Project Tracker

Project Report

By
Poonam Shedge – AF04957504

Index

Acknowledgement

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

Name	Enrollment No
Poonam Shedge	**AF04957504**

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

The **Project Tracker** is a web-based application designed to simplify project management by providing a centralized platform to monitor tasks, deadlines, and team performance. It allows project managers to create and manage multiple projects, assign tasks to team members, track progress in real-time, and generate detailed reports. The system ensures efficient communication, task prioritization, and timely completion of projects, reducing delays and enhancing productivity. By integrating task management, progress tracking, and reporting in a single platform, the Project Tracker serves as a comprehensive tool for organizations aiming to streamline their project workflows and achieve project goals effectively.
.

1. Introduction

In today's fast-paced business environment, organizations often manage multiple projects simultaneously, each with its own set of tasks, deadlines, and resources. Managing these projects manually or using fragmented tools can lead to delays, miscommunication, and inefficiencies. To address these challenges, the **Project Tracker** has been developed as a comprehensive software solution for effective project management.

Project Tracker is designed to provide a centralized platform where project managers and teams can plan, monitor, and execute projects with ease. It allows users to create new projects, define tasks, assign responsibilities, and set priorities. Each task can be tracked in real-time, providing a clear view of its progress, status, and completion percentage. This ensures that team members are accountable for their work and deadlines are met consistently.

The system also offers robust reporting and analytics features, enabling managers to generate detailed progress reports, monitor resource utilization, and analyze team performance. By providing insights into project workflows, Project Tracker helps organizations make informed decisions, identify potential bottlenecks, and optimize overall productivity.

Furthermore, Project Tracker enhances team collaboration by providing a unified platform for communication, notifications, and updates. It ensures that all stakeholders, including managers, team members, and clients, stay informed about project developments. By reducing manual effort, improving transparency, and streamlining workflows, Project Tracker empowers organizations to achieve their project goals efficiently and maintain a competitive edge in the market.

In summary, Project Tracker is not just a tool but a strategic solution that transforms how projects are planned, executed, and monitored, making project management more organized, efficient, and productive.

The primary objective of the present work is to design and develop a **Project Tracker** system that facilitates efficient project management within an organization. The system aims to address common challenges such as missed deadlines, lack of task clarity, and inefficient communication among team members.

The specific objectives include:

1. **Effective Project Monitoring:** To provide real-time tracking of all ongoing projects, ensuring that managers and team members have up-to-date information on progress and task completion.
2. **Task Assignment and Management:** To enable the assignment of tasks to individual team members with clearly defined responsibilities and deadlines.
3. **Deadline Management:** To ensure timely completion of tasks and projects by setting reminders, alerts, and notifications for pending deadlines.
4. **Reporting and Analytics:** To generate detailed progress reports and performance analytics for better decision-making and resource planning.
5. **Enhanced Collaboration:** To facilitate improved communication among team members and stakeholders through a centralized platform.
6. **Prioritization of Tasks:** To help teams identify high-priority tasks and allocate resources efficiently for maximum productivity.
7. **Centralized Project Information:** To maintain all project-related data in one platform for easy access, retrieval, and documentation.
8. **Reduction of Errors and Delays:** To minimize manual errors, avoid duplication of work, and streamline project workflows.

The overall goal is to develop a user-friendly and scalable system that assists organizations in **efficiently planning, executing, and completing projects on time**, thereby improving productivity, accountability, and operational effectiveness

System Analysis

In most organizations, managing multiple projects simultaneously is a complex task. Traditional project management methods, whether manual or using basic tools, often face several challenges:

- **Lack of Centralized Information:** Project details, task assignments, and progress updates are scattered, making it difficult for managers and team members to access accurate information.
- **Inefficient Task Assignment:** Assigning tasks manually or via emails leads to confusion, delays, and unclear responsibilities.
- **Difficulty in Tracking Progress:** Monitoring multiple projects and individual tasks in real-time is challenging, which may result in missed deadlines and delayed project completion.
- **Poor Communication and Collaboration:** Team members and stakeholders often face communication gaps, affecting project coordination and workflow.
- **Limited Reporting and Analytics:** Generating reports on project status, resource utilization, and performance is time-consuming and error-prone.
- **Resource Management Issues:** Without proper tracking, resources may be underutilized or overburdened, leading to inefficiency.

To identify the problems and limitations in the existing project management methods.

- To assess the need for a new system to improve efficiency and productivity.
- To evaluate technical, operational, and financial feasibility of the Project Tracker system.

Centralized Project Management:

All project-related information is stored in one platform, making it easy to access, update, and manage.

- **Efficient Task Assignment:**

Tasks can be assigned clearly to team members with deadlines and priorities, reducing confusion and overlapping responsibilities.

- **Real-Time Progress Tracking:**

Monitor the status of projects and tasks instantly, enabling managers to identify delays and take corrective actions quickly

The **Proposed System** aims to overcome the limitations of existing project management methods by providing a **centralized, automated, and user-friendly platform** to manage projects, tasks, and team collaboration efficiently. Users can browse articles, comment on news stories, and receive instant updates **Centralized Dashboard:**

- Provides a comprehensive view of all ongoing projects, tasks, deadlines, and team performance in one place.

Project and Task Management:

- Create, update, and manage multiple projects.
- Assign tasks to team members with clear deadlines and priority levels.

Real-Time Progress Tracking:

- Track task completion and project status in real-time.
- Identify delays or bottlenecks quickly to take corrective actions.

18.1 Feasibility Study

A **Feasibility Study** is conducted to evaluate whether the proposed Project Tracker system can be successfully developed and implemented within the given constraints of technology, resources, and cost. It helps in identifying potential challenges and ensures that the system is practical, cost-effective, and beneficial for the organization**Types of Feasibility Analysis**

Technical Feasibility:

19 Determines whether the organization has the required hardware, software, and technical expertise to develop and maintain the Project Tracker system.
20 The system will use technologies like **React (frontend), MySQL (database), and Node.js/PHP (backend)**, which are readily available and widely supported.
21 Ensures that the system can handle multiple users, projects, and real-time updates effectively.

Economic (Financial) Feasibility:

22 Evaluates whether the benefits of implementing the Project Tracker outweigh the costs.
23 Costs include software development, hosting, maintenance, and training.
24 Benefits include improved productivity, timely project completion, reduced manual effort, and better resource utilization.

Operational Feasibility:

25 Assesses whether the system meets the operational needs of users and integrates well with current workflows.
26 Project Tracker is designed to be user-friendly, requiring minimal training for project managers and team members.
27 Supports improved communication, task tracking, and reporting, making it practical for daily project management operations.

Schedule Feasibility:

- Determines whether the system can be developed and implemented within the required timeframe.

a. Project Tracker is designed to follow a structured development timeline, ensuring timely deployment without delaying

1. 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

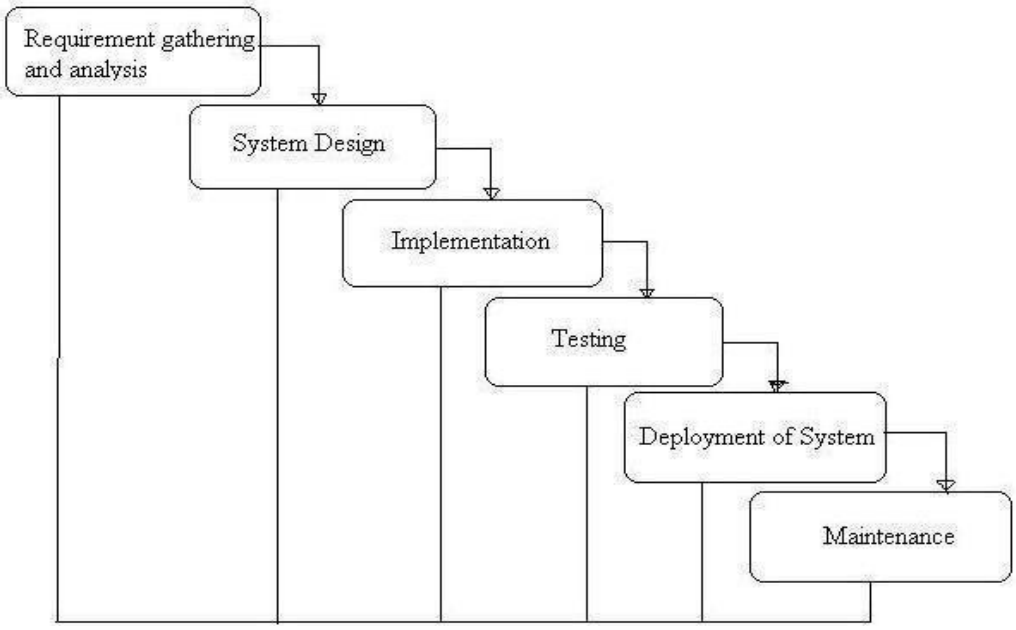
- Operating System: Windows/Linux/MacOS.
- Development Tools: Node.js, MySQL, React.
- Browser: Google Chrome / Mozilla Firefox.

2. Proposed System

The proposed *ProjectTracker* 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 **Node.js)**.
    - 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.

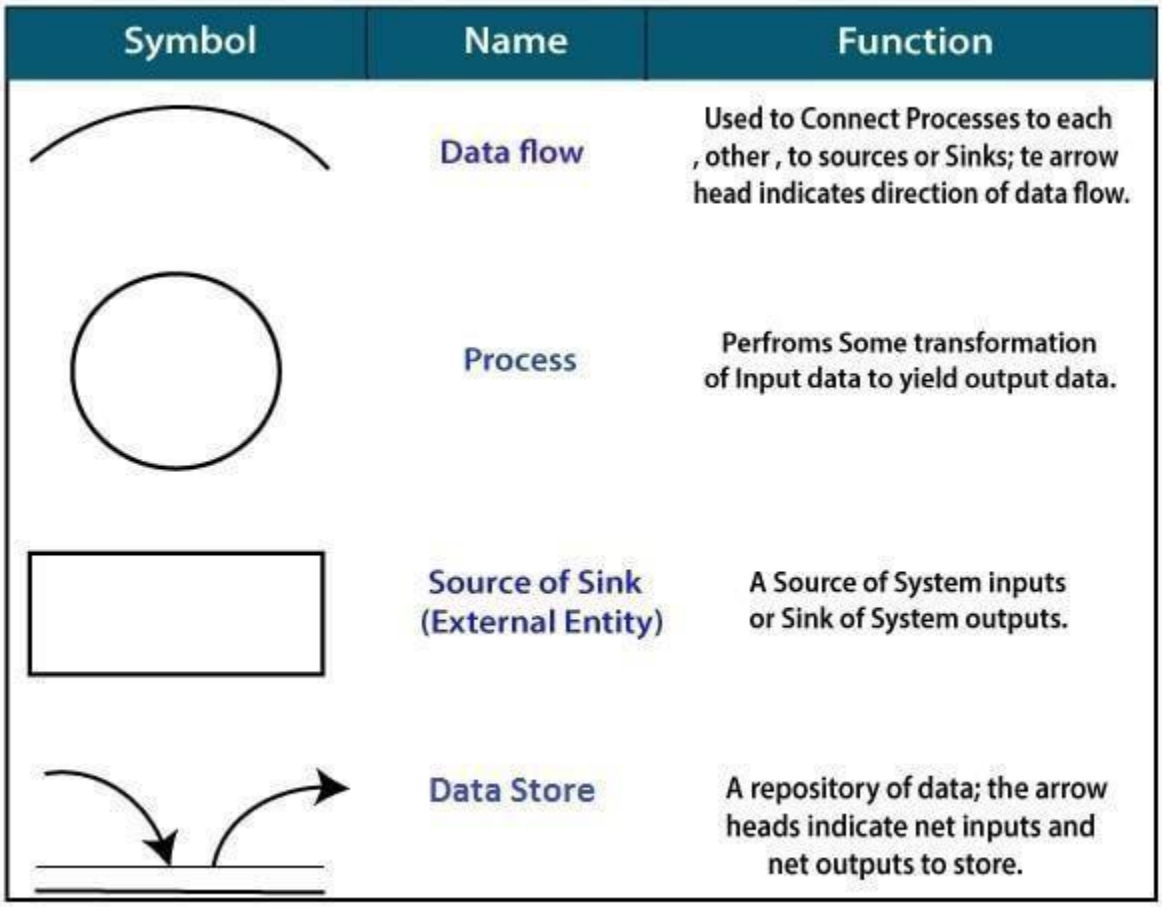
It shows how data enter and leaves the system, what changes the information, and where data is stored.

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:



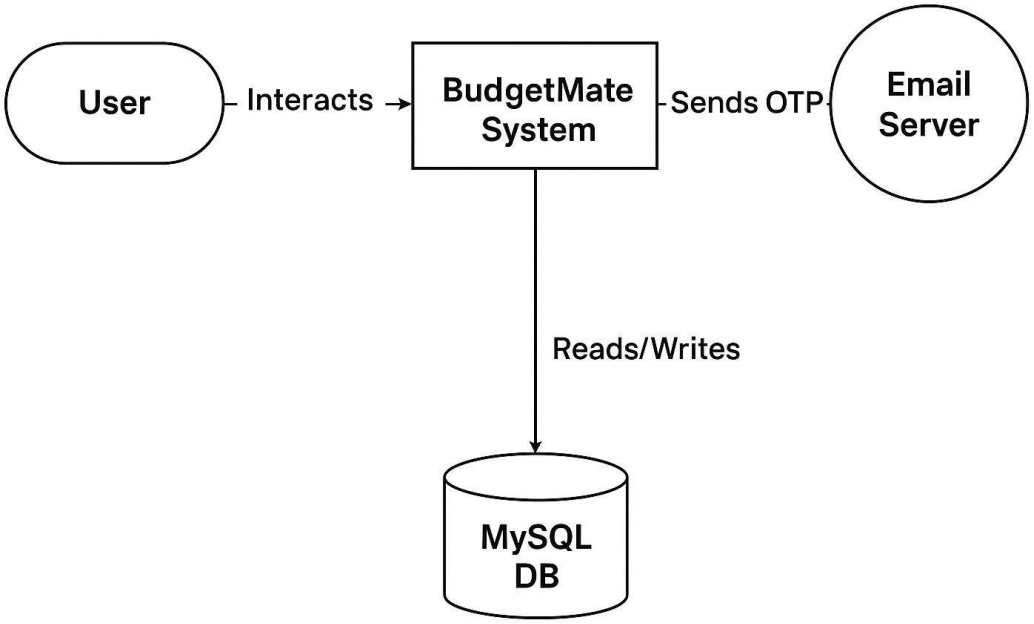
Symbols for Data Flow Diagrams

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 ProjectTracker

The **Zero-Level DFD**, also called the **Context Diagram**, represents the entire **ProjectTracker 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).

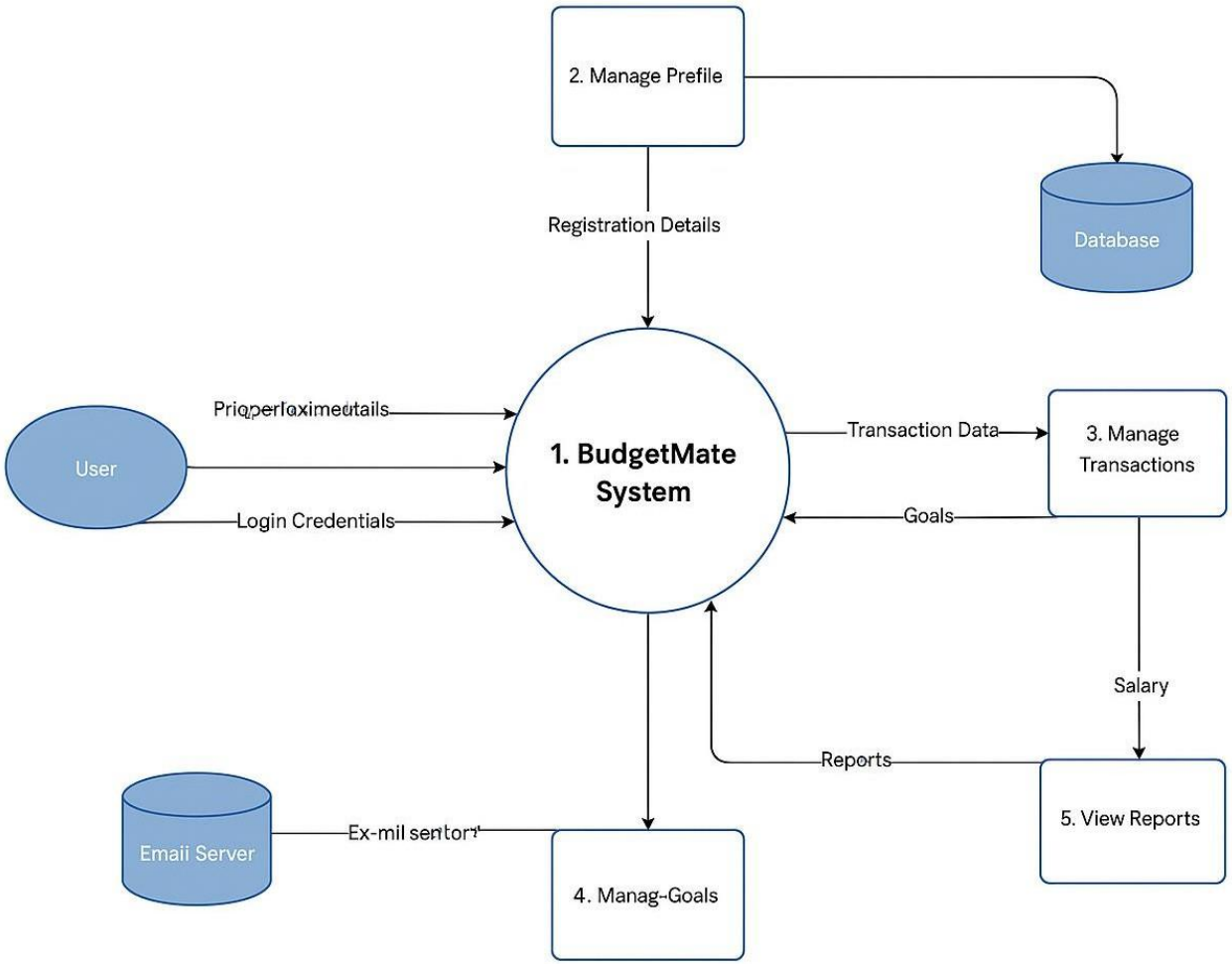
Central Process: ProjectTracker System

- The **ProjectTracker 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 **ProjectTracker 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.

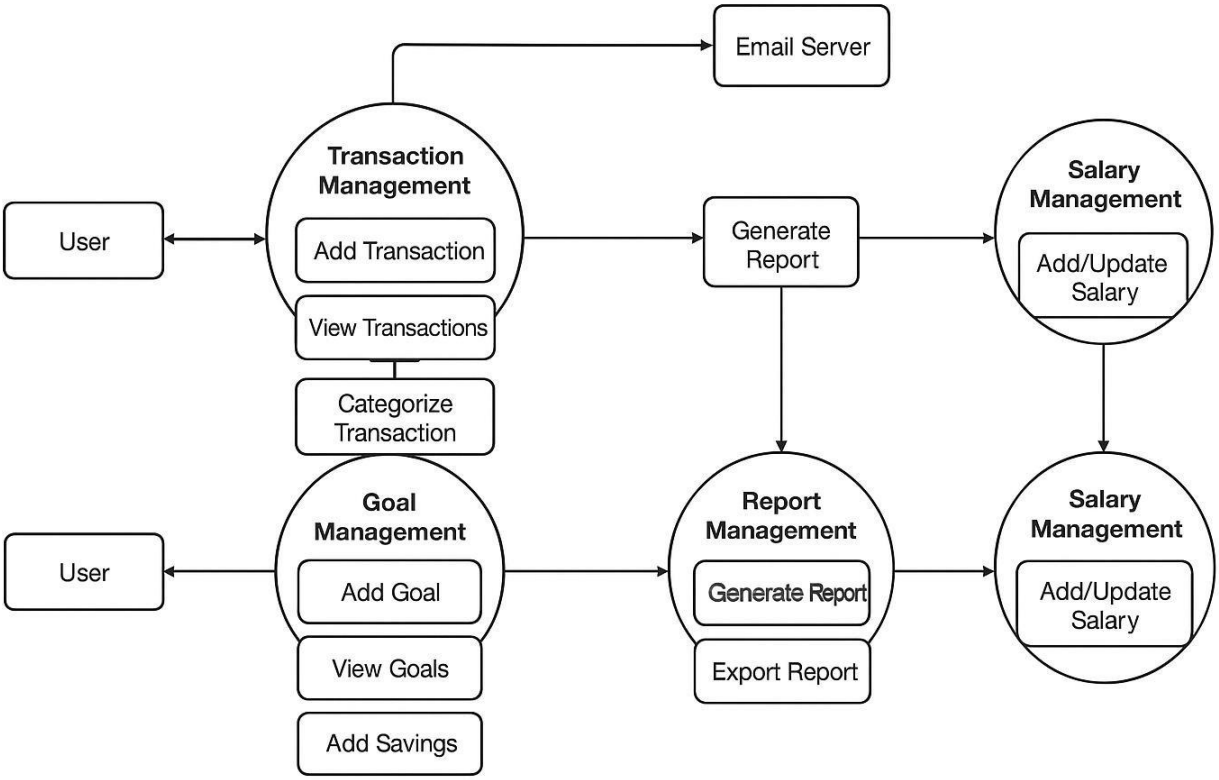
External Entities

- **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.

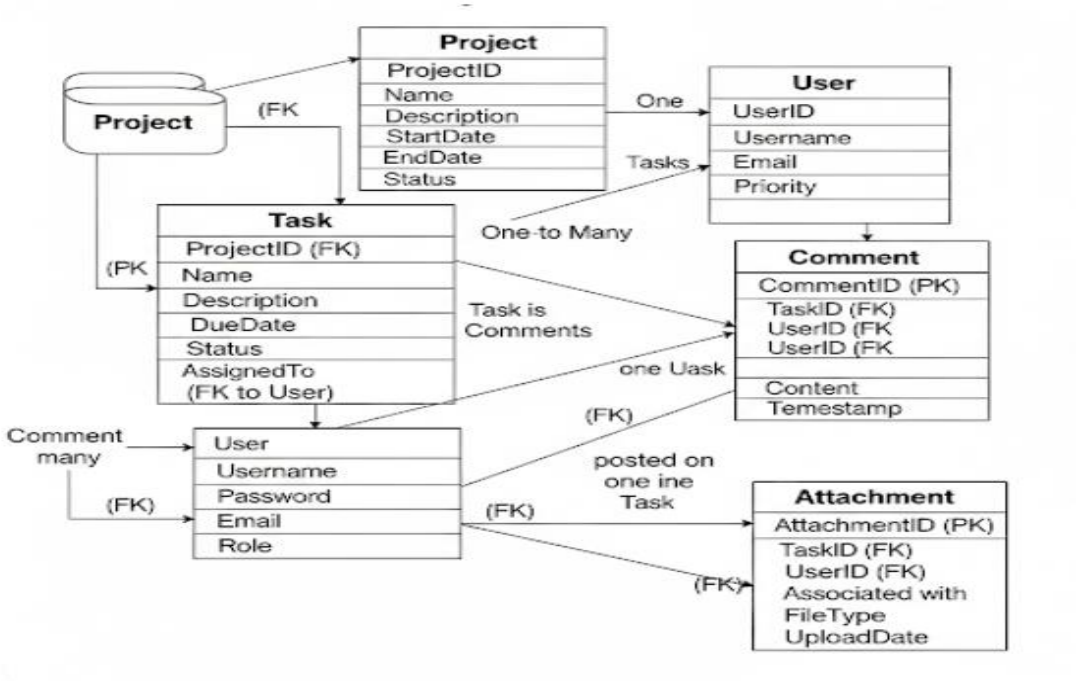
5. Dashboard & Visualization

- **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 **ProjectTracker** 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.

2. 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 ProjectTracker:

- 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 ProjectTracker 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.

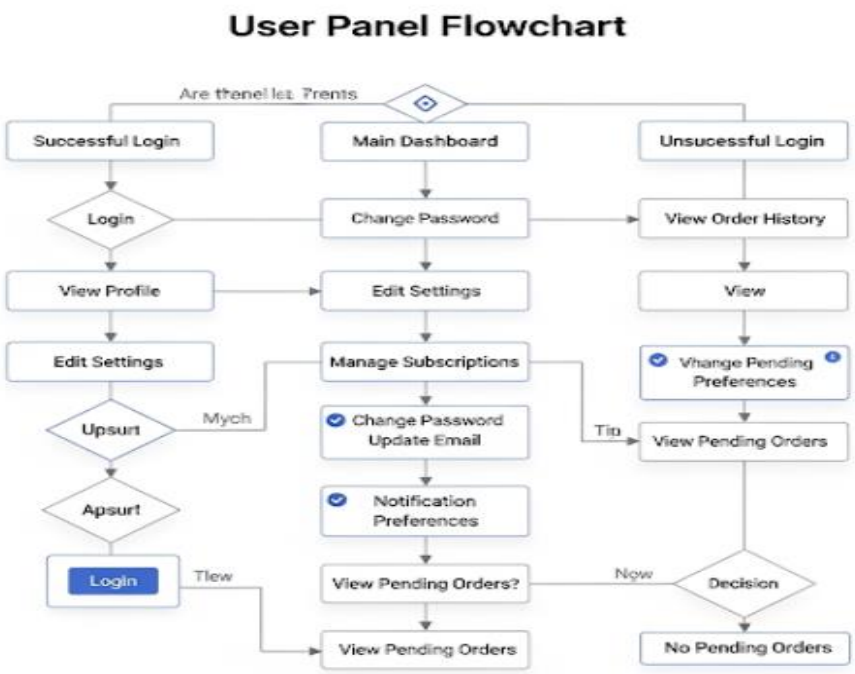
One goal → Many savings transactions (via goal_id).

One user → One salary record.

C) Procedural Design

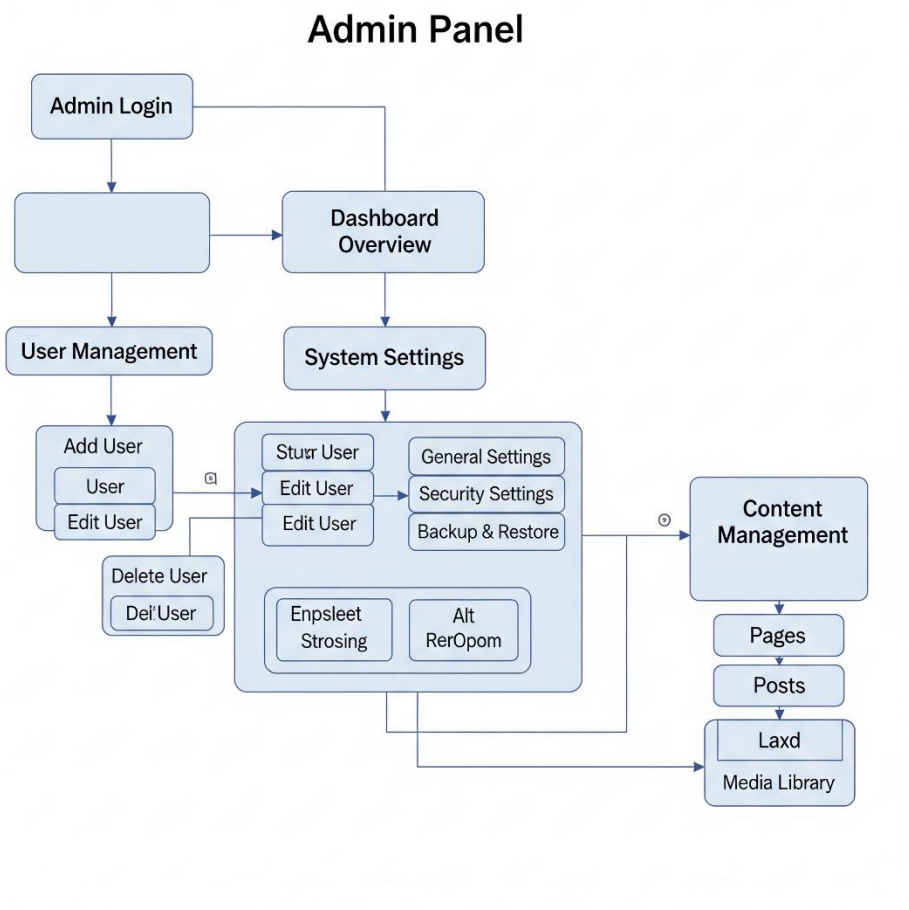
1. User Panel Design

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

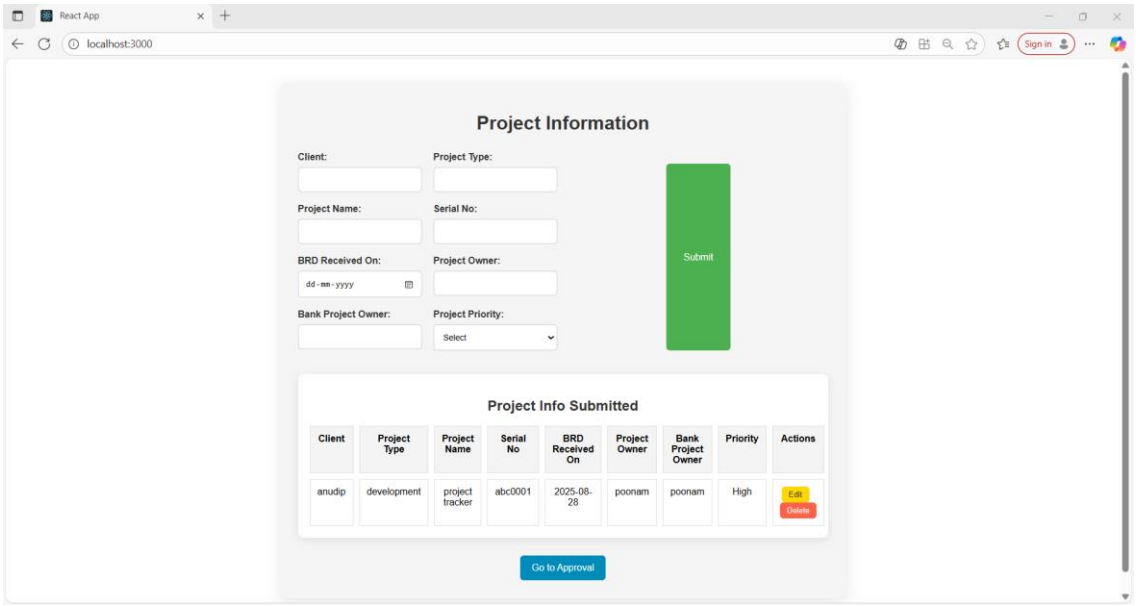


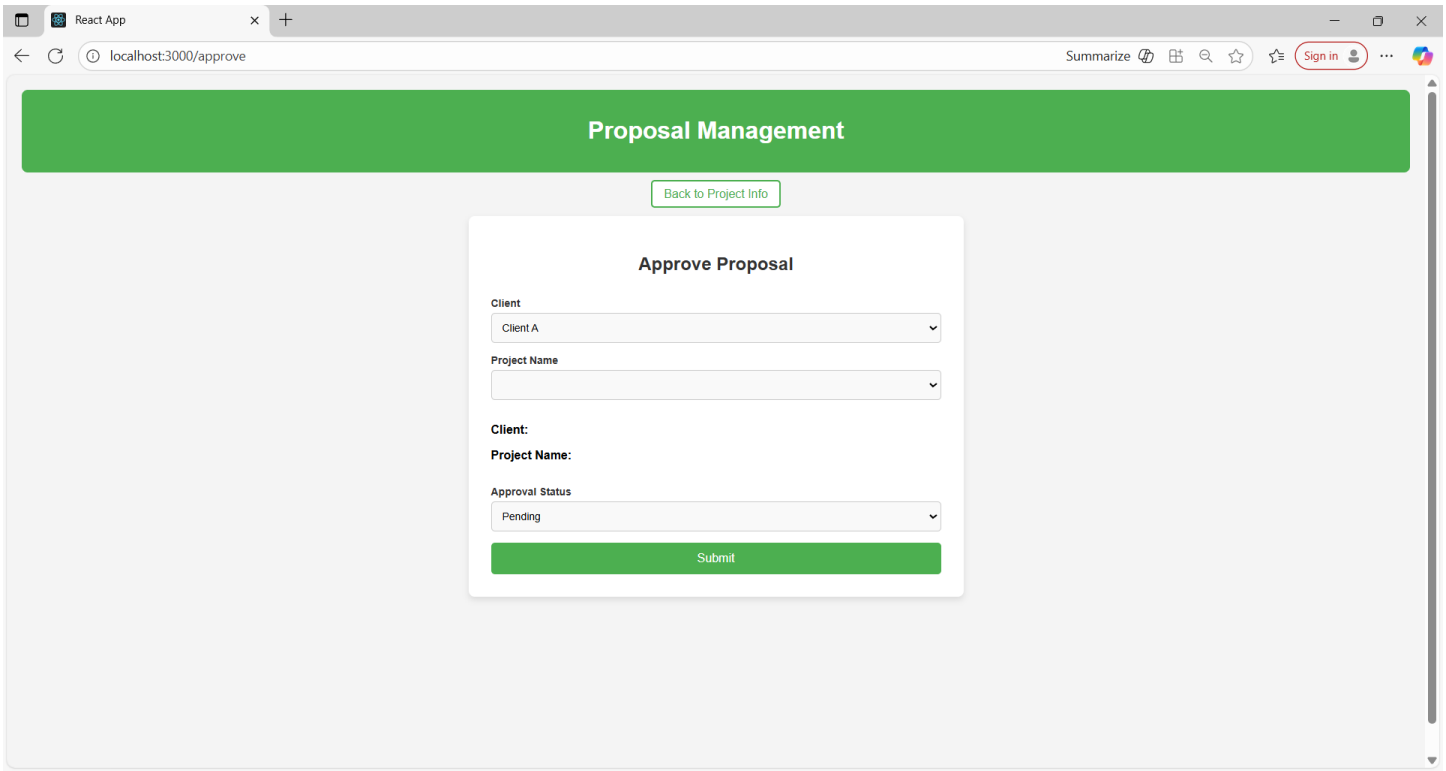
2. Admin Panel Design

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)





:

3. Implementation

The implementation phase involves translating the system design into a working software product. The ProjectTracker 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.

4. Testing

The testing phase ensures that the ProjectTracker 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.

3. Testing Tools

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

4. 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.

5. Results and Discussion

The development and implementation of *ProjectTracker* 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.

6. Conclusion and Future scope

1. Conclusion

- A project tracker is an essential tool for effective project management. It provides a centralized platform for teams to plan, execute, and monitor projects from start to finish. The system helps in breaking down complex projects into manageable tasks, assigning responsibilities, and setting clear deadlines.
- Key features of a project tracker include:
- **Task Management:** Creating, assigning, and tracking the status of individual tasks.
- **Collaboration:** Facilitating communication and sharing of files and comments among team members.
- **Reporting:** Generating reports and charts to visualize project progress and identify potential bottlenecks.
- **Resource Management:** Allocating and monitoring resources to ensure efficient utilization.
- By using a project tracker, organizations can improve team coordination, increase transparency, and ensure that projects are completed on time and within budget.
-

7. Bibliography and References

In preparing this project report on **ProjectTracker – 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.