
   - Transactions Table (transactions)
   - Stores each transaction: id, user_id, amount, category, date, notes, goal_id.

2. Goals Table (goals)
   - Stores goal details: id, user_id, goal_name, goal_amount, description, end_date, achieved.

3. Salary Table (salary)
   - Stores monthly salary for each user: id, user_id, monthly_salary.

Relationships Between Tables (ER/Class Diagram)

One user → Many transactions.

One user → Many goals.

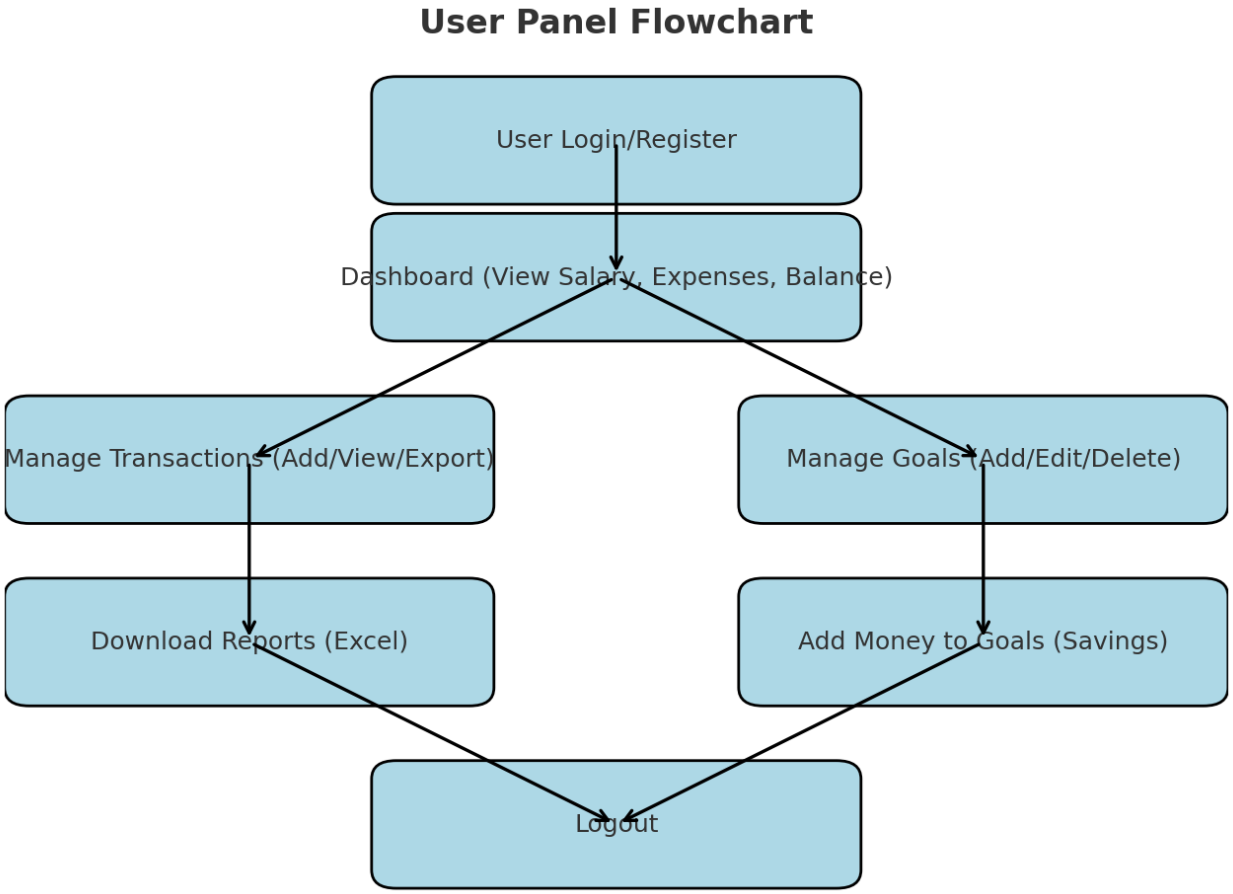
One goal → Many savings transactions (via goal_id).

One user → One salary record.

C) Procedural Design

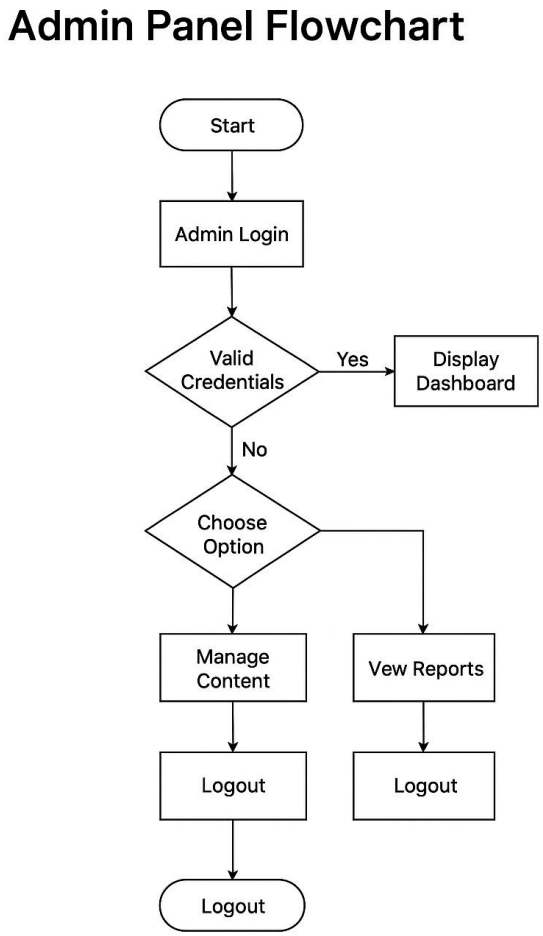
1. User Panel Design

Users log in/register.
Can manage income, expenses, goals, and savings.
Dashboard displays salary, expenses, balance, and pie chart.
Users can add savings towards goals, and reports update dynamically.
Flowchart – User Panel



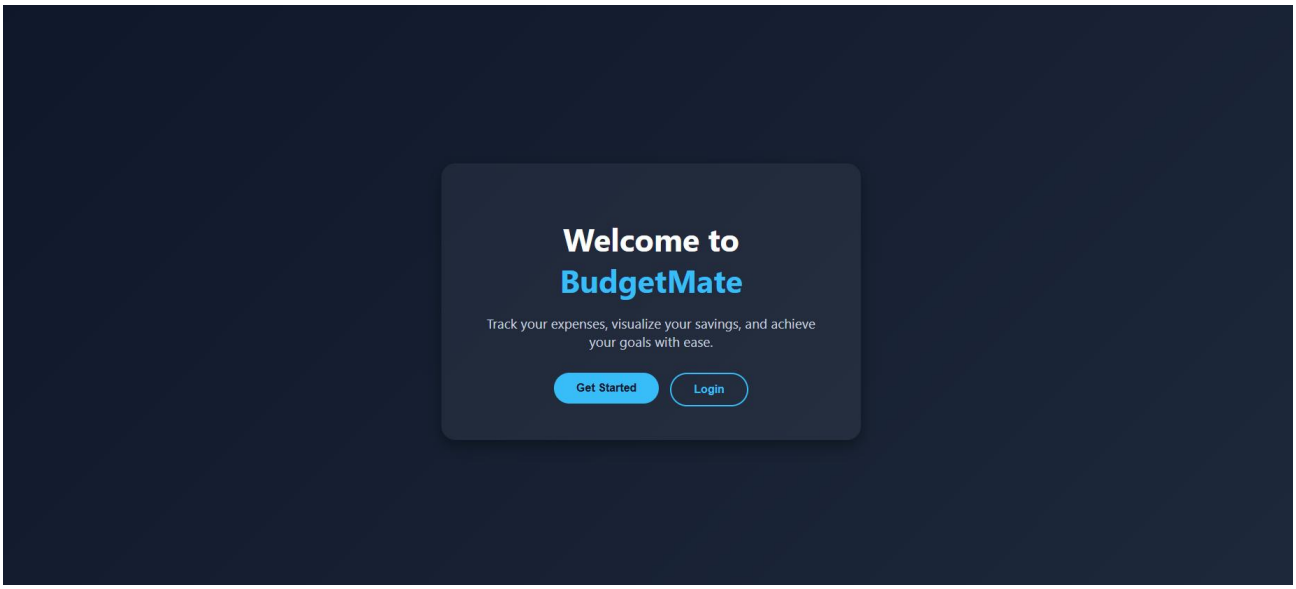
2. Admin Panel Design

Admin logs in with secure credentials.
Access to dashboard with user and transaction overview.
Can view reports, manage users, and oversee platform activities.
Flowchart – Admin Panel

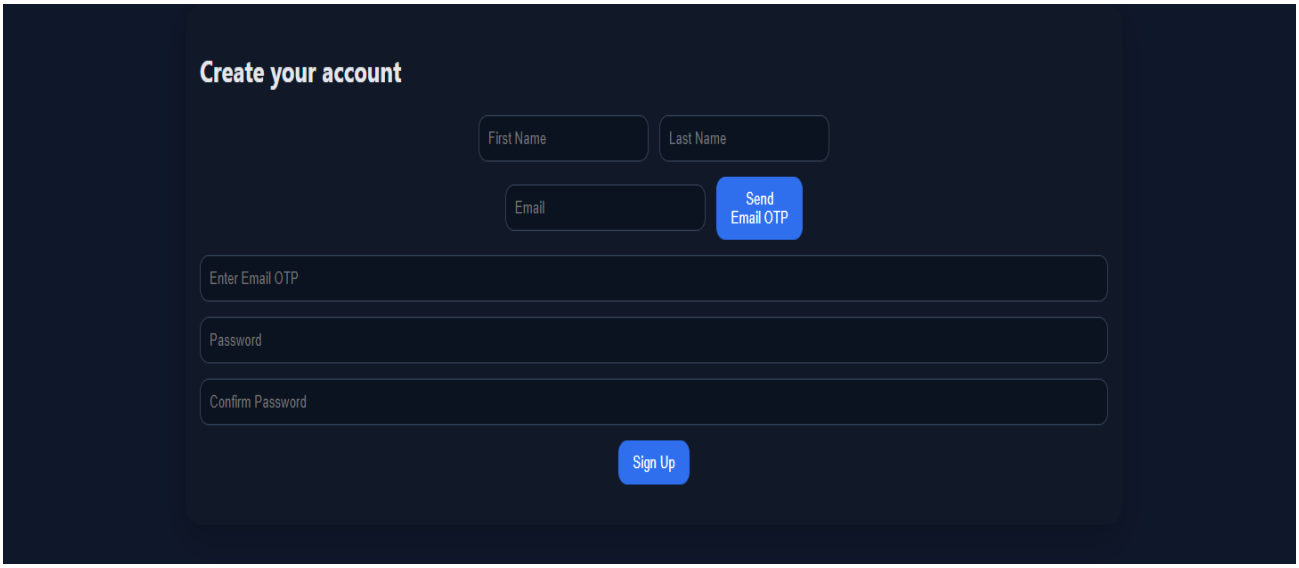


D. Screenshots (Sample Pages)

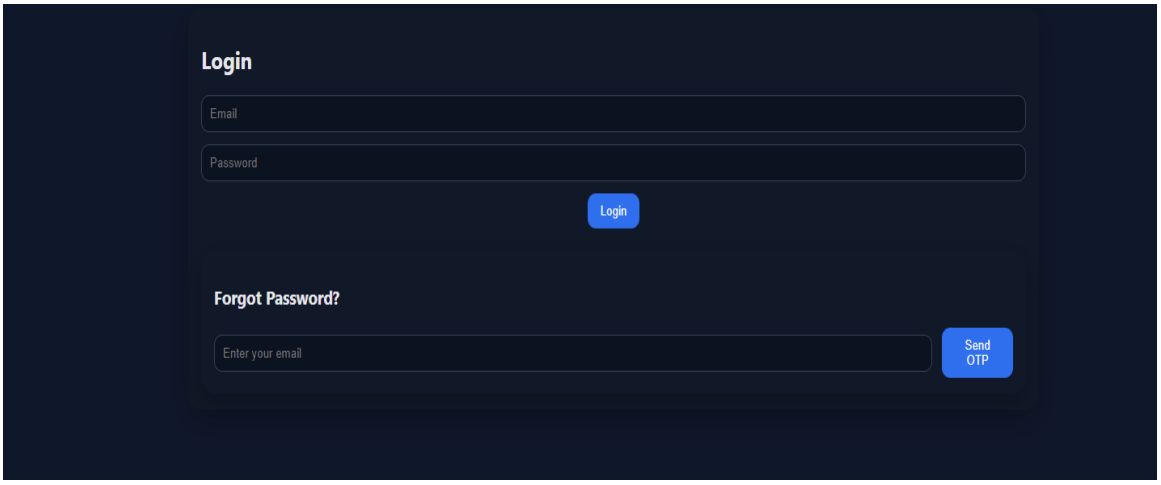
a. Landing Page – Clean UI with Sign Up / Login.



b. Signup Page – Name, Email, Password

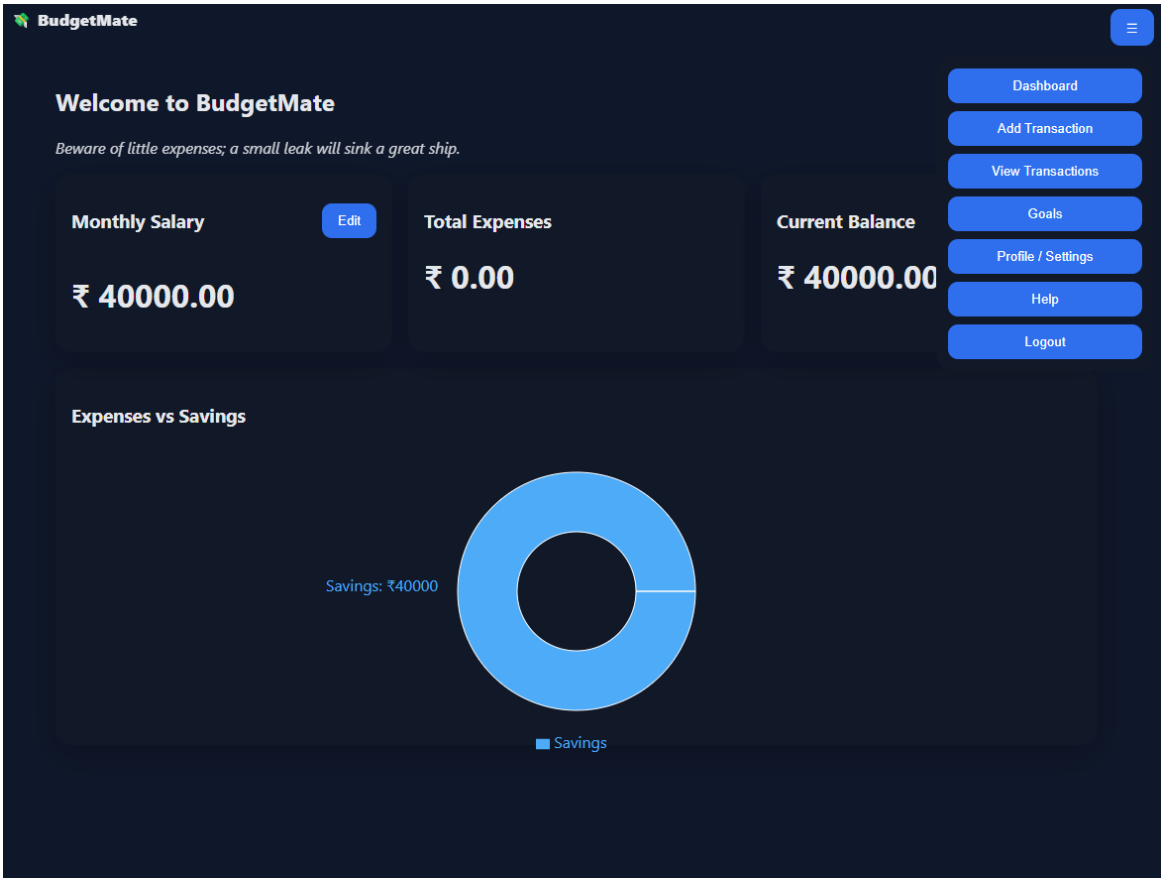


c. Login Page – Email, Password, Forgot Password

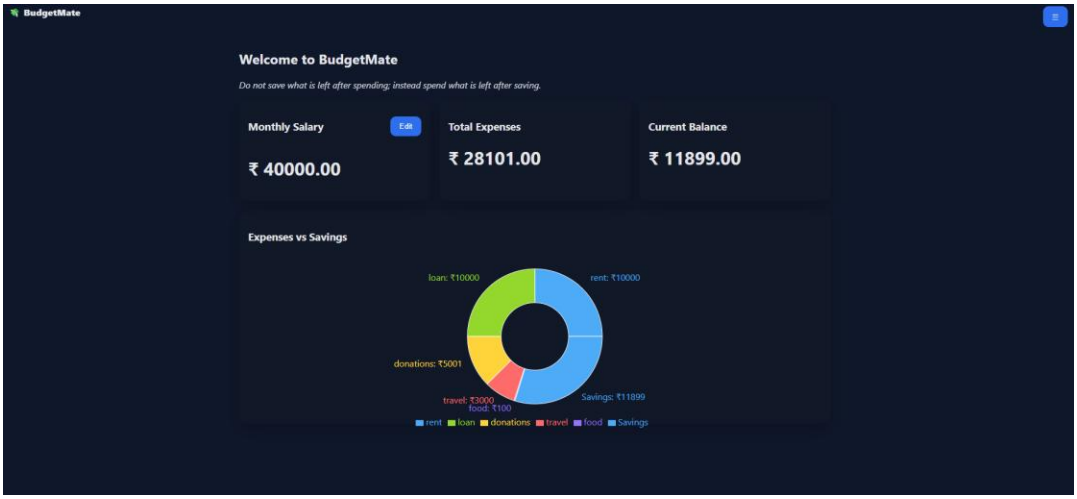


d. Dashboard – Salary, Expenses, Balance, Pie Chart.

2. Before performing transactions

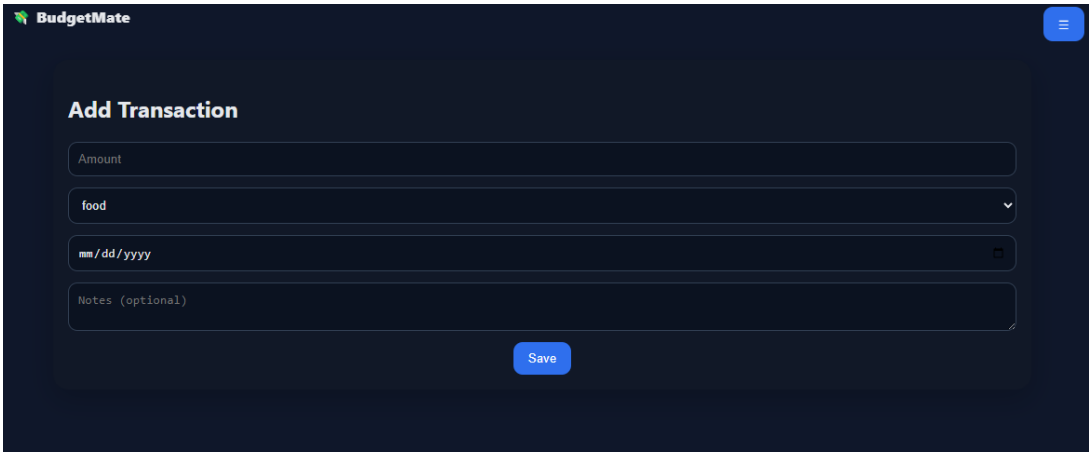


3. After Performing Transactions

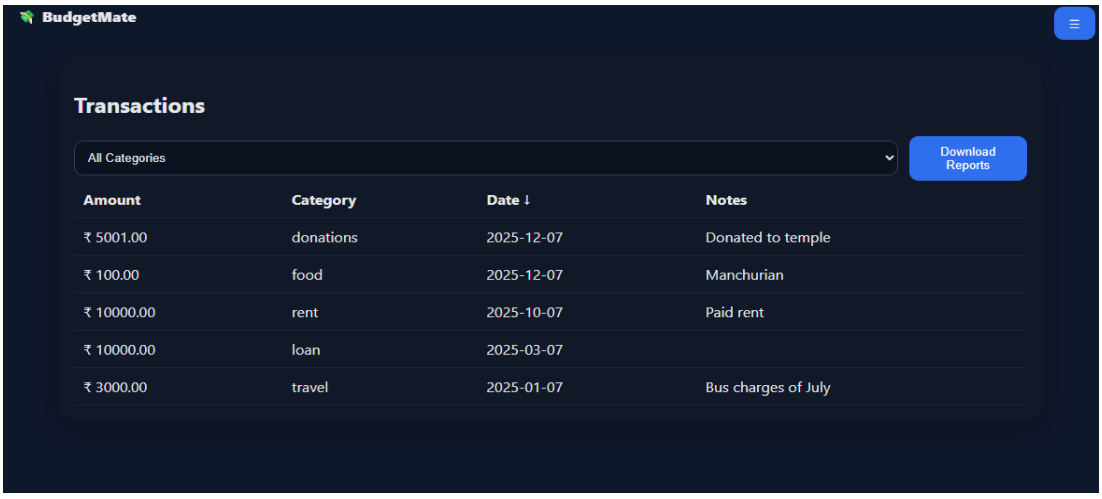


e. Transactions Page – List, filter, sort, and export.

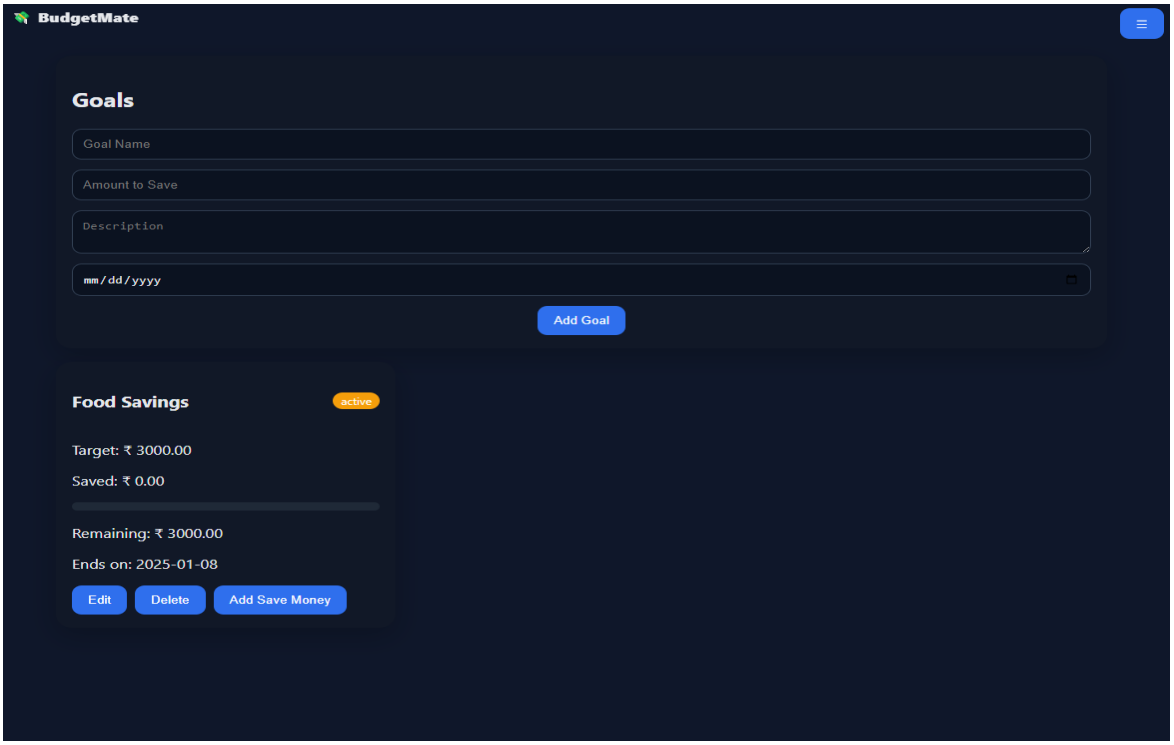
1. Add Transaction Page:



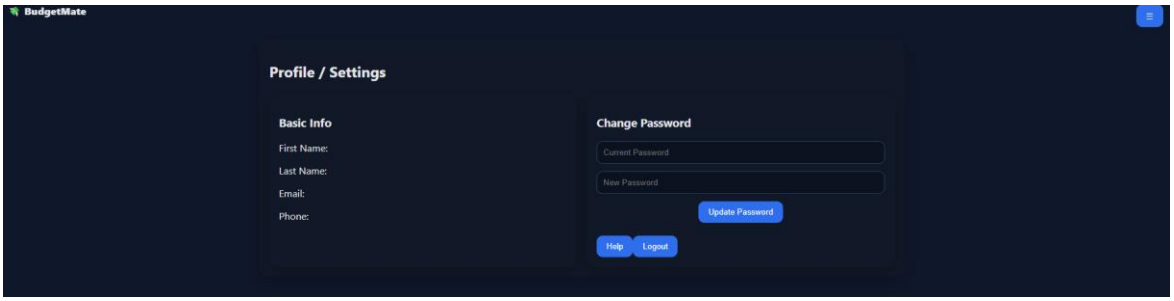
2. View Transaction Page:



f. Goals Page – Create, track, edit, delete, and add savings.



g. Profile Page



4. Implementation

The implementation phase involves translating the system design into a working software product. The BudgetMate application was developed using **React.js** for the frontend, **Node.js with Express.js** for the backend, and **MySQL** as the database. This section describes the key aspects of implementation.

1. Technology Stack

- **Frontend:** React.js, HTML5, CSS3, JavaScript (with inline styling for components).
- **Backend:** Node.js, Express.js for REST API development.
- **Database:** MySQL (Relational DBMS).
- **Authentication:** JWT (JSON Web Token) for secure login sessions.
- **Libraries & Tools:**
    - Axios for API requests
    - Recharts for visualization (Pie-Chart, Bar-Chart)
    - bcrypt.js for password hashing
    - .env for environment variables

2. Backend Implementation

The backend is responsible for handling authentication, goal management, transactions, and summary generation.

- **Authentication API:**
    - /login – verifies user credentials and returns JWT token.
    - /signup – registers new users with hashed passwords.
    - Middleware auth.js ensures secure access.
- **Goal Management API:**
    - POST /goals – create a new goal.
    - GET /goals – fetch user goals with progress status.
    - PUT /goals/:id – update goal details if not expired/achieved.
    - DELETE /goals/:id – delete goal if active.
    - POST /goals/:id/save – add savings to a goal.
- **Transactions API:**
    - Handles expense and savings records linked to goals.
    - Provides category-wise breakdown for dashboard.

3. Frontend Implementation

The frontend was built using **React.js functional components**.

- **Landing Page:**
  - Welcomes users with styled headings, buttons for login & signup.
  - Minimal and responsive design.
- **Dashboard Page:**
  - Displays Salary, Total Expenses, Current Balance.
  - **Pie-Chart (Expenses vs Savings):** Attractive, theme-based colours.
  - Editable Salary Modal.
  - Motivational quotes auto-rotating.
- **Goals Page:**
  - Users can **Add Goals**, **Edit**, **Delete**, and **Add Save Money**.
  - Progress bar increases dynamically when money is saved.
  - Status updates automatically (active, achieved, expired).
- **Authentication Pages:**
  - Login and Signup forms with validation.

4. Database Implementation

The database was designed in **MySQL** using 3 key tables and some supporting ones.

1. **Users Table** – stores user credentials and profile details.
2. **Goals Table** – stores goal details such as name, amount, end date, and description.
3. **Transactions Table** – stores expenses and savings transactions, linked to goals.

🔗 Relationships:

- One User → Many Goals.
- One Goal → Many Transactions.

5. Security Implementation

- Passwords are encrypted using **bcrypt** before storage.
- Authentication is handled using **JWT tokens** to protect routes.
- Role-based access (Admin/User) ensures only authorized users can manage the system.

5. Testing

The testing phase ensures that the BudgetMate system functions as expected, is reliable, and meets user requirements. Both **functional testing** and **non-functional testing** were performed to validate the application.

1. Testing Objectives

- Verify that all modules (Authentication, Goals, Transactions, Dashboard) work correctly.
- Ensure data integrity between frontend, backend, and database.
- Check user experience and responsiveness on multiple devices.
- Validate security mechanisms like password hashing and JWT authentication.

2. Types of Testing

a. Unit Testing

- Each API endpoint was tested individually.
- Example:
    - POST /login tested with valid and invalid credentials.
    - POST /goals/:id/save tested with valid amount, invalid (negative) values.

b. Integration Testing

- Verified seamless interaction between **frontend (React)** and **backend (Express API)**.
- Example: Adding savings in the **Goals Page** immediately reflects in **Dashboard Pie Chart**.

c. System Testing

- The entire application was tested as a whole.
- Scenarios:
    - User creates a goal → Adds savings → Dashboard updates correctly.
    - User tries to edit an **expired/achieved** goal → Error message shown.
    - Download report feature → Generates Excel file correctly.

d. Security Testing

- Ensured unauthorized users cannot access protected routes without a valid JWT token.
- Password stored in database in encrypted form only.

e. Usability Testing

- Verified UI responsiveness on desktop and mobile screens.
- Checked intuitive navigation (Navbar links, buttons, modals).

3. Test Cases

Test Case ID	Description	Input	Expected Output	Result
TC01	User Login	Valid credentials	Redirect to Dashboard	☑ Passed
TC02	User Login	Invalid credentials	Error message	☑ Passed
TC03	Add Goal	Goal details	Goal saved in DB	☑ Passed
TC04	Edit Goal	Expired goal	Error: Cannot edit	☑ Passed
TC05	Save Money	Amount = 500	Progress increases, Dashboard savings updated	☑ Passed
TC06	Delete Goal	Active goal	Goal removed from DB	☑ Passed
TC07	Transactions Export	Click download	Excel file generated	☑ Passed
TC08	JWT Auth	No token in request	Error: Unauthorized	☑ Passed

4. Testing Tools

- **Postman** – API endpoint testing.
- **MySQL Workbench** – Database validation.
- **Jest / Manual Testing** – Unit & functional testing.
- **Browser Developer Tools** – UI responsiveness testing.

5. Test Results

- All core functionalities passed successfully.
- System is stable, secure, and ready for deployment.
- Minor UI adjustments were made to improve chart labels and modal usability.

6. Results and Discussion

The development and implementation of *BudgetMate* provided several important outcomes. This section highlights the system's effectiveness, user experience, and limitations observed during testing and deployment.

1. Results

- **Functional Results**
  - **User Authentication**: Secure login and signup using JWT and password hashing works seamlessly.
  - **Goals Module**: Users can successfully create, edit, delete goals, and add savings. Progress bar updates dynamically.
  - **Transactions Module**: Users can add, filter, sort transactions, and download reports in Excel format.
  - **Dashboard Module**: Displays a summary of salary, expenses, balance, and a visually appealing pie chart of expenses vs. savings.
  - **Reports**: Export functionality works and provides a structured Excel file for offline analysis.

- **Non-Functional Results**
  - **Usability**: The interface is clean, responsive, and easy to navigate.
  - **Performance**: Database queries execute quickly, ensuring smooth user interactions.
  - **Security**: Passwords are stored in hashed format, and routes are secured with JWT.
  - **Scalability**: The modular design supports future additions (e.g., budget recommendations, AI insights).

2. Discussion

1. **Achievement of Objectives**
   - All key project objectives (expense tracking, goal setting, savings visualization, and reporting) were successfully achieved.
   - The integration between **Goals** and **Dashboard** ensures real-time updates of savings, making the system practical and user-friendly.

2. **User Experience**
   - The **progress bar in goals** and **pie chart in dashboard** provide intuitive financial insights.
   - Inline modals for editing goals and adding savings improved user interaction.

3. **Challenges Faced**
    - Handling date formats between frontend and backend required extra adjustments (e.g., removing timestamps).
    - Aligning pie chart labels within the container was challenging but resolved with styling improvements.

4. **Limitations**
    - Currently, there is no feature for **multi-user sharing** (e.g., family/group budgets).
    - Predictive financial analytics (e.g., monthly forecast) are not yet implemented.
    - Mobile UI could be further enhanced with gesture-based interactions.

5. **Future Enhancements**
    - AI-powered **budget recommendations**.
    - Notifications/reminders for goal deadlines.
    - Dark mode for better accessibility.
    - Multi-language support for wider usability.

7. Conclusion and Future scope

1. Conclusion

The development of **BudgetMate – A Personal Budget Management System** has successfully demonstrated the design and implementation of a practical, user-friendly, and secure financial management tool.

The system enables users to:

- Track their **income, expenses, and savings** efficiently.
- Create and manage **financial goals**, with real-time progress monitoring.
- Visualize spending patterns and savings through **interactive dashboards** (progress bar, pie chart).
- Export reports in **Excel format** for offline use and analysis.

Key achievements include:

- Seamless **integration between goals and dashboard**, ensuring accurate reflection of savings.
- A **responsive UI** with inline forms and modals that enhance usability.
- A **secure backend** using authentication, hashed passwords, and modular API design.

Overall, *BudgetMate* helps users build financial discipline, make informed decisions, and achieve long-term financial stability.

2. Future Scope

Although BudgetMate fulfils its current objectives, several enhancements can make the system more robust and intelligent in the future:

1. **AI & Predictive Analytics**
    - Introduce AI-driven insights to recommend personalized budgets and predict monthly expenses.
2. **Notifications & Reminders**
    - Provide email/SMS/app notifications for bill payments, goal deadlines,

or overspending alerts.

3. **Multi-User & Family Accounts**
    - Allow shared budgeting for families, groups, or small businesses.
4. **Gamification**
    - Add reward points or badges when users achieve goals, encouraging savings habits.
5. **Mobile App Version**
    - Extend BudgetMate into Android/iOS applications for easier on-the-go access.
6. **Multi-Language Support**
    - Provide localization to support users in different regions and languages.
7. **Advanced Reporting**
    - Enable graphical trend analysis (bar charts, line charts) for long-term financial tracking.

8. Bibliography and References

In preparing this project report on **BudgetMate – A Personal Budget Management System**, various books, research papers, articles, and online resources were consulted. These references helped in understanding the concepts of system design, database management, and financial management applications.

Books

1. Rajaraman, V. – *Fundamentals of Computers*, Prentice Hall of India.
2. Abraham Silberschatz, Henry Korth, S. Sudarshan – *Database System Concepts*, McGraw Hill.
3. Roger S. Pressman – *Software Engineering: A Practitioner's Approach*, McGraw Hill.
4. Ian Sommerville – *Software Engineering*, Pearson Education.

Research Papers / Journals

1. International Journal of Computer Applications (IJCA) – *Personal Finance Management Systems: A Review*.
2. IEEE Xplore Digital Library – Articles on **Budgeting Tools** and **Financial Applications**.

Websites

1. https://reactjs.org/ – Official React documentation.
2. https://expressjs.com/ – Express.js framework documentation.
3. https://www.mysql.com/ – MySQL official documentation.
4. https://recharts.org/ – Documentation for pie charts and graphs used in the dashboard.
5. https://www.npmjs.com/ – Node.js and package dependencies.

Other Resources

- Lecture notes, project guidelines, and practical references provided during coursework.
- Discussions with mentors and peers that contributed to system requirements and design.