KARPAGAM COLLEGE OF ENGINEERING
Rediscover | Refine | Redefine
(Autonomous)

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

WEB APPLICATION DEVELOPMENT USING JAVA

PERSONAL FINANCE MANAGEMENT TOOL

SEMSTER V

2024

J.JAYARAJ

717822P221
CSE-B

Abstract:

- A Personal Finance Management Tool allows users to manage personal finances and admin to manage user accounts. It features a login system, an admin dashboard for user management, and a user dashboard for tracking income, expenses, and financial summaries.

Table of Contents:

1. Introduction:

- Overview: The Personal Finance Management Tool is designed to help users efficiently track and manage their income and expenses while providing administrative control over user accounts.
- Purpose: It aims to streamline financial management for individuals and offer a centralized platform for monitoring financial health.
- Key Features: The tool includes a secure login system, user dashboards for expense/income tracking, and admin dashboards for user account management.
- Benefits: By simplifying financial tracking and management, users can gain better insights into their financial habits, while admins can oversee user activity and maintain system security.

2. System Analysis:

- **Requirements Analysis:**

  Functional Requirements: These describe the key features the frontend should provide.

  Login Page:

  Create a login interface where users can enter their username and password.
  Basic validation for input fields (e.g., empty fields, password length).
  UI response for login success or failure (this will be handled purely in the frontend for now, simulating login status).

  User Dashboard:

  Expense and Income Tracker:

  Provide input fields for users to add income and expense entries**.**
  Allow users to select categories for each entry (e.g., food, rent, salary).
  List all entered income/expenses dynamically in a table format using JavaScript.

Financial Summary:

Display total income, total expenses, and the balance (income - expenses) in real-time.

Edit/Delete Entries:

Allow users to edit or delete individual income or expense entries from the table.

Admin Dashboard:

Provide a page for user management (simulated with frontend data).

Allow admin to add, edit, or delete "users" (handled via JS to simulate functionality).

Responsiveness:

Ensure all pages are fully responsive and work seamlessly across devices (mobile, tablet, desktop) using CSS media queries.

Non-Functional Requirements:

These describe the qualities of the frontend application, focusing on performance, design, and usability.

Performance:

Ensure quick rendering of data on the frontend, especially when handling multiple expense/income entries.

Minimize the use of heavy JavaScript operations to maintain smooth performance.

Usability:

The interface should be simple, intuitive, and user-friendly.

Navigation between different pages (login, dashboard) should be seamless.

Navigation between different pages (login, dashboard) should be seamless.

Design:

The tool should have a visually appealing and modern design using CSS.

Use consistent color schemes, fonts, and layouts to create a cohesive user experience.

Icons and buttons should be clearly labeled and visually distinguishable .

Accessibility:

Ensure that the application is accessible to users with disabilities by following basic web accessibility guidelines (e.g., using semantic HTML, alt text for images, proper form labeling).

Security (Frontend Perspective):

Implement basic input sanitization to prevent potential JavaScript injection attacks, even though this is only a frontend project.

Ensure sensitive user data (like passwords) is not exposed in any form, even though this will be simulated for now.

3. **User Requirements:**

End Users:

Users should be able to easily add, view, and manage their personal financial data (income/expenses) on the dashboard.

Users expect real-time updates of financial summaries (e.g., total income, total expenses).

The interface should be intuitive, making it easy for users to track and visualize their financial data.

Admin Users :

Admins should have the ability to manage user-related data (simulated in frontend).

- **Specifications of the System:**
  Programming Language: HTML,CSS,JS

4.Implementation:

- **HTML**: Defines the structure of the page, including a form for adding transactions, sections for financial summaries, and a transaction list.
- **CSS**: Provides the styling for the page, including responsive design and simple layouts for financial data and forms.
- **JavaScript**:
  - Handles adding and removing transactions.
  - Calculates the total income, total expenses, and balance.
  - Updates the DOM in real-time to reflect user actions.

References:

W3Schools – HTML,CSS and JavaScript Tutorials:

- A beginner-friendly platform for learning HTML, CSS, and JavaScript with interactive examples.
- Link: https://www.w3schools.com/

SourceCode:

Login.html:

```
<html>
<head>
    <title>Login</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body style="background-image: url('index.jpeg'); background-size: cover; background-
position: center;">
    <form id="loginForm">
        <h1>Login</h1>
        <input type="text" id="username" placeholder="Username" required>
        <input type="password" id="password" placeholder="Password" required>
        <button type="submit">Login</button>
        <div id="error-message"></div>
    </form>
    <script src="login.js"></script>
</body>
</html>
```

Login.js:

```
document.getElementById('loginForm').addEventListener('submit', function(event) {
event.preventDefault();
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;
    const validUsername = "admin";
    const validPassword = "password123";
    if (username === validUsername && password === validPassword) {
        window.location.href = "admin_dashboard.html";
    } else {
        document.getElementById('error-message').innerText = "Invalid username or
password.";
    }
});
```

admin_dashboard.html:

```
<html>
<head>
    <title>Admin Dashboard</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body style="background-image: url('admin_dashboard.jpeg'); background-size: cover;
background-position: center;">
    <h1>Admin Dashboard</h1>
```

```
   <div class="button-container">
        <button><a href="user_dashboard.html" style="text-decoration: none; color:
white;">Go to User Dashboard</a></button>
    </div>
    <div class="button-container">
        <button id="addUser">Add User</button>
    </div>
    <div class="modal-overlay" id="modalOverlay"></div>
    <div id="userModal">
        <h2 id="modalTitle">Add User</h2>
        <input type="text" id="username" placeholder="Username" required>
        <input type="email" id="useremail" placeholder="Email" required>
        <button id="submitUser">Submit</button>
        <button id="closeModal">Cancel</button>
    </div>
    <table id="userTable"> <thead> <tr>
                <th>Username</th>
                <th>Email</th>
                <th>Actions</th> </tr> </thead>
    </table>
  <script src="admin_dashboard.js"></script>
</body>
</html>
```

admin_dashboard.js:

```
let editingUserIndex = null;
document.getElementById('addUser').onclick = function() {
    openModal('Add User');
};
document.getElementById('closeModal').onclick = function() {
    closeModal();
};
document.getElementById('submitUser').onclick = function() {
    const username = document.getElementById('username').value;
    const email = document.getElementById('useremail').value;
    if (username && email) {
        if (editingUserIndex !== null) {
            updateUserInTable(username, email);
        } else {
            addUserToTable(username, email);       }
        clearInputs();
        closeModal();
    } else {
        alert('Please fill in both fields.');
    }
};
function openModal(title, username = '', email = '') {
    document.getElementById('modalTitle').innerText = title;
    document.getElementById('username').value = username;
    document.getElementById('useremail').value = email;
    document.getElementById('userModal').style.display = 'block';
    document.getElementById('modalOverlay').style.display = 'block';
}function closeModal() {
    document.getElementById('userModal').style.display = 'none';
    document.getElementById('modalOverlay').style.display = 'none';
    editingUserIndex = null;
}
function addUserToTable(username, email) {
    const userTableBody = document.getElementById('userTableBody');
    const newRow = userTableBody.insertRow();
    newRow.insertCell(0).innerText = username;
    newRow.insertCell(1).innerText = email;
    const actionCell = newRow.insertCell(2);
    actionCell.innerHTML = `
        <button onclick="editUser(${userTableBody.rows.length - 1})">Edit</button>
        <button onclick="deleteUser(this)">Delete</button>
    `;
}
function editUser(index) {
    const userTableBody = document.getElementById('userTableBody');
    const row = userTableBody.rows[index];

   const username = row.cells[0].innerText;
    const email = row.cells[1].innerText;
    editingUserIndex = index;
    openModal('Edit User', username, email);
}
function updateUserInTable(username, email) {
    const userTableBody = document.getElementById('userTableBody');
    const row = userTableBody.rows[editingUserIndex];
    row.cells[0].innerText = username;
    row.cells[1].innerText = email;
}function deleteUser(button) {
    const userTableBody = document.getElementById('userTableBody');
    const rowIndex = button.parentNode.parentNode.rowIndex;
    userTableBody.deleteRow(rowIndex);
}
```

```
function clearInputs() {
    document.getElementById('username').value = '';
    document.getElementById('useremail').value = '';
}
```

User_dashboard.html:

```
   <html>
<head>
    <title>User Dashboard</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body  style="background-image: url('User_dashboard.jpeg'); background-size: cover;
background-position: center;">
    <h1>User Dashboard</h1>
    <button><a href="admin_dashboard.html" style="text-decoration: none; color:
black;">Back to Admin Dashboard</a></button>
    <form id="transactionForm">
        <select id="transactionType">
            <option value="income">Income</option>
            <option value="expense">Expense</option>
        </select><input type="number" id="amount" placeholder="Amount" required>
        <input type="text" id="description" placeholder="Description">
        <button type="submit">Add Transaction</button>
    </form>
    <div id="summary"></div>
    <h2>Transaction List</h2>
    <table id="transactionList">
        <thead><tr><th>Type</th>
                <th>Amount</th>
                <th>Description</th> </tr> </thead></table>
    <script src="user_dashboard.js"></script>
</body>
</html>
```

user_dashboard.js:

```
const transactionForm = document.getElementById('transactionForm');
const transactionList =
document.getElementById('transactionList').getElementsByTagName('tbody')[0];
let transactions = [];
transactionForm.addEventListener('submit', function(event) {
    event.preventDefault();
    const transactionType = document.getElementById('transactionType').value;
    const amount = parseFloat(document.getElementById('amount').value);
    const description = document.getElementById('description').value;
```

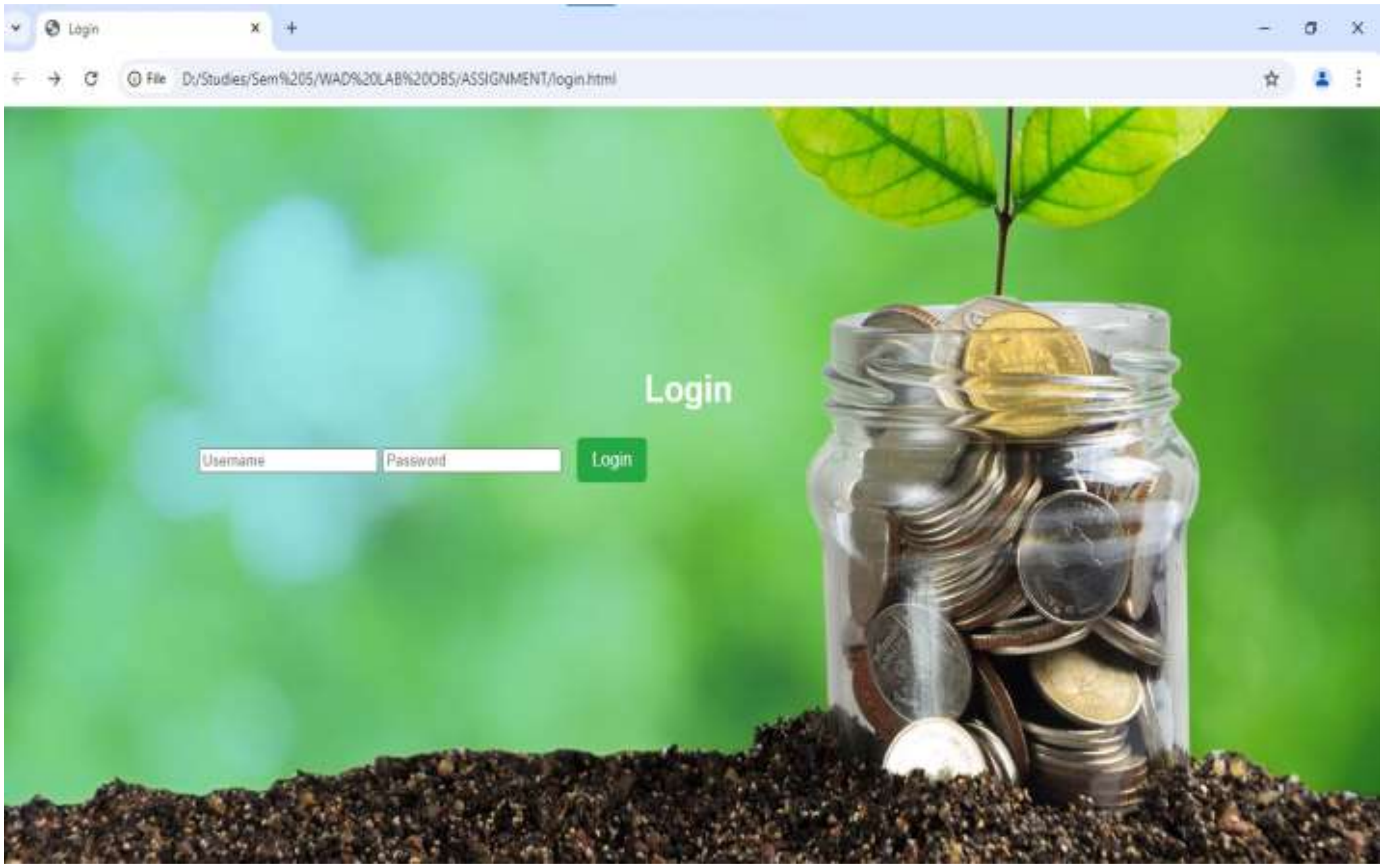
```
   transactions.push({ type: transactionType, amount, description });
    displayTransactions();
    transactionForm.reset();
});
function displayTransactions() {
    transactionList.innerHTML = '';
    transactions.forEach(transaction => {
        const row = transactionList.insertRow();
        row.insertCell(0).innerText = transaction.type.charAt(0).toUpperCase() +
transaction.type.slice(1);
        row.insertCell(1).innerText = transaction.amount.toFixed(2);
        row.insertCell(2).innerText = transaction.description;
    });
    updateSummary();
}

function updateSummary() {
    const incomeTotal = transactions
        .filter(t => t.type === 'income')
        .reduce((sum, t) => sum + t.amount, 0);
    const expenseTotal = transactions
        .filter(t => t.type === 'expense')
        .reduce((sum, t) => sum + t.amount, 0);
    const netTotal = incomeTotal - expenseTotal;
    const summaryDiv = document.getElementById('summary');
    summaryDiv.innerHTML = `<h3>Summary</h3>
        <p>Total Income: $${incomeTotal.toFixed(2)}</p>
        <p>Total Expense: $${expenseTotal.toFixed(2)}</p>
        <p>Net Total: $${netTotal.toFixed(2)}</p>`; }
```

Style.css:

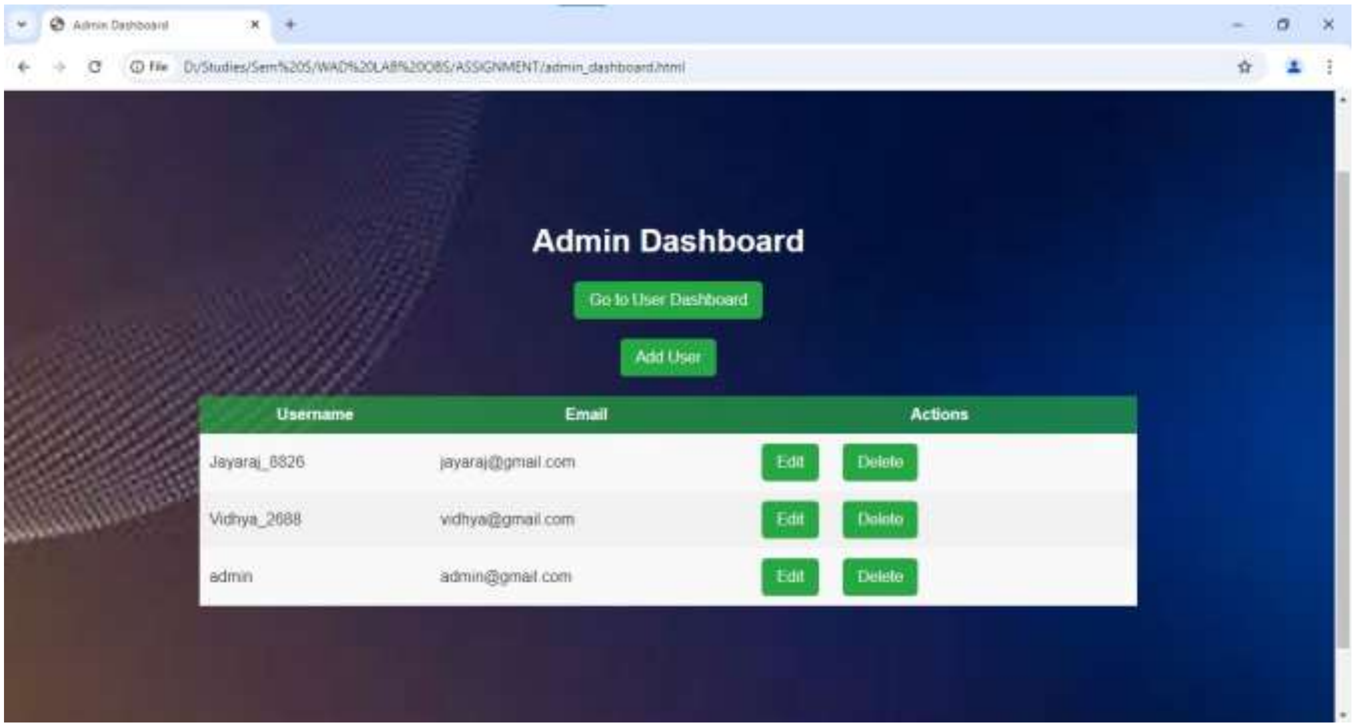
```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 200px;
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    color: white;
}h1 {
    text-align: center;
    color: white; }
.button-container {
    display: flex;
    justify-content: center;
    margin-bottom: 20px;
}
button {
    margin: 0 10px;
    padding: 10px 15px;
    cursor: pointer;
    border: none;
    border-radius: 5px;
    background-color: #28a745;
    color: white;
    font-size: 16px; }
button:hover {
    background-color: #218838; }
#userModal {
    display: none;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(0, 0, 0, 0.8);
    padding: 20px;
    border-radius: 10px;
    max-width: 400px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
    z-index: 1000;
}#userModal input {
    display: block;
    margin: 10px 0;
    width: 100%;
    padding: 10px;
    box-sizing: border-box;
}.modal-overlay {
    display: none;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    z-index: 999;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}th {
    background-color: rgba(40, 167, 69, 0.7);
    color: white;
    padding: 10px; }td {
    background-color: #f9f9f9;
    color: #333;
    padding: 10px;
    text-align: left; }
tr:nth-child(even) td {
    background-color: #f2f2f2;
}tr:hover td {
    background-color: #d1e7dd;}
.color-box {
    width: 20px;
    height: 20px;
    display: inline-block;
    border-radius: 3px; }
.color-box.active {
    background-color: #28a745;
}.color-box.inactive {
    background-color: #dc3545; }
.color-box.pending {
    background-color: #ffc107;
}#summary {
    color: white;
    margin-top: 20px;
}
#error-message {
    color: red;
    text-align: center;
}
```

OUTPUT:

Login_Page:



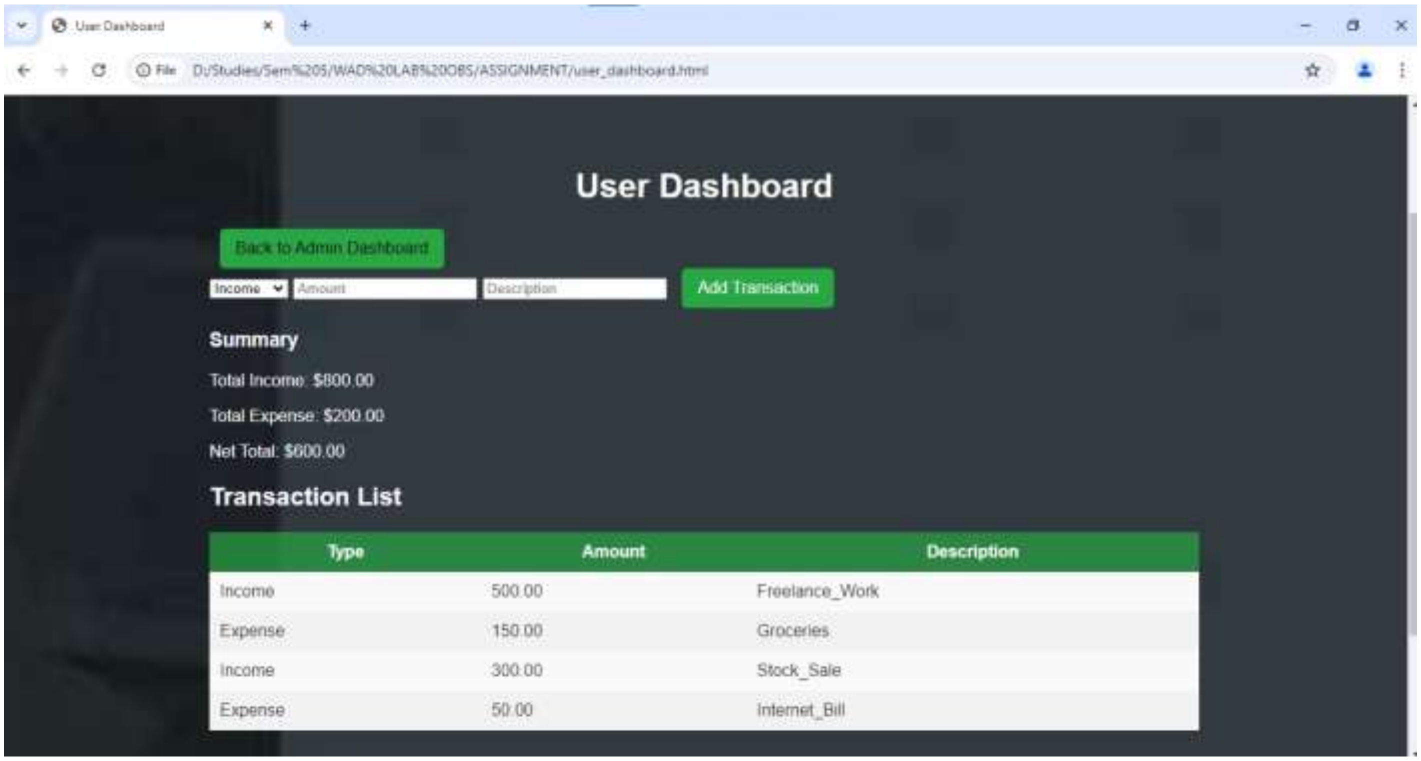
Admin_Dashboard:


Admin Dashboard
D:/Studies/Sem%205/WAD%20LAB%20OBS/ASSIGNMENT/admin_dashboard.html
Go to User Dashboard
Add User

Admin Dashboard

Username	Email	Actions
Jayaraj_8826	jayaraj@gmail.com	Edit Delete
Vidhya_2688	vidhya@gmail.com	Edit Delete
admin	admin@gmail.com	Edit Delete

User_Dashboard:



User Dashboard

Summary

Total Income: $800.00

Total Expense: $200.00

Net Total: $600.00

Transaction List

Type	Amount	Description
Income	500.00	Freelance_Work
Expense	150.00	Groceries
Income	300.00	Stock_Sale
Expense	50.00	Internet_Bill