



21PE18/21SE18/21FE18/21DE18/21YE18

WEB APPLICATION DEVELOPMENT USING JAVA

PERSONAL FINANCE MANAGEMENT TOOL

SEMSTER V

2024

J.JAYARAJ

717822P221

CSE-B

Abstract:

- The **Personal Finance Management Tool** is a comprehensive web-based application designed to help users track income, expenses, and manage their finances efficiently. The system comprises two primary roles: **Admin** and **User**. Admins manage user accounts and monitor transactions, while users can input financial data, generate reports, and visualize their financial standing. The backend is developed using **Java (Servlet + JDBC)** to ensure secure transaction management, while the database leverages **MySQL** for storing user data, transactions, and generating financial reports. This system is designed to be scalable, user-friendly, and secure, making it suitable for personal or small business use.

Table of Contents:

S.NO	TITLE	PAGE_NO
1	Introduction	2-3
2	System Analysis	3-3
3	System Design	3-4
4	Implementation	4-4
5	References	4-4
6	Source Code & Output	5-10

1. Introduction:

- **Purpose:**

The Personal Finance Management Tool is designed to help users track and manage their financial transactions, including income and expenses, providing clear visibility into their financial health and promoting better financial management practices.

- **Target Audience:**

This tool is ideal for individuals and small business owners who need a simple yet effective system to monitor personal finances, manage budgets, and generate reports without relying on complex financial software.

- **Technology Stack:**

The backend system is built using **Java (Servlet + JDBC)** for secure transaction management, with **MySQL** as the database for storing and retrieving user information and financial data. The frontend uses **HTML, CSS, and JavaScript** for user interaction.

- **System Roles:**

The system supports two roles: **Admins**, who manage user accounts and monitor all transactions, and **Users**, who track their own financial data (income and expenses) and generate reports to understand their financial status.

- **Security and Scalability:**

To ensure data security, user credentials are encrypted, and role-based access control is implemented. The system is designed to handle multiple users and can scale with increasing data without performance degradation.

2. System Analysis:

Problem Definition:

Many individuals struggle to maintain a comprehensive record of their income and expenses, leading to difficulty in tracking financial health. A system that allows users to input, manage, and track their finances in one place can help overcome these challenges by providing visibility and accountability.

Objectives:

- Allow users to securely log in and track their financial transactions.
- Enable admins to manage user accounts and oversee system-wide financial activities.
- Ensure transaction data is securely stored and easily retrievable for generating reports.
- Implement error handling and validation to avoid incorrect data inputs.

User Roles:

1.Admin:

- Can create, update, and delete user accounts.
- Can monitor users' transactions and generate system-wide financial reports.

2.User:

- Can log in, track income and expenses, and manage personal finance.
- Can generate reports to evaluate financial health over specific periods.

System Requirements:

- **Functional Requirements:**

- User Registration and Login: Secure login using usernames and passwords.
- Transaction Management: Add, edit, and delete transactions (income/expense).
- Report Generation: Summarize total income, expenses, and savings.

- **Non-Functional Requirements:**

- **Security:** User passwords must be hashed, sensitive data encrypted, and role-based access implemented.
- **Scalability:** The system should support multiple users and large data sets.
- **Performance:** Quick response time for data retrieval and transaction management.

3. System Design:

Architecture:

The application follows a **3-tier architecture**:

1. **Presentation Layer (Frontend):**

- Technologies: HTML, CSS, JavaScript
 - Provides user interaction through web forms and displays results (user dashboards, admin dashboards).
2. **Application Layer (Backend):**
 - Technologies: Java (Servlet + JDBC)
 - Contains the core business logic for handling transactions, user management, and generating reports.
 3. **Data Layer (Database):**
 - Technologies: MySQL
 - Stores user details, financial transactions, and generates queries for reporting.

Database Design:

- **Users Table:**
 - Columns: id, username, password, email
- **Transactions Table:**
 - Columns: id, user_id, type, amount, description

Key Components:

1. **User Authentication:**
 - Implemented via Java Servlets handling login and registration, with validation and password encryption.
2. **Transaction Management:**
 - Add/Edit/Delete functionality for transactions, managed via Servlets interacting with the MySQL database through JDBC.
3. **Admin Management:**
 - Admin can view and manage user accounts, check system-wide financial trends, and generate reports.
4. **Reporting:**
 - SQL queries retrieve summarized transaction data for report generation (e.g., monthly expenses).

4. Implementation:

- **Login:**

The system checks the username and password from the database.
- **Add Transaction:**

Once logged in, the user can add transactions specifying the type (income or expense), amount, and description.
- **Generate Report:**

The user can view a simple report showing total income and expenses.

References:

- Sierra, K., & Bates, B. (2005). Head First Java. O'Reilly Media.
- MySQL Documentation. Retrieved from <https://dev.mysql.com/doc/>

SourceCode:

Database SQL:

```
CREATE DATABASE finance_management;
USE finance_management;
CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL
);
CREATE TABLE Transactions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    type ENUM('income', 'expense') NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    description VARCHAR(255),
    FOREIGN KEY (user_id) REFERENCES Users(id)
);
```

Login servlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        try {
            // JDBC connection
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/finance_management", "root",
"password");
            PreparedStatement ps = con.prepareStatement("SELECT * FROM Users WHERE username=? AND password=?");
            ps.setString(1, username);
            ps.setString(2, password);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                HttpSession session = request.getSession();
                session.setAttribute("username", username);
                response.sendRedirect("user_dashboard.html");
            } else {
                // Send error message
                response.sendRedirect("login.html?error=Invalid username or password");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Transaction.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class TransactionServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```

HttpSession session = request.getSession();
String username = (String) session.getAttribute("username");
String transactionType = request.getParameter("transactionType");
double amount = Double.parseDouble(request.getParameter("amount"));
String description = request.getParameter("description");
try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/finance_management", "root",
"8826");
    PreparedStatement ps = con.prepareStatement("SELECT id FROM Users WHERE username=?");
    ps.setString(1, username);
    ResultSet rs = ps.executeQuery();
    int userId = 0;
    if (rs.next()) {
        userId = rs.getInt("id");
    }
    ps = con.prepareStatement("INSERT INTO Transactions (user_id, type, amount, description) VALUES (?, ?, ?, ?)");
    ps.setInt(1, userId);
    ps.setString(2, transactionType);
    ps.setDouble(3, amount);
    ps.setString(4, description);
    ps.executeUpdate();
    response.sendRedirect("user_dashboard.html?success=Transaction added successfully");
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

AdminServlet.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class AdminServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String action = request.getParameter("action");
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/finance_management", "root",
"8826");
            if ("addUser".equals(action)) {
                String username = request.getParameter("username");
                String email = request.getParameter("email");
                String password = "defaultPassword";
                PreparedStatement ps = con.prepareStatement("INSERT INTO Users (username, email, password) VALUES (?, ?,
?");
                ps.setString(1, username);
                ps.setString(2, email);
                ps.setString(3, password);
                ps.executeUpdate();
                response.sendRedirect("admin_dashboard.html?success=User added successfully");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}
}
DBconnection.java
import java.sql.*;

public class DBConnection {
    private static Connection connection = null;
    public static Connection getConnection() throws SQLException, ClassNotFoundException {
        if (connection == null || connection.isClosed()) {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/finance_management", "root", "password");
        }
        return connection;
    }
}

```

Web.xml

```

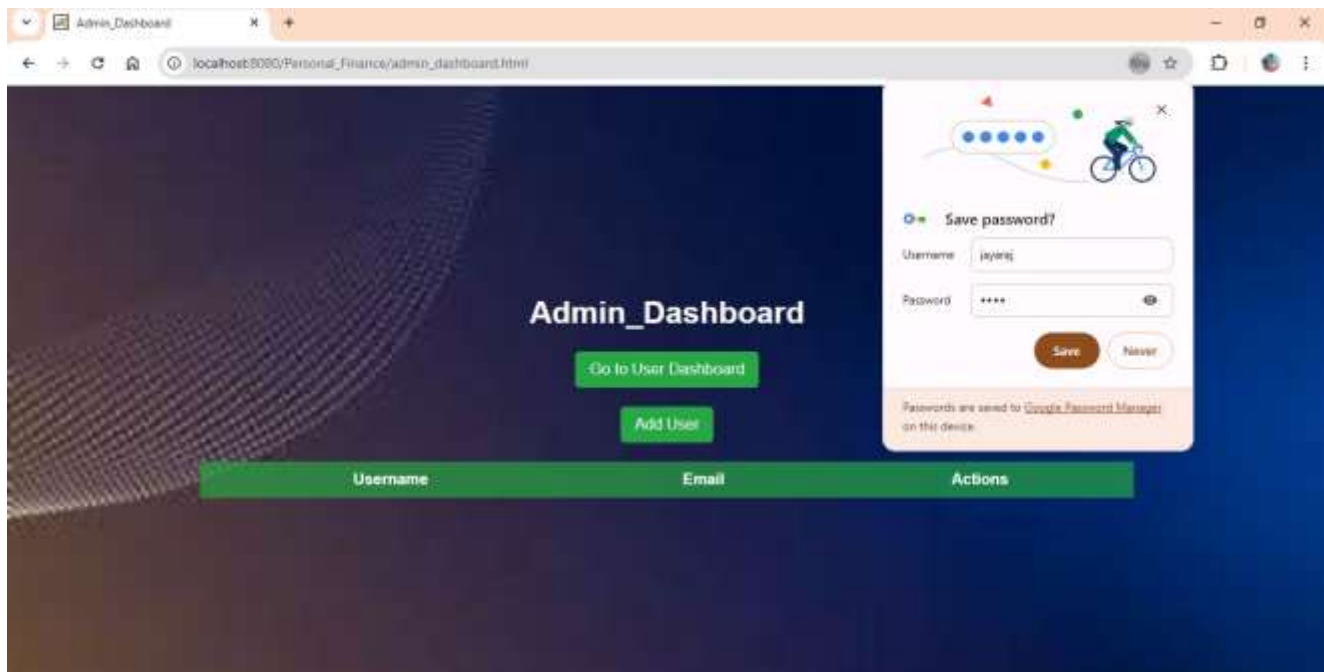
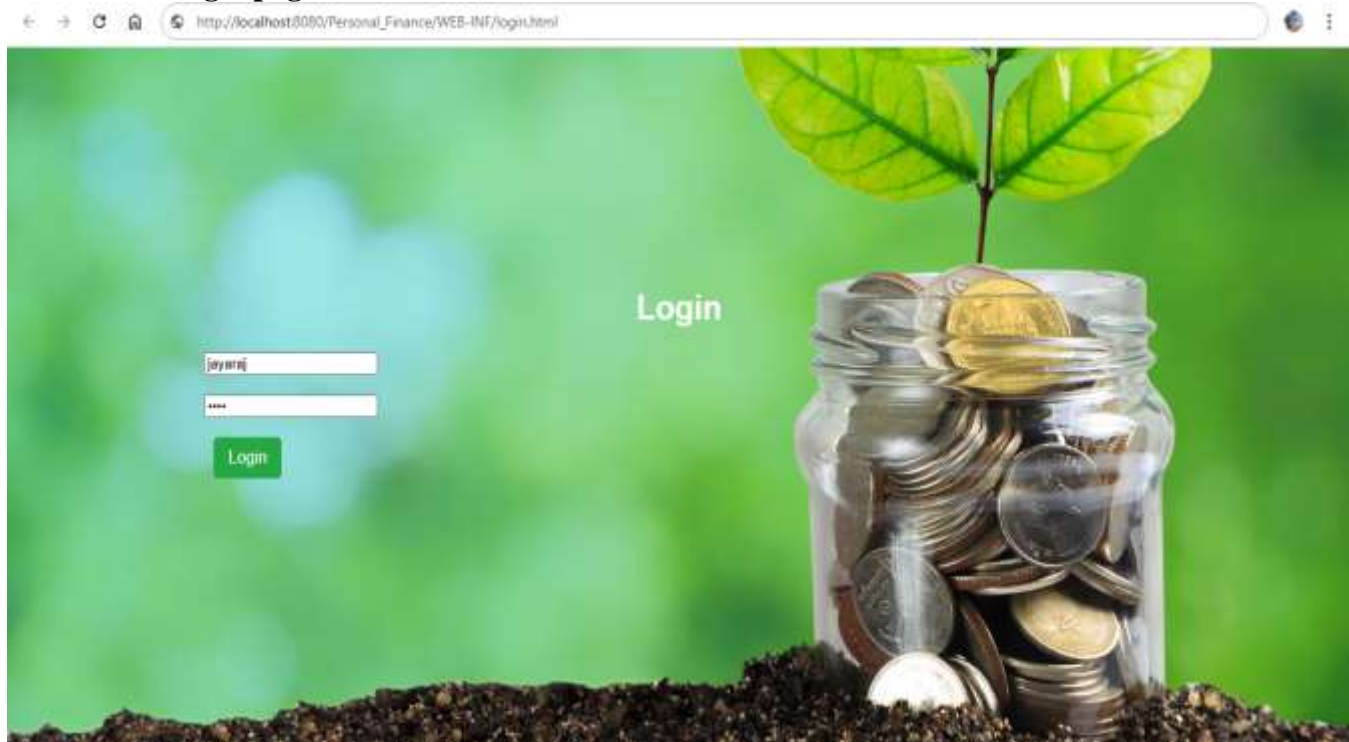
<web-app>
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>TransactionServlet</servlet-name>
        <servlet-class>TransactionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>TransactionServlet</servlet-name>
        <url-pattern>/transaction</url-pattern>    </servlet-mapping>
    <servlet>
        <servlet-name>AdminServlet</servlet-name>
        <servlet-class>AdminServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>AdminServlet</servlet-name>
        <url-pattern>/admin</url-pattern>
    </servlet-mapping>
</web-app>

```

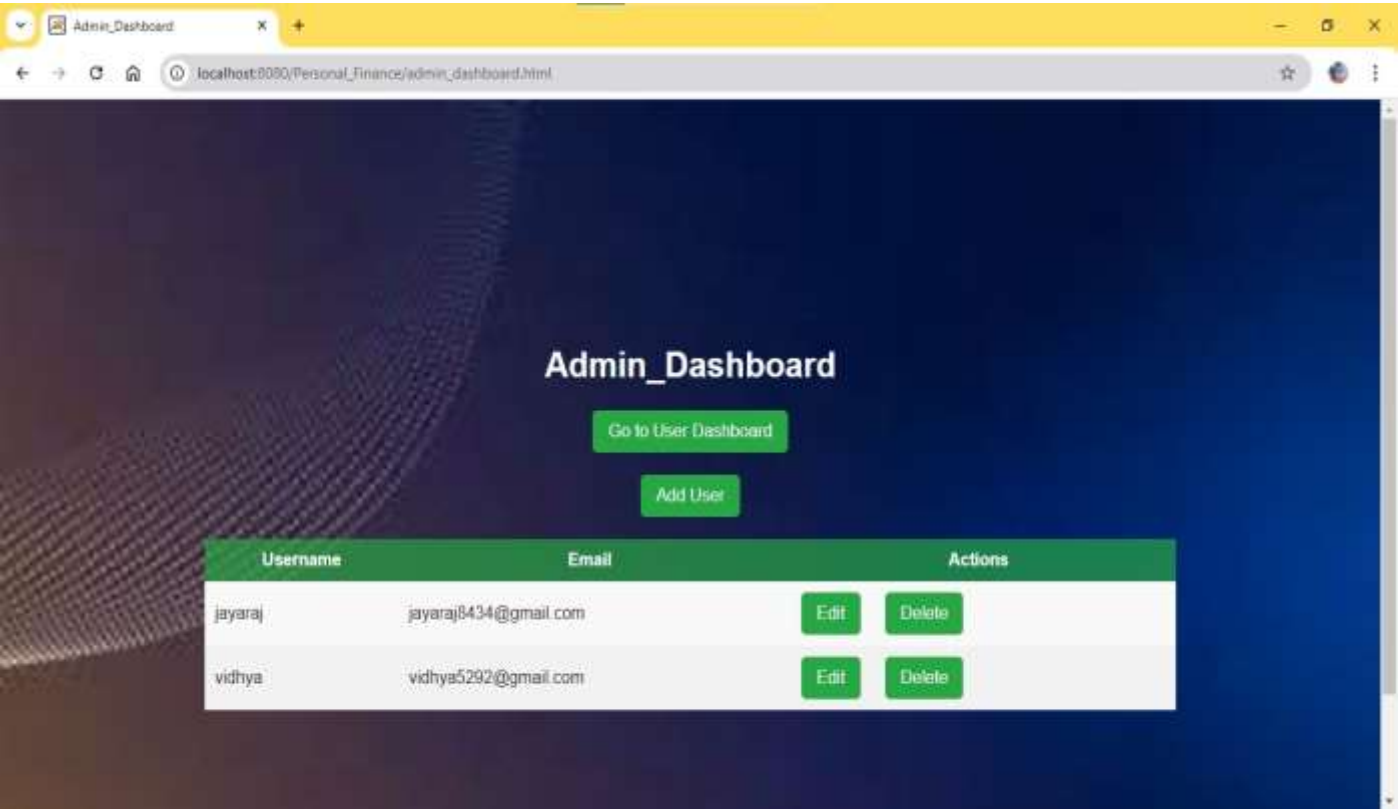
OUTPUT:

WEB SERVLET PAGE:

Login page:




AdminServletPage:




SQL:

AdminServletPage:


Result Grid







Filter Rows:

Edit:



	id	username	password	email
	1	jayaraj	8434	jayaraj8434@gmail.com
	2	vidhya	5292	vidhya5292@gmail.com
	NULL	NULL	NULL	NULL

WEB SERVLET PAGE:

TransactionServletPage:

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/Personal_Finance/user_dashboard.html'. The page title is 'User Dashboard'. It features a dark-themed interface with a sidebar on the left containing a 'Back to Admin Dashboard' button. The main content area includes an 'Add Transaction' section with a dropdown for 'Transaction Type' (set to 'Income'), an 'Amount' input field (containing '1000.00'), and a 'Description' input field (containing 'Salary Payment'). Below this is a green 'Add Transaction' button. A 'Summary' section displays 'Total Income: \$3000.00', 'Total Expense: \$650.00', and 'Balance: \$2350.00'. At the bottom is a 'Transaction List' table with the following data:

ID	User ID	Type	Amount	Description
3	1	income	1000.00	Salary Payment
4	1	expense	150.00	Groceries
5	2	income	2000.00	Freelance Payment
6	2	expense	500.00	Rent

SQL:

TransactionServletPage:

The screenshot shows a SQL query result grid with a toolbar at the top containing a 'Result Grid' button, a 'Filter Rows' input field, and an 'Edit' button. The table below displays the query results:

	id	user_id	type	amount	description
▶	3	1	income	1000.00	Salary Payment
	4	1	expense	150.00	Groceries
	5	2	income	2000.00	Freelance Payment
	6	2	expense	500.00	Rent
⊙	NULL	NULL	NULL	NULL	NULL