**PERSONAL FINANCE PLANNER**

**JAVA MINI PROJECT REPORT**

***Submittedby :***

# JASHAREEN J 231801066

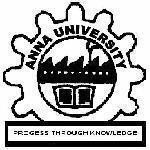
# KRITHIKA MA 231801086

# 

*In partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY IN**

## ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



**RAJALAKSHMI ENGINEERING COLLEGE,**

**ANNA UNIVERSITY, CHENNAI: 602 105**

**NOV 2024**

**RAJALAKSHMI ENGINEERING COLLEGE**

**CHENNAI 600 025**

# BONAFIDE CERTIFICATE

## Certified that this report title “PERSONAL FINANCE PLANNER” is

## the Bonafide work of JASHAREEN J (231801066) and KRITHIKA MA

## (231801086) who carried out the mini project work under my supervision.

Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

|  |  |
| --- | --- |
|  |  |
| SIGNATURE | SIGNATURE |
| Mr.JR.Gnanasekar, |  |
| HEAD OF THE DEPARTMENT,  Professor,  Department of AI&DS,  Rajalakshmi Engineering College Chennai – 602 105. | SUPERVISOR,  Assistant Professor,  Department of AI&DS,  Rajalakshmi Engineering College, Chennai – 602 105. |

Submitted for the JAVA Mini project review held on

Internal Examiner External Examiner

**ACKNOWLEDGEMENT**

Initially I thank the Almighty for being with us through every walk of my life and showering his blessings through the endeavor to put forth this report.

My sincere thanks to our Chairman **Mr. S. MEGANATHAN, M.E., F.I.E.,** and our Chairperson **Dr. (Mrs.)THANGAM MEGANATHAN, M.E., Ph.D.,** for providing me

the requisite infrastructure and sincere endeavoring educating me in their premier institution.

My sincere thanks to **Dr.S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time.

I express my sincere thanks to **Mr.JM.Gnanasekar.,M.E.,Ph.D.,** Head of the

Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. I convey my sincere and deepest gratitude to our internal guide, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally I express my gratitude to my parents and classmates for their moral support and valuable suggestions during the course of the project.

**ABSTRACT**

The Personal Finance Planner System is a user-friendly application designed to assist individuals in effectively managing their personal finances by tracking income, expenses, and savings. The system addresses common challenges in financial management, such as lack of expense tracking, unplanned spending, and difficulty in calculating savings. By providing a centralized platform, the system enables users to organize and analyze their financial activities with ease and accuracy.

The platform incorporates essential features tailored for individual users. Users can record their total income, input expense details (such as expense name and amount), and receive an automatic calculation of total expenses and remaining savings. A detailed breakdown of expenses is also provided, allowing users to gain insights into their spending habits and make informed financial decisions.

The project utilizes MySQL as the database management system, ensuring reliable and secure storage of income and expense details. The Java programming language is employed for backend development, providing robust and efficient processing of financial data. Additionally, VS Code serves as the development environment, with Maven used for managing dependencies, such as the MySQL JDBC driver.

**Key features of the system include:**

1.Expense Management: Users can add multiple expenses, categorizing them with a name and amount.

2.Savings Calculation: The system computes total savings by subtracting recorded expenses from the user’s total income.

3.Expense Summary: A clear and detailed breakdown of expenses is displayed alongside income and savings data.

4.Database Integration: All financial data is securely stored in a MySQL database for persistence and future reference.

**Background and Need for Automation**

In today’s fast-paced world, managing personal finances has become a complex and time-consuming task. Traditionally, individuals track their income, expenses, and savings manually or with the help of basic spreadsheets. This process is prone to errors, time-consuming, and lacks the ability to provide an overview of one's financial health in real-time. As a result, many people struggle with budgeting effectively, leading to unnecessary spending or failure to meet financial goals.

Automation in personal finance management addresses these challenges by providing an efficient, accurate, and real-time solution. By automating the tracking of expenses, income, and savings, individuals can easily monitor their financial status, make informed decisions, and plan their finances better. This reduces the chances of human error, improves financial discipline, and helps in achieving financial goals more effectively.

The need for automation becomes even more crucial when handling large amounts of financial data, where manual tracking becomes cumbersome and inefficient. With the advancement in technology, creating an automated system that integrates seamlessly with databases and provides real-time updates has become feasible and necessary

**Core Components of the Personal Finance Planner (PFP)**

The Personal Finance Planner (PFP) is designed to provide a holistic approach to managing personal finances, with key components that interact seamlessly to ensure effective financial tracking and management. Below are the core components of the PFP:

1. **User Authentication System**  
   The PFP includes a secure user authentication system that ensures only authorized users can access their financial data. This component handles user registration, login, and password management, ensuring data security and privacy. The system supports basic features like password reset and user profile management.
2. **Income Management**  
   The income management component allows users to record and track their income sources. Users can enter their salary, freelance earnings, or other income streams. The system calculates total income based on the entered data and provides an overview of income history.
3. **Expense Tracking**  
   This component is central to the Personal Finance Planner. It allows users to input and categorize their expenses (e.g., groceries, rent, entertainment, utilities). The system tracks the total expenses for a given period, compares them against income, and provides insights on spending patterns.
4. **Savings Calculation**  
   Based on the total income and recorded expenses, the savings component calculates the user's savings for a given period. It helps users evaluate whether they are saving enough and provides suggestions for increasing savings based on expense trends.
5. **Budgeting**  
   The budgeting component enables users to set up budget goals for different categories (e.g., groceries, entertainment, savings). It monitors actual spending against the set budget and provides feedback on whether users are staying within their financial limits.
6. **Expense Categorization**  
   Users can categorize their expenses into predefined categories (e.g., household, personal, transportation). This component ensures that spending is organized and helps users understand which areas they are overspending.
7. **Reports and Financial Summary**  
   The system generates financial reports that provide users with a clear picture of their income, expenses, savings, and budgeting over time. This component includes visual aids such as graphs and pie charts to offer an easy-to-understand summary of financial health.
8. **Database Integration**  
   A crucial backend component, the database stores user data securely. Using MySQL, the PFP ensures all financial data (income, expenses, savings, budget) is stored in an organized manner, enabling efficient querying and retrieval of financial records.
9. **User Interface (UI)**  
   The user interface provides an intuitive experience for users to interact with the system. Developed using HTML, CSS, and JavaScript, the UI ensures easy navigation through the system, allowing users to quickly input their financial data, view reports, and track progress toward their financial goals.
10. **Database Security**  
    The system includes measures to ensure data security and prevent unauthorized access. Sensitive financial information is stored securely in the database, and the application uses encryption techniques to safeguard user privacy.

These core components work together to provide users with an easy-to-use platform for managing their personal finances. Each component ensures that users can efficiently track, manage, and improve their financial situation through automated processes and data analysis.

**Features and Functionalities of the Personal Finance Planner**

1. User Authentication: Secure login, registration, and password reset functionalities for personalized access.
2. Income Management: Allows users to input and track their total income from various sources.
3. Expense Tracking: Users can record, categorize, and analyze their expenses, with visual summaries of spending patterns.
4. Savings Calculation: Automatically calculates savings by subtracting expenses from income, with options for goal setting.
5. Budgeting: Users can create and manage budgets, with notifications for budget limits.
6. Financial Reports: Generates detailed reports, with graphs and export options to PDF or CSV for offline use.
7. Security Features: Ensures data security through encryption and secure database management.

**Benefits of the Personal Finance Planner (PFP)**

1. Improved Budgeting: The ability to set and manage budgets ensures that users can control their spending, avoid overspending, and stay within their financial limits.
2. Savings Growth: The PFP calculates savings automatically, helping users stay on track to reach their financial goals, such as saving for emergencies or future investments.
3. Data Security: With secure data storage and encryption, the PFP ensures users' financial data is protected from unauthorized access, offering peace of mind.
4. Financial Reporting: The ability to generate detailed financial reports allows users to review.
5. Accessibility and Convenience: Being a web-based application, the PFP can be accessed anytime and from anywhere, making financial management easy and flexible.
6. Motivation and Goal Setting: The PFP enables users to set financial goals and track their progress, boosting motivation to save and improve financial health.
7. Cost-Effective: As a free or low-cost solution, the PFP provides a budget-friendly way to manage finances compared to hiring financial advisors.

**Future enhancements for the Personal Finance Planner (PFP) could include:**

Integration with Bank Accounts, Expense Prediction, Mobile Application, Investment Tracking, Financial Advice & Alert.

**TABLE OF CONTENTS**

**1.INTRODUCTION**

1.1 Overview

1.2 Problem statement

1.3 Objective

1.4 Module

**2.SURVEY OF TECHNOLOGIES**

2.1. Development Environment

2.2 Languages

2.2.1 SQL

2.2.2 Java

**3.REQUIREMENTS AND ANALYSIS**

3.1 Requirement Specification

3.2 Hardware and Software Requirements

3.3 Architecture Diagram

3.4 ER Diagram

3.5 Normalization

**4.PROGRAM CODE**

4.1 Backend design

4.2 Frontend design

**5.RESULTS AND DISCUSSION**

**6.CONCLUSION**

**7.REFERENCES**

**1.Introduction**

The Personal Finance Planner (PFP) is an innovative solution designed to help individuals manage their finances efficiently. In an era where financial management is increasingly becoming complex due to various income streams, expenses, and savings goals, this project seeks to provide a comprehensive and automated platform for tracking and planning personal finances. The Personal Finance Planner allows users to effortlessly record their income, track daily expenses, monitor savings, and generate financial reports, enabling them to make informed financial decisions.

This project aims to address the common problem of managing personal finances manually, which often leads to errors, confusion, and missed opportunities for saving and investing. By leveraging the power of technology, the PFP automates key financial management tasks such as calculating total expenses, savings, and categorizing expenses into specific categories (e.g., rent, utilities, entertainment). It provides an intuitive interface where users can easily input their financial data, view real-time summaries, and receive insights into their financial health.

The Personal Finance Planner integrates with a database to securely store user data and ensure data persistence over time. It employs technologies such as Java for backend logic, MySQL for database management, and a responsive user interface that ensures an engaging user experience. The project ultimately aims to provide a user-friendly, efficient, and secure platform for individuals to gain control over their financial future.

**1.1 Overview**

The main objective of the Personal Finance Planner System is to provide a user-friendly platform that helps individuals manage their financial resources effectively. This application allows users to track their income, record expenses, and calculate their savings. The system ensures that users can gain insights into their financial habits and make informed decisions about their spending and savings.

The project includes features such as expense recording, savings calculation, and a detailed summary of financial data. The software automatically stores details of income and expenses in a secure database and provides a clear breakdown for easy analysis. Each expense entry is uniquely identified and securely stored in the system’s database, allowing for efficient data management and future reference.

The Personal Finance Planner System requires users to enter their total income and expense details. The application computes total expenses, calculates remaining savings, and displays a summary of the user's financial information. The interface is simple, intuitive, and user-friendly, ensuring accessibility for all types of users. The data is securely managed, allowing for privacy and accuracy in calculations.

The Personal Finance Planner System is a powerful and flexible tool designed to deliver real, tangible benefits for individuals managing their personal finances. It streamlines financial planning processes and reduces the complexity involved in manual expense tracking.

The Personal Finance Planner System is suitable for individuals across various backgrounds, as it provides an integrated end-to-end financial management solution. It supports effective financial decision-making by offering relevant insights into income, expenses, and savings.

The system is designed to improve financial management efficiency and promote better financial literacy. By enabling users to track and analyze their finances effectively, the Personal Finance Planner System empowers individuals to make informed decisions and improve their overall financial well-being. Managing key processes, such as expense recording and savings calculation, is critical for successful personal financial planning, and this system addresses these challenges seamlessly.

**1.2 Problem Statement**

“Managing personal finances is often challenging due to the lack of proper tools for tracking income, expenses, and savings. Manual methods are time-consuming, prone to errors, and fail to provide meaningful insights into spending habits. This makes it difficult for individuals to analyze their financial patterns and plan effectively. A digital solution is needed to streamline financial management and promote better decision-making”.

**1.3 Objective**

The objective of the Personal Finance Planner mini-project is to develop a comprehensive and user-friendly application that helps individuals effectively manage their financial data, including income, expenses, savings, and budgets.

The project aims to streamline financial tracking by providing users with a centralized platform to record and categorize their financial transactions, offering clear insights into their financial status. By automating calculations for total income, expenses, and savings, the system reduces errors and enhances time efficiency, allowing users to focus on making informed financial decisions.

Additionally, the application facilitates budget planning and monitoring, enabling users to set financial goals and track their adherence to planned budgets. A secure login system ensures data privacy, while the intuitive interface ensures accessibility for users of all technical backgrounds.

This project not only addresses the challenges of manual financial management but also lays the groundwork for potential scalability with advanced features like financial analytics and predictive recommendations in the future.By achieving these objectives, the Personal Finance Planner project aims to provide individuals with a reliable and efficient tool to manage their finances, empowering them to achieve financial stability and long-term goals.

**1.4 Modules**

Income Module: Tracks user earnings and financial inflows.

Expense Module: Records and categorizes expenses for better financial control.

Savings Module: Monitors savings goals and progress.

Budgeting Module: Helps in creating and managing budgets.

Reporting Module: Generates financial reports for insights and decision-making.

**2. SURVEY OF TECHNOLOGIES**

The development of the Personal Finance Planner (PFP) system leverages a combination of modern, reliable, and widely adopted technologies to ensure a high-performing, scalable, and efficient application. This section highlights the key technologies utilized in the system, focusing on their role in the development process and how they contribute to achieving the project's goals.

The core technologies employed in the PFP system include:

1. **Java**: As the primary programming language, Java is used for building the application's backend. Known for its platform independence, security features, and object-oriented principles, Java ensures that the system is both scalable and maintainable.
2. **MySQL**: MySQL serves as the relational database management system (RDBMS) for storing user and transaction data. With its efficiency in handling large volumes of data, robust query processing, and high availability, MySQL is a suitable choice for managing the financial records in the system.
3. **JDBC (Java Database Connectivity)**: JDBC is used for establishing a seamless connection between the Java application and the MySQL database. It ensures reliable data retrieval, updates, and transactions between the application logic and database.
4. **Visual Studio Code (VS Code)**: VS Code is the integrated development environment (IDE) chosen for writing and debugging the backend code. With its support for Java extensions and SQL integration, VS Code offers a smooth development experience, enabling efficient coding and testing of the PFP system.
5. **HTML/CSS (Frontend)**: For the user interface, HTML and CSS are used to create an intuitive and responsive layout, ensuring that users can interact with the system effortlessly.

**2.1. Development Environment**

For the development of the Personal Finance Planner (PFP) system, Visual Studio Code (VS Code) is used as the primary Integrated Development Environment (IDE). VS Code is a versatile and lightweight IDE that provides an efficient platform for building, debugging, and managing the backend and frontend code. Its simplicity and extensive range of features make it an ideal choice for developers working on various technologies.

Key features of VS Code that enhance the development experience include:

* **Syntax Highlighting and IntelliSense**: These features significantly reduce coding errors and improve development speed by offering real-time code suggestions and auto-completions.
* **Extensions**: VS Code supports various extensions for Java, MySQL, and Git, which allow seamless integration with these technologies. This enables developers to work across multiple languages and tools within a single environment.
* **Built-in Terminal**: The inclusion of an integrated terminal in VS Code allows developers to run Java programs and execute database queries directly from the IDE, improving workflow efficiency.
* **Version Control Integration**: VS Code's Git integration simplifies version control tasks such as committing, pushing, and pulling changes, ensuring smooth collaboration and management of the project’s source code.

**2.2. Languages**

The Personal Finance Planner (PFP) system utilizes a combination of SQL and Java to manage the backend and application logic, respectively. These languages are chosen for their robustness, efficiency, and versatility in handling complex tasks.

**2.2.1 SQL**

Structured Query Language (SQL) is integral to the PFP system for managing the database and handling all data-related operations. SQL enables efficient querying, manipulation, and maintenance of the database, ensuring the proper storage and retrieval of financial data.

Key applications of SQL in the PFP system include:

* **Database Schema Design**: SQL is used to create tables and define the relationships between them. For instance, tables such as Income, Expenses, and Users store vital information like income amounts, expense categories, and user details.
* **Data Manipulation**: SQL supports CRUD operations (Create, Read, Update, Delete) to ensure the system remains updated with user inputs. For example:
  + Inserting new expense records using an INSERT INTO Expenses query.
  + Retrieving users' monthly expenses via a SELECT query that filters data based on specific criteria.
* **Constraints**: SQL constraints, such as NOT NULL and FOREIGN KEY, are employed to ensure data integrity. For example, an expense record cannot be inserted unless the user ID exists in the database.
* **Query Optimization**: SQL is optimized using techniques like indexing, which improves the performance of queries, ensuring the system can handle large datasets efficiently, even with complex calculations and reports.

Overall, SQL ensures that the financial data is handled securely, efficiently, and accurately throughout the operation of the PFP system.

4o mini

**2.2.2 Java**

Java is the primary language used for the application logic of the Personal Finance Planner (PFP) system. It facilitates the connection between the user interface and the database while leveraging its object-oriented features to enhance modularity, maintainability, and scalability of the system.

**Applications of Java in PFP:**

* **Object-Oriented Programming (OOP):** Java’s OOP principles allow entities like income, expenses, and users to be modeled as objects, each with attributes and methods. This simplifies the overall structure of the system and makes the code more readable and easier to maintain.
* **Database Interaction:** Java communicates with the MySQL database through JDBC (Java Database Connectivity). This ensures secure execution of SQL queries and proper data retrieval, which is then displayed to users via the interface.
* **User Interface:** JavaFX and Swing are used to create dynamic forms and interactive elements such as expense entry forms and reports. For example:
  + A form for users to enter their monthly income or new expense details.
  + A table that dynamically updates with a list of recorded expenses.
* **Multithreading:** Java’s support for multithreading allows the system to handle multiple users at once without performance degradation, especially useful for systems with high user activity.
* **Error Handling:** Java's robust exception handling framework ensures that the system responds gracefully to runtime issues, such as invalid user inputs or database connection failures.

**Advantages of Java for PFP:**

* **Platform Independence:** Java’s "write once, run anywhere" philosophy makes it an ideal choice for the PFP system, ensuring compatibility across multiple platforms and operating systems.
* **Security:** Java's built-in security features, such as encryption and authentication, protect user data and prevent unauthorized access.
* **Rich Libraries:** Java offers a wide array of libraries, including logging libraries like Log4j and report generation libraries like Apache POI, which enhance the functionality of the PFP system.

**3. REQUIREMENTS AND ANALYSIS**

In this section, we explore the detailed requirements and analysis of the Personal Finance Planner system. The section covers both functional and non-functional requirements, addressing how the system will cater to user needs like tracking income, managing expenses, calculating savings, and providing budgeting tools. We will also outline the system’s architecture, focusing on the design process, which includes key diagrams such as the system architecture diagram and Entity-Relationship (ER) diagram. Additionally, we will discuss the database structure and normalization techniques to ensure a secure, efficient, and scalable system that can handle growing financial data and provide a seamless user experience.

**3.1 Requirement Specification**

The requirement specification section defines the functional and non-functional requirements for the Personal Finance Planner (PFP) system. Some features of the system, such as income tracking, expense management, savings calculation, and budgeting tools. It details how the system serves both individual users who need to manage their finances and administrators who may oversee multiple accounts. The functional requirements ensure that users can input financial data, view detailed summaries, and generate reports, while non-functional requirements focus on system performance, security, and usability to ensure a seamless and efficient experience.

**3.1.1 Functional Requirements**

The functional requirements for the Personal Finance Planner (PFP) system outline the core features and functionalities needed for users to effectively manage their finances. The system must allow users to input and track their income sources, such as salary, freelance earnings, and other revenue streams. Users should be able to categorize income by source and date, and the system should calculate and display the total income based on the entered data.

For expense management, the system should allow users to add, modify, and delete their expenses. These expenses must be categorized by type, such as groceries, utilities, or entertainment.The system should track all transactions, displaying a running total of expenses, and store relevant details such as the expense name, amount, date, and category.

In terms of savings tracking, the system should automatically calculate the difference between the total income and expenses, providing the user with an updated savings balance. Users should also have the option to set savings goals and monitor their progress over time, adjusting figures as their financial situation changes.

Budget management is another critical feature, allowing users to define a monthly or annual budget to keep their spending in check. The system should provide alerts when users exceed their allocated budget in specific categories, and it should offer flexibility to update budget limits and categories as needed.

**3.1.2 Non-Functional Requirements**

The Personal Finance Planner (PFP) system must meet several non-functional requirements to ensure its effectiveness and quality.

**Performance** is crucial, with the system needing to handle multiple user inputs without delays and display real-time financial updates.

**Security** is essential to protect sensitive financial data, with encryption, strong authentication, and two-factor authentication (2FA).

**Usability** requires an intuitive interface with clear visualizations for easy data entry and understanding.Growing user data and **reliable**, with backup mechanisms and regular updates to maintain performance and data integrity.

**3.2 Hardware and Software Requirements**

The Personal Finance Planner system requires a computer or server with at least 4 GB of RAM and 500 GB of storage. For software, it needs a web server (Apache/Nginx), a database system (MySQL/PostgreSQL), and a backend language (PHP/Java). Additionally, modern web browsers and development tools like VS Code are required for smooth operation and development.

**Hardware Requirements**

The Personal Finance Planner (PFP) system requires specific hardware configurations for both server and client-side components. On the server side, the processor should be at least an Intel Core i5 or higher to handle data processing and querying efficiently. A minimum of 8 GB of RAM is necessary to manage multiple database transactions and server processes simultaneously. Storage should include a 500 GB SSD to ensure fast data storage and retrieval, enabling the system to handle extensive financial records with minimal latency. Stable network connectivity, through Ethernet or Wi-Fi, is essential for seamless local and remote access.

For the client-side, a minimum of an Intel Core i3 processor is recommended for machines accessing the system. A minimum of 4 GB of RAM will allow the client-side application to operate smoothly on most devices. The system should be compatible with screens having a resolution of at least 1366x768 pixels for optimal interface visibility. Additionally, the operating system should be Windows 10 or later, or any Linux-based OS, to ensure compatibility with development tools and libraries.

**Software Requirements**

The Personal Finance Planner (PFP) system relies on specific software components for efficient operation. MySQL 8.0 or later is required as the database management system to handle and manage financial data effectively. The connection between the Java application and the MySQL database is established using JDBC (Java Database Connectivity), which ensures seamless communication between the backend and the database.

For the application’s backend and logic implementation, Java (version 3.9 or higher) is the chosen programming language. Java provides the necessary libraries and tools for creating a robust and scalable backend. Visual Studio Code serves as the integrated development environment (IDE) for this project, offering extensions for both Java and SQL to streamline the development process and facilitate coding, debugging, and managing the database queries efficiently.

**3.3 Architecture Diagram**

The architecture diagram for the Personal Finance Planner (PFP) system illustrates the separation of the system into three primary layers: the presentation layer, the application logic layer, and the database layer. This three-tier architecture is designed to enhance modularity, maintainability, and scalability.

1. **Presentation Layer**: This is the user interface layer where users interact with the system. It includes the web or application interface where users can input their income, add expenses, view their savings, and track their financial data.
2. **Application Logic Layer**: This layer processes all the business logic of the system. It handles the core functionality of the PFP system, including calculations of total income, expenses, savings, and handling user inputs. The logic layer ensures that the system performs the necessary computations based on user actions.
3. **Database Layer**: The database layer is responsible for managing the storage and retrieval of financial data. It communicates with the application logic to store income, expenses, and other financial records. The database layer is built on MySQL, where all financial information is securely stored and can be queried by the application logic.

**3.4 Entity-Relationship (ER) Diagram**

The Entity-Relationship (ER) diagram models the relationships between key entities in the Personal Finance planner System. It is a critical tool for designing the database schema and understanding how different entities are connected.

[ User ]

|---> [ Login ] (1-to-1)

|

|---> [ Income ] (1-to-Many)

|

|---> [ Expense ] (1-to-Many)

| |

| ---> [ Category ] (1-to-Many)

|

|---> [ Savings ] (1-to-Many)

|

|---> [ Budget ] (1-to-Many)

The ER diagram for the Personal Finance Planner is as shown above in the Figure.

The Personal Finance Planner database system mainly consists of 6 entities, namely User, Login, Income, Expense, Savings, and Budget.

• User and Login have a 1:1 cardinality

• User and Income have a 1:N cardinality ratio

• User and Expense have a 1:N cardinality ratio

• User and Savings have a 1:N cardinality ratio

• User and Budget have a 1:N cardinality ratio

• Expense and Category have a 1:N cardinality ratio

This ER diagram showcases how the data is structured and interconnected, ensuring smooth management of the personal finance planning process.

**4 PROGRAM CODE**

**4.1 BACKEND DESIGN**

The backend design of the Personal Finance Planner (PFP) system is responsible for managing the application's core functionality, such as processing user input, interacting with the database, and performing calculations. The backend ensures that the financial data is securely stored and retrieved, and it handles the logic for budgeting, expense tracking, and income management

PROGRAM

**pom.xml :**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.yourcompany</groupId>

<artifactId>yourproject</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- MySQL JDBC Driver -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.34</version>

</dependency>

<!-- Other dependencies -->

</dependencies>

</project>

Backend for database connection

**Finance planner :**

package backend;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class FinancePlanner {

private double totalIncome;

private List<Expense> expenses;

private Connection connection;

// Constructor

public FinancePlanner(double totalIncome) {

this.totalIncome = totalIncome;

this.expenses = new ArrayList<>();

// Establish database connection

try {

// Make sure to replace with your actual DB credentials

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/finance\_db", "root", "070604@MAk");

} catch (SQLException e) {

e.printStackTrace();

System.exit(1); // Exit if connection fails

}

}

public void addExpense(String name, double amount) {

expenses.add(new Expense(name, amount));

// Store expense in the database

try (PreparedStatement stmt = connection.prepareStatement("INSERT INTO expenses (name, amount) VALUES (?, ?)")) {

stmt.setString(1, name);

stmt.setDouble(2, amount);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public double calculateTotalExpenses() {

double total = 0;

for (Expense expense : expenses) {

total += expense.getAmount();

}

return total;

}

public double calculateSavings() {

return totalIncome - calculateTotalExpenses();

}

public void displaySummary() {

System.out.println("Income: " + totalIncome);

System.out.println("Total Expenses: " + calculateTotalExpenses());

System.out.println("Total Savings: " + calculateSavings());

System.out.println("Expenses Breakdown:");

for (Expense expense : expenses) {

System.out.println("- " + expense.getName() + ": " + expense.getAmount());

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your total income: ");

double income = scanner.nextDouble();

scanner.nextLine(); // consume the leftover newline

FinancePlanner planner = new FinancePlanner(income);

while (true) {

System.out.print("Enter expense name (or 'done' to finish): ");

String name = scanner.nextLine();

if (name.equalsIgnoreCase("done")) {

break;

}

System.out.print("Enter expense amount: ");

double amount = scanner.nextDouble();

scanner.nextLine(); // consume the leftover newline

planner.addExpense(name, amount);

}

planner.displaySummary();

scanner.close();

}

}

**Expense class :**

package backend;

public class Expense {

private String name;

private double amount;

public Expense(String var1, double var2) {

this.name = var1;

this.amount = var2;

}

public String getName() {

return this.name;

}

public double getAmount() {

return this.amount;

}

}

**Overview:**

FinancePlanner → Holds multiple Expense objects.

addExpense() → Adds an expense to expenses list and stores in DB.

calculateTotalExpenses() → Calculates total of all expenses.

calculateSavings() → Subtracts total expenses from totalIncome.

displaySummary() → Displays income, expenses, savings, and expense breakdown.

**Backend Implementation**

In the backend of the Personal Finance Planner (PFP) project, the core objective was to effectively manage financial data, including income and expenses, while maintaining seamless integration with a MySQL database. The backend is designed with a FinancePlanner class that manages the user's total income and tracks expenses, with a relational database connection used to store the expenses. Each expense is represented by the Expense class, encapsulating the expense name and amount.

By leveraging JDBC, I established a connection to the MySQL database where expenses are inserted, enabling persistent storage. The backend uses PreparedStatements to ensure safe insertion of data into the database and calculates the total expenses and savings dynamically.

**4.2 FRONTEND DESIGN**

The frontend design of the Personal Finance Planner (PFP) system plays a crucial role in ensuring a user-friendly interface and seamless interaction with the backend. It focuses on presenting the financial data effectively, allowing users to easily input, modify, and view their financial information.

**HTML:**

* The structure of the page includes sections for income input, expense input, a summary of the calculations, and a button to calculate savings.
* It uses a <ul> element to dynamically display the list of expenses as the user adds them.

**JavaScript:**

* The JavaScript code defines three main variables:
  + totalIncome stores the total income entered by the user.
  + expenses is an array to keep track of each expense (name and amount).
  + totalExpense keeps track of the total expense amount.
* The addExpense() function adds an expense to the list of expenses and updates the total expenses.
* The calculateSavings() function calculates the savings by subtracting totalExpense from totalIncome and updates the summary with the calculated values.

**CSS:**

* The design is clean and minimalistic with some basic styling:
  + The .container is centered and styled with a white background and shadow for better presentation.
  + Input fields and buttons are styled for clarity and ease of use.
  + The expense list is styled to have a soft background color, rounded corners, and spacing between items.

**PROGRAM**

**Index.html :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Finance Planner</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="container">

<h1>Finance Planner</h1>

<div class="input-section">

<label for="income">Enter your total income:</label>

<input type="number" id="income" placeholder="Income" />

</div>

<div class="expense-section">

<h3>Expenses</h3>

<div class="expense-input">

<input type="text" id="expenseName" placeholder="Expense name" />

<input type="number" id="expenseAmount" placeholder="Amount" />

<button onclick="addExpense()">Add Expense</button>

</div>

<ul id="expenseList"></ul>

</div>

<div class="summary">

<h3>Summary</h3>

<p id="totalIncome">Income: $0</p>

<p id="totalExpenses">Total Expenses: $0</p>

<p id="totalSavings">Total Savings: $0</p>

</div>

<div class="submit">

<button onclick="calculateSavings()">Calculate Savings</button>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**script.js :**

let totalIncome = 0;

let expenses = [];

let totalExpense = 0;

function addExpense() {

const expenseName = document.getElementById('expenseName').value;

const expenseAmount = parseFloat(document.getElementById('expenseAmount').value);

if (expenseName && !isNaN(expenseAmount) && expenseAmount > 0) {

expenses.push({ name: expenseName, amount: expenseAmount });

totalExpense += expenseAmount;

const expenseList = document.getElementById('expenseList');

const expenseItem = document.createElement('li');

expenseItem.textContent = ${expenseName}: $${expenseAmount.toFixed(2)};

expenseList.appendChild(expenseItem);

// Clear input fields

document.getElementById('expenseName').value = '';

document.getElementById('expenseAmount').value = '';

} else {

alert('Please provide valid expense details.');

}

}

function calculateSavings() {

totalIncome = parseFloat(document.getElementById('income').value);

if (isNaN(totalIncome) || totalIncome <= 0) {

alert('Please enter a valid income.');

return;

}

const totalSavings = totalIncome - totalExpense;

// Display results

document.getElementById('totalIncome').textContent = Income: $${totalIncome.toFixed(2)};

document.getElementById('totalExpenses').textContent = Total Expenses: $${totalExpense.toFixed(2)};

document.getElementById('totalSavings').textContent = Total Savings: $${totalSavings.toFixed(2)};

}

**style.css :**

body {

font-family: Arial, sans-serif;

background-color: #f5f5f5;

margin: 0;

padding: 0;

}

.container {

max-width: 600px;

margin: 50px auto;

padding: 20px;

background-color: #fff;

border-radius: 10px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

h1 {

text-align: center;

color: #4CAF50;

}

.input-section,

.expense-section,

.summary {

margin-bottom: 20px;

}

input[type="number"],

input[type="text"] {

padding: 10px;

margin: 10px 0;

width: calc(50% - 20px);

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box;

}

button {

background-color: #4CAF50;

color: white;

padding: 10px 15px;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

.expense-input {

display: flex;

justify-content: space-between;

}

.expense-input input {

width: calc(40% - 20px);

}

ul {

padding-left: 0;

list-style: none;

}

li {

background-color: #f2f2f2;

padding: 8px;

margin: 5px 0;

border-radius: 5px;

}

**Overview**

FinancePlanner (HTML Structure) → Contains sections for income input, expense input, and summary display.

addExpense() (JavaScript Functionality) → Takes expense name and amount, adds it to the expense list, and dynamically displays it in the expense section.

calculateTotalExpenses() (JavaScript Calculation) → Sums up the total amount of all expenses entered by the user.

calculateSavings() (JavaScript Calculation) → Subtracts total expenses from the total income to calculate savings.

displaySummary() (HTML & JavaScript Interaction) → Updates the page with the user's total income, total expenses, total savings, and a breakdown of expenses

**Frontend Implementation**

The frontend of the Personal Finance Planner (PFP) project is built using HTML, CSS, and JavaScript. It includes an input section for income and expenses, where users can enter the details. Expenses are displayed in a dynamic list, and the summary section shows total income, total expenses, and savings. JavaScript handles input validation, updates the UI, and calculates the savings by subtracting expenses from income, displaying the results on the page.

**5.RESULT AND DISCUSSION**

The Personal Finance Planner (PFP) project successfully achieved its goal of helping users manage their finances by allowing them to input, track, and calculate income, expenses, and savings. Upon entering income and various expenses, the application dynamically updates the expense list and calculates the total expenses. The savings are then calculated by subtracting the total expenses from the income, and the results are displayed in the summary section.

**Results:**

1. **Income and Expense Input:** Users can input their total income and expenses easily using simple forms.
2. **Dynamic Expense Tracking:** Each expense is added to a list, and the total expenses are updated in real time.
3. **Savings Calculation:** After entering income and expenses, users can click a button to calculate and display their total savings.

**Discussion:**

* **User Experience:** The interface is user-friendly, with clear input fields for income and expense details, along with an intuitive button for calculations.
* **Scalability:** The system is designed for personal use; however, it can be expanded to accommodate more complex financial planning features such as categories for expenses, savings goals, and reports.
* **Technologies:** The combination of HTML, CSS, and JavaScript offers a lightweight and efficient solution for the frontend. The backend, although not implemented in this case, can be seamlessly integrated for advanced functionality like persistent storage and multi-user support using a database like MySQL.

The project successfully demonstrates how a simple finance planner can be implemented, providing users with an easy-to-use tool to monitor their finances.

**6.CONCLUSIONS**

The Personal Finance Planner (PFP) project successfully fulfills its purpose of helping users track and manage their finances by providing a straightforward interface for entering income and expenses, calculating total savings, and displaying detailed financial summaries. The project achieved the following key objectives:

1. **Financial Tracking**: The system allows users to easily input their income and various expenses, and automatically calculates the total expenses and savings.
2. **User-Friendly Interface**: The clean and simple user interface ensures that users can interact with the system with minimal effort, making it accessible to a wide range of users.
3. **Dynamic Updates**: Real-time updates for expenses and savings help users keep track of their finances efficiently.

In summary, the Personal Finance Planner serves as an effective tool for personal finance management, offering core functionalities with the potential for future enhancements such as database storage, multi-user support, and more detailed financial analysis features.

**7.REFERENCES**

1. "Java Database Connectivity (JDBC) Overview," Oracle. <https://www.oracle.com/java/technologies/jdbc-ucp.html>
2. "SQL Syntax," W3Schools. https://www.w3schools.com/sql/
3. "JavaFX," Oracle. <https://openjfx.io/>
4. "Introduction to CSS," Mozilla Developer Network. <https://developer.mozilla.org/en-US/docs/Web/CSS>
5. "Understanding HTML," W3Schools. https://www.w3schools.com/html/
6. "Java Programming Language," Oracle. <https://www.oracle.com/java/>
7. "Visual Studio Code Documentation," Microsoft. <https://code.visualstudio.com/docs>
8. "The Importance of Personal Finance Management," Investopedia. https://www.investopedia.com/articles/personal-finance/122015/importance-personal-finance-management.asp