

**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

**Bank locker management system**

**A CAPSTONE PROJECT REPORT**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE**

**Submitted by**

**B.N. Vivek Reddy (192210527)**

**Y. Rohith (192210716)**

**P. Pradeep Kumar (192210698)**

**Under the Supervision of**

**Ms. B. Jeevashri**

**JUNE 2024**

**DECLARATION**

We, **Vivek, Rohith and Pradeep** students of **Bachelor of Engineering in Computer Science Engineering**, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **Bank Locker Management System** is the outcome of our own Bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

1. B.N Vivek 192210527
2. Y. Rohith 192210716
3. P. Pradeep 192210698

Date:

Place:

**CERTIFICATE**

This is to certify that the project entitled **“Bank Locker Management System”** submitted by **Vivek, Rohith and Pradeep Kumar** has been carried out under my supervision. The project has been submitted as per the requirements in the current semester of B.E Computer Science.

Teacher-in-charge

**Ms. B. Jeevashri**

**Table of Contents**

|  |  |
| --- | --- |
| **S.NO** | **TOPICS** |
|  | **Abstract** |
| 1 | **Introduction** |
| 2 | **Project Description** |
| 3 | **Problem Description** |
| 4 | **Tool Description** |
| 5 | **Operations** |
| 6 | **Approach / Module Description / Functionalities**  6.1 Home  6.2 About Us  6.3 Help  6.4 Contact Us |
| 7 | **Implementation** |
| 8 | **Result** |
| 9 | **Conclusion** |
|  | **References** |

**Abstract**

The Bank Locker Management System represents a sophisticated software solution engineered to modernize and optimize the administration of bank lockers. This system is meticulously crafted to elevate security measures, elevate customer service standards, and enhance the operational efficiency of locker allocations and transactions. Central to its functionality is the comprehensive tracking of locker details, customer information, and transaction histories. Through its secure and intuitive interface, the system empowers bank personnel to efficiently oversee and manage locker operations while upholding stringent security protocols. This abstract provides an overview of the core functionalities and objectives of the Bank Locker Management System, highlighting its pivotal role in fostering operational excellence and safeguarding customer assets.

1. **Introduction**

In response to the growing necessity for heightened security and streamlined operations within banking institutions, the development of a robust and user-friendly Bank Locker Management System has become imperative. This project endeavours to design and implement such a system to cater to these critical requirements. By automating the processes of allocation, monitoring, and management of bank lockers, the system aims to mitigate manual errors, fortify security measures, and elevate overall customer satisfaction. The Bank Locker Management System is envisioned as a comprehensive solution that will revolutionize how banks manage their locker services.

Traditionally, manual methods have been prone to human errors and inefficiencies. Automating these processes not only enhances accuracy but also ensures that customer assets are securely managed with minimal risk of discrepancies or oversights. This project seeks to leverage modern technological advancements to provide a seamless and efficient interface for both bank staff and customers. By centralizing locker allocation and management, the system will enable banks to optimize resource allocation and operational workflows. Moreover, it will empower customers with enhanced transparency and accessibility to their locker details and transaction histories. The following sections will delve into the detailed design, implementation strategies, and anticipated benefits of the Bank Locker Management System, highlighting its potential to significantly improve operational efficiency and security standards within banking environments.

1. **project description**

This project involves the development of a sophisticated Bank Locker Management System tailored to meet the operational needs of banking institutions. The system aims to automate and streamline the processes associated with managing bank lockers, offering robust functionalities to enhance efficiency, security, and customer satisfaction.

**Key Objectives and Features:**

The primary objective of the Bank Locker Management System is to automate and optimize the allocation, monitoring, and access control of bank lockers. It will facilitate seamless customer registration and profile management, ensuring accurate and secure handling of personal and locker-related information. Automated locker allocation algorithms will be employed to efficiently assign lockers based on availability and customer preferences, minimizing manual intervention and optimizing resource utilization.

Security is a paramount feature of the system, incorporating robust access control mechanisms, authentication protocols, and encryption techniques to safeguard locker contents and protect against unauthorized access. Comprehensive transaction logging and reporting capabilities will provide real-time insights into locker usage, facilitating operational transparency, and compliance with regulatory requirements.

**Technological Framework:**

The system will be developed using a high-level programming language known for its scalability and versatility, coupled with a reliable database management system to store and manage large volumes of customer and transaction data securely. The user interface will be designed to be intuitive and user-friendly, ensuring ease of navigation and accessibility for both bank staff and customers.

This project aims to set a new standard in bank locker management by integrating advanced technologies with best practices in security and user experience design. By automating complex processes and enhancing operational efficiencies, the Bank Locker Management System will empower banking institutions to deliver superior service while maintaining the highest standards of security and customer trust.

1. **Problem description**

Traditional methods of managing bank lockers manually pose significant challenges in terms of efficiency, security, and customer service. These manual processes are labor-intensive and prone to errors, leading to potential security breaches and operational inefficiencies. Paper-based records can be easily misplaced, and manual data entry increases the risk of inconsistencies and unauthorized access. These challenges underscore the need for a modernized approach to locker management that integrates automation and stringent security measures. The Bank Locker Management System aims to address these issues by providing an automated and secure platform for managing locker operations. By automating processes such as locker allocation, access control, and transaction monitoring, the system will reduce reliance on manual intervention. This automation not only enhances operational efficiency but also minimizes the risk of human errors that could compromise security or cause disruptions in service delivery.

The system will leverage a high-level programming language and a robust database management system to ensure scalability, reliability, and ease of maintenance. These technological choices are crucial in supporting the system's functionality and facilitating seamless integration with existing banking infrastructure. Moreover, adherence to best practices in software engineering throughout the development lifecycle will uphold the system's quality and reliability, meeting the stringent requirements of modern banking operations. In summary, the Bank Locker Management System represents a proactive solution to the inefficiencies and security risks associated with manual locker management processes. By embracing automation and adopting advanced security protocols, the system aims to elevate service standards, enhance operational transparency, and strengthen customer trust in banking institutions.

1. **Tool description**

**User interface:**

**Programming Languages:**

**• JavaScript:** Used for building interactive and dynamic user interfaces. Often used in combination with frameworks like React or Angular for frontend development.

• **HTML/CSS:** For structuring the user interface and styling elements.

• **XAMPP** tool are also used for this website.

**Frontend Frameworks:**

**• React.js:** A popular JavaScript library for building user interfaces. It allows for the creation of reusable UI components and supports efficient rendering of data.

• **Angular:** A platform and framework for building single-page client applications using HTML and TypeScript.

• **Vue.js:** Another JavaScript framework for building reactive user interfaces.

**Features:**

**Mobile Optimization:** Optimize the currency converter for mobile devices with responsive design and native app development (e.g., using React Native). Ensure consistent performance across various screen sizes and operating systems.

**Feedback Mechanisms:** Implement feedback loops and user surveys to gather input and improve the currency converter based on user preferences and suggestions. Continuous iteration based on user feedback can enhance overall satisfaction and usability

1. **Operations**

The Bank Locker Management System encompasses several critical operations designed to streamline and enhance the management of bank lockers. These operations include:

1. **Login:**
   * Users, including bank staff and administrators, will authenticate securely to access the system. Authentication protocols ensure that only authorized personnel can perform operations within the system.
2. **Assign Locker:**
   * This operation involves the automated assignment of lockers to customers based on availability, size preferences, and other criteria. The system will handle the allocation process seamlessly, reducing manual errors and optimizing resource utilization.
3. **Reports:**
   * The system generates comprehensive reports that provide insights into locker utilization, transaction histories, and operational trends. These reports help management make informed decisions, track performance metrics, and ensure compliance with regulatory requirements.
4. **Search Locker:**
   * Users can search for specific lockers based on various parameters such as locker number, size, or customer information. This functionality facilitates quick retrieval of information and efficient management of locker assignments.
5. **Account Settings - Password Update, Delete Records, etc.:**
   * Customers and authorized personnel can manage their account settings through the system. This includes updating passwords for enhanced security, deleting records when necessary, and modifying personal information. These operations are conducted securely to protect customer data and maintain confidentiality.
6. **Approach**

The development approach for the Bank Locker Management System will involve a structured process to ensure efficiency, security, and usability throughout the project lifecycle.

**Data Source for Locker Management:**

* Ensure the database schema supports all necessary functionalities, including customer registration, locker allocation, and transaction recording.

**Design the Application:**

* Plan the architecture and workflow of the Bank Locker Management System to meet operational requirements and regulatory standards.

**Set Up the Environment:**

* Create a development environment with the necessary software tools and frameworks (e.g., ides, version control systems) to support application development and testing.

**Integrate Backend and Frontend:**

* Design a responsive and intuitive frontend interface using HTML, CSS, and javascript to facilitate user interactions and display locker-related information effectively.

**Test the Application:**

* Conduct rigorous testing procedures, including unit testing, integration testing, and user acceptance testing, to ensure the system operates as intended across different scenarios.

**Deploy the Application:**

* Deploy the Bank Locker Management System on a secure and scalable infrastructure, considering factors such as server architecture, database scaling, and data backup mechanisms.

**Maintenance and Updates:**

* Establish a maintenance plan to monitor system performance, address user feedback, and implement necessary updates or patches.

**Module Description:**

**User Interface (UI) Module**

* **Description:**
  + 1. Home
    2. About Us
    3. Help
    4. Contact Us

**Functionalities**

**Core Functionalities:**

* 1. **Locker Allocation:** Automated assignment of lockers based on availability and customer preferences.
  2. **Transaction Monitoring:** Real-time recording and tracking of locker-related activities, including deposits, withdrawals, and maintenance.
  3. **Customer Registration:** Secure collection and management of customer information for locker access.

**Additional Functionalities:**

* 1. **Audit Trail:** Comprehensive logging of all actions related to locker operations for security and compliance purposes.
  2. **Notification System:** Alerts and notifications for customers regarding locker availability, maintenance schedules, and policy updates.
  3. **Reporting:** Generation of reports and analytics to provide insights into locker utilization, operational trends, and compliance metrics.

**Advanced Functionalities:**

* 1. **Security Enhancements:** Integration of advanced security measures such as biometric authentication, CCTV monitoring, and encrypted data transmission.
  2. **Integration with Banking Systems:** Seamless integration with core banking systems for real-time updates and synchronized data management.
  3. **Mobile Accessibility:** Development of a mobile-friendly interface or dedicated mobile application for customers to manage lockers remotely.

1. **Implementation**

**Login Page:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login - Bank Locker Management System</title>

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

</head>

<body>

<div class="container">

<div class="login-container">

<h2>Bank Locker</h2>

<form action="verify.php" method="post">

<div class="form-group">

<label for="username">Username:</label>

<input type="text" placeholder="Enter your username" id="username" name="username" required>

</div>

<div class="form-group">

<label for="password">Password:</label>

<input type="password" placeholder="password" id="password" name="password" required>

</div>

<button type="submit">Login</button>

<p>New? Register Here<a href="signup.html">sign up</a></p>

</form>

<p id="error-message" class="error-message"></p>

</div>

</div>

</body>

</html>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Define your database connection details

$servername = "localhost:3307";

$dbusername = "root";

$dbpassword = "";

$dbname = "banklocker";

// Create connection

$conn = new mysqli($servername, $dbusername, $dbpassword, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Process form data

$username = $\_POST["username"];

$password = $\_POST["password"];

// Check if username already exists

$check\_username\_query = "SELECT \* FROM login WHERE Username='$username' and Password='$password'";

$result = $conn->query($check\_username\_query);

if ($result->num\_rows > 0) {

echo "Authentication Verified";

header('Location:home.html');

exit;

} else {

echo "Access Denied " . $check\_username\_query . "<br>" . $conn->error;

}

// Close connection

$conn->close();

}

?>

body {

font-family: Arial, sans-serif;

/\* Blue background \*/

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-image:url('images/1.jpg');

background-size: cover; /\* Ensures the image covers the entire background \*/

background-repeat: no-repeat; /\* Prevents the image from repeating \*/

background-position:center;

}

.container {

display: flex;

justify-content: center;

align-items: center;

height: 80%;

}

.login-container {

background-color: #b1c8df; /\* White background \*/

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

width: 200px;

}

h2 {

text-align: center;

font-family:Arial, Helvetica, sans-serif;

margin-bottom: 20px;

color: #000000; /\* Blue text \*/

font-weight:bold;

}

.form-group {

margin-bottom: 15px;

}

p {

font-family: Arial, sans-serif;

font-size: 10px;

color: #333;

}

a {

color: #007bff; /\* Blue color for links \*/

text-decoration: none; /\* Remove underline \*/

font-weight: bold; /\* Make text bold \*/

margin-left: 5px; /\* Add some space between 'New? Register Here' and 'Sign Up' \*/

}

a:hover {

text-decoration: underline; /\* Underline on hover \*/

cursor:pointer;

}

label {

font-weight: bold;

display: block;

margin-bottom: 5px;

color: #323232b4; /\* Dark grey text \*/

}

input[type="text"],

input[type="password"] {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

box-sizing: border-box;

}

button {

width: 100%;

padding: 12px;

background-color: #2980b9; /\* Darker blue button \*/

color: #fff;

border: none;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background-color: #1a65ac; /\* Slightly darker on hover \*/

}

.error-message {

color: red;

text-align: center;

margin-top: 10px;

display: none;

}

**Register page:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>SignUp-Bank Locker</title>

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

</head>

<body>

<div class="container">

<div class="login-container">

<h2>Sign Up</h2>

<form action="index.php" method="post">

<div class="form-group">

<label for="username">Username:</label>

<input type="text" placeholder="create username" id="username" name="username" required>

</div>

<div class="form-group">

<label for="password">Create Password:</label>

<input type="password" placeholder="password" id="password" name="password" required>

</div>

<div class="form-group">

<label for="username">email:</label>

<input type="text" placeholder="Enter your email" id="username" name="email" required>

</div>

<button type="submit">Sign UP</button>

<p><a href="index.html">sign in</a></p>

</form>

<p id="error-message" class="error-message"></p>

</div>

</div>

</body>

</html>

**Home page:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Banker Dashboard</title>

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

</head>

<body>

<div class="container">

<aside class="sidebar">

<h2>Bank Locker</h2>

<ul>

<li>Dashboard</li>

<li><a href="search.html">Locker Details</a></li>

<li><a href="sub.html">Assign Locker</a></li>

<li><a href="pages.html">Pages</a></li>

<li><a href="settings.html">Account Settings</a></li>

</ul>

</aside>

<main class="content">

<section id="dashboard">

<h2>Dashboard</h2>

<style>

.user-box {

width: 200px;

padding: 20px;

background-color: #5493df;

border: 1px solid #ccc;

border-radius: 8px;

text-align: center;

font-size: 1.2em;

cursor: pointer; /\* Make the box clickable \*/

}

.user-list {

display: none; /\* Hide user list by default \*/

margin-top: 10px;

padding: 10px;

background-color: #e0e0e0;

border: 1px solid #aaa;

border-radius: 5px;

text-align: left;

font-size: 1em;

}

table {

width: 100%;

border-collapse: collapse;

}

table, th, td {

border: 1px solid black;

padding: 8px;

}

th {

background-color: #f2f2f2;

}

</style>

<script>

// Function to fetch and display users on box click

function displayUsers() {

var userList = document.getElementById('user-list');

// Check if user list is already visible

if (userList.style.display === 'block') {

userList.style.display = 'none';

} else {

// Fetch user data from server using AJAX

var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {

if (xhr.readyState === XMLHttpRequest.DONE) {

if (xhr.status === 200) {

// Display fetched user data

userList.innerHTML = xhr.responseText;

userList.style.display = 'block'; // Show user list

} else {

alert('Error fetching user data: ' + xhr.status);

}

}

};

xhr.open('GET', 'dashboard.php', true);

xhr.send();

}

}

</script>

<div class="user-box" onclick="displayUsers()">

Active Users <span id="user-count">

<!-- PHP will populate this span with the user count -->

</span>

</div>

<div id="user-list" class="user-list">

<!-- User list fetched dynamically via AJAX -->

</div>

</main>

</div>

<footer>

<div class="container">

<p>&copy; 2024 Your Bank. All rights reserved.</p>

</div>

</footer>

</body>

</html>

**Dashboard:**

<?php

// Database configuration

$servername = "localhost:3307";

$username = "root";

$password = "";

$dbname = "banklocker";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Query to get users

$sql = "SELECT fullname, email, mobile\_number, address, IDproof, locker\_number, key\_number FROM locker";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

// Output data of each row

echo "<table border='1'>

<tr>

<th>Full Name</th>

<th>Email</th>

<th>Mobile Number</th>

<th>Address</th>

<th>ID Proof</th>

<th>Locker Number</th>

<th>Key Number</th>

</tr>";

while($row = $result->fetch\_assoc()) {

echo "<tr>";

echo "<td>" . htmlspecialchars($row["fullname"]) . "</td>";

echo "<td>" . htmlspecialchars($row["email"]) . "</td>";

echo "<td>" . htmlspecialchars($row["mobile\_number"]) . "</td>";

echo "<td>" . htmlspecialchars($row["address"]) . "</td>";

echo "<td>" . htmlspecialchars($row["IDproof"]) . "</td>";

echo "<td>" . htmlspecialchars($row["locker\_number"]) . "</td>";

echo "<td>" . htmlspecialchars($row["key\_number"]) . "</td>";

echo "</tr>";

}

echo "</table>";

} else {

echo "0 results";

}

// Close connection

$conn->close();

?>

**Assign locker:**

<?php

// Check if the form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Database connection details

$servername = "localhost:3307";

$username = "root";

$password = "";

$dbname = "banklocker";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Get form data

$full\_name = mysqli\_real\_escape\_string($conn, $\_POST['fullname']);

$email = mysqli\_real\_escape\_string($conn, $\_POST['email']);

$mobile\_number = mysqli\_real\_escape\_string($conn, $\_POST['mobile\_number']);

$address = mysqli\_real\_escape\_string($conn, $\_POST['address']);

$id\_proof = mysqli\_real\_escape\_string($conn, $\_POST['IDproof']);

$locker\_number = mysqli\_real\_escape\_string($conn, $\_POST['locker\_number']);

$key\_number = mysqli\_real\_escape\_string($conn, $\_POST['key\_number']);

// Insert data into database

$sql = "INSERT INTO locker (fullname, email, mobile\_number, address, IDproof, locker\_number, key\_number)

VALUES ('$full\_name', '$email', '$mobile\_number', '$address', '$id\_proof', '$locker\_number', '$key\_number')";

if ($conn->query($sql) === TRUE) {

echo "Locker assigned successfully!";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

}

?>

**Locker Details:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Locker Search</title>

<style>

body {

font-family: Arial, sans-serif;

background-image: url(images/m1.jpg); /\* Replace with your actual background image \*/

background-size: cover;

background-repeat: no-repeat;

background-position: center;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

width: 100%;

max-width: 400px;

background-color: rgba(255, 255, 255, 0.9); /\* Semi-transparent white background \*/

padding: 30px;

border-radius: 8px;

box-shadow: 0 0 15px rgba(0, 0, 0, 0.3);

text-align: center;

}

h2 {

color: #333;

margin-bottom: 20px;

}

form {

display: flex;

flex-direction: column;

align-items: center;

}

label {

font-size: 1.1em;

margin-bottom: 10px;

color: #555;

}

input[type="text"] {

width: 100%;

padding: 10px;

margin-bottom: 20px;

border: 1px solid #ccc;

border-radius: 5px;

font-size: 1em;

box-sizing: border-box;

transition: border-color 0.3s;

}

input[type="text"]:focus {

outline: none;

border-color: #4CAF50; /\* Green border color on focus \*/

}

button {

width: 100%;

padding: 12px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 1.1em;

transition: background-color 0.3s;

}

button:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<div class="container">

<h2>Search for Locker or Key Number</h2>

<form action="search1.php" method="POST">

<label for="number">Enter Locker or Key Number:</label>

<input type="text" id="number" name="number" required>

<button type="submit">Search</button>

</form>

</div>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Search Results</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

padding: 20px;

}

.container {

max-width: 600px;

margin: 50px auto;

padding: 20px;

background-color: #fff;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

h2 {

text-align: center;

color: #333;

}

p {

margin-bottom: 10px;

font-size: 1.1em;

line-height: 1.6;

}

.no-results {

text-align: center;

color: #777;

}

.go-home {

text-align: center;

margin-top: 20px;

}

.go-home a {

text-decoration: none;

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border-radius: 5px;

transition: background-color 0.3s;

}

.go-home a:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<?php

// Database connection parameters

$servername = "localhost:3307"; // Adjust port if necessary

$username = "root";

$password = "";

$dbname = "banklocker";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Initialize variables to store search results

$fullname = $email = $mobile\_number = $address = $IDproof = $locker\_number = "";

// Retrieve search input

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$search\_number = $\_POST['number'];

// Prepare SQL statement

$sql = "SELECT fullname, email, mobile\_number, address, IDproof, locker\_number FROM locker WHERE locker\_number = ? OR key\_number = ?";

// Use prepared statements to prevent SQL injection

$stmt = $conn->prepare($sql);

$stmt->bind\_param("ss", $search\_number, $search\_number);

// Execute the statement

$stmt->execute();

// Bind result variables

$stmt->bind\_result($fullname, $email, $mobile\_number, $address, $IDproof, $locker\_number);

// Fetch values

if ($stmt->fetch()) {

echo "<div class='container'>";

echo "<h2>Search Result:</h2>";

echo "<p><strong>Fullname:</strong> $fullname</p>";

echo "<p><strong>Email:</strong> $email</p>";

echo "<p><strong>Mobile Number:</strong> $mobile\_number</p>";

echo "<p><strong>Address:</strong> $address</p>";

echo "<p><strong>ID Proof:</strong> $IDproof</p>";

echo "<p><strong>Locker Number:</strong> $locker\_number</p>";

echo "</div>";

} else {

echo "<div class='container no-results'>";

echo "<h2>No results found for '$search\_number'.</h2>";

echo "</div>";

}

// Close statement

$stmt->close();

}

// Close connection

$conn->close();

?>

<div class="go-home">

<a href="home.html">Go Home</a>

</div>

</body>

</html>

**Account Settings:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Account Settings</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

padding: 20px;

}

.container {

max-width: 600px;

margin: 20px auto;

padding: 20px;

background-color: #fff;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

h2 {

color: #333;

text-align: center;

margin-bottom: 20px;

}

form {

display: flex;

flex-direction: column;

}

label {

margin-bottom: 10px;

color: #555;

}

input[type="text"], input[type="password"] {

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

margin-bottom: 10px;

}

button {

padding: 10px 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #45a049;

}

.go-home {

position: fixed;

bottom: 20px; /\* Adjust as needed \*/

left: 50%;

transform: translateX(-50%);

background-color: #4CAF50;

color: white;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

.go-home:hover {

background-color: #45a049;

}

</style>

</head>

<body>

<div class="container">

<h2>Account Settings</h2>

<!-- Form to Update Password -->

<form action="update.php" method="POST">

<label for="username">Enter Username to Update Password:</label>

<input type="text" id="username" name="username" required>

<label for="password">New Password:</label>

<input type="password" id="password" name="password" required>

<button type="submit">Update Password</button>

</form>

<!-- Form to Delete Record -->

<form action="deletion.php" method="POST">

<label for="username">Enter Username to Delete Account:</label>

<input type="text" id="username" name="username" required>

<button type="submit">Delete Account</button>

</form>

<button class="go-home" onclick="window.location.href='home.html';">Go Home</button>

</div>

</body>

</html>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Define your database connection details

$servername = "localhost:3307";

$dbusername = "root";

$dbpassword = "";

$dbname = "banklocker";

// Create connection

$conn = new mysqli($servername, $dbusername, $dbpassword, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Process form data

$username = $\_POST["username"];

$password = $\_POST["password"];

// Check if username already exists

$check\_username\_query = "SELECT \* FROM login WHERE Username='$username'";

$result = $conn->query($check\_username\_query);

if ($result->num\_rows > 0) {

// Username already exists, update password

$update\_query = "UPDATE login SET Password='$password' WHERE Username='$username'";

if ($conn->query($update\_query) === TRUE) {

echo "Password updated successfully";

} else {

echo "Error updating password: " . $conn->error;

}

} else {

// Username doesn't exist, display error message

echo "Username does not exist. Please register first.";

}

// Close connection

$conn->close();

}

?>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Define your database connection details

$servername = "localhost:3307";

$dbusername = "root";

$dbpassword = "";

$dbname = "banklocker";

// Create connection

$conn = new mysqli($servername, $dbusername, $dbpassword, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Process form data

$username = $\_POST["username"];

// Check if username exists

$check\_username\_query = "SELECT \* FROM login WHERE username='$username'";

$result = $conn->query($check\_username\_query);

if ($result->num\_rows > 0) {

// Username exists, proceed with deletion

$delete\_query = "DELETE FROM login WHERE username='$username'";

if ($conn->query($delete\_query) === TRUE) {

echo "Record deleted successfully";

} else {

echo "Error deleting record: " . $conn->error;

}

} else {

// Username doesn't exist, display error message

echo "Username does not exist. Cannot delete.";

}

// Close connection

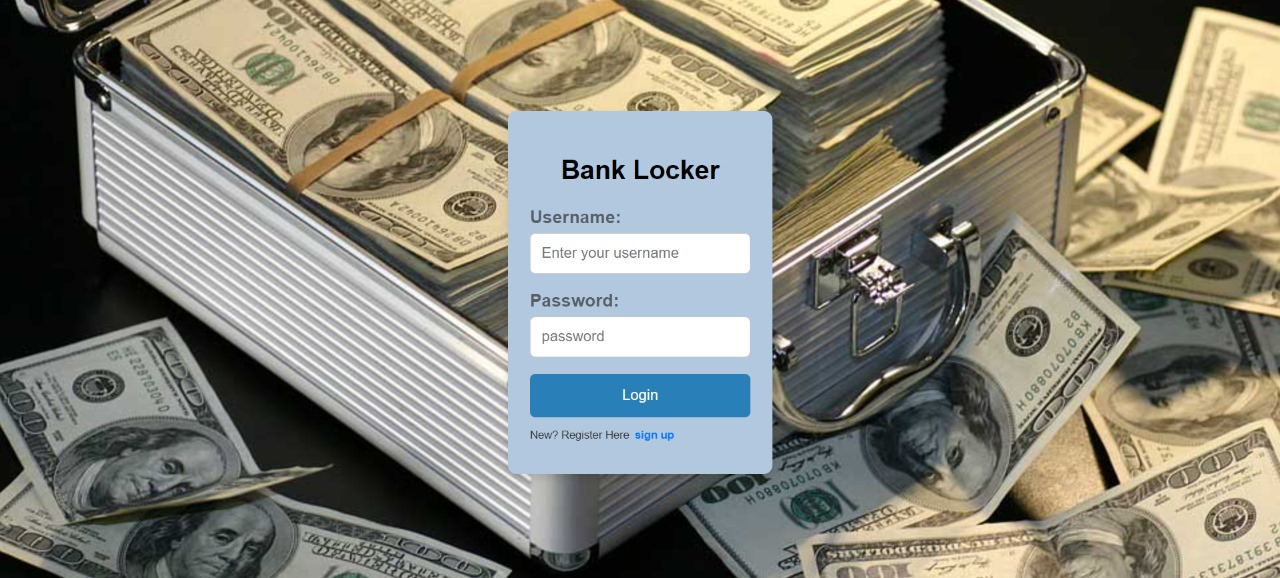
$conn->close();

}

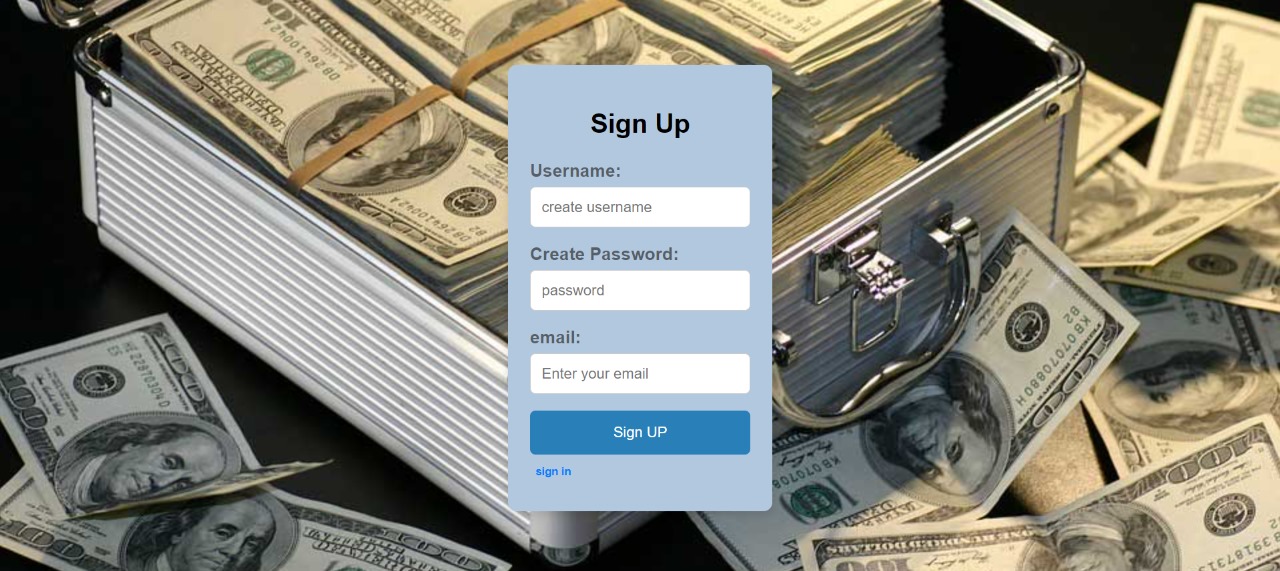
?>

1. **Result**

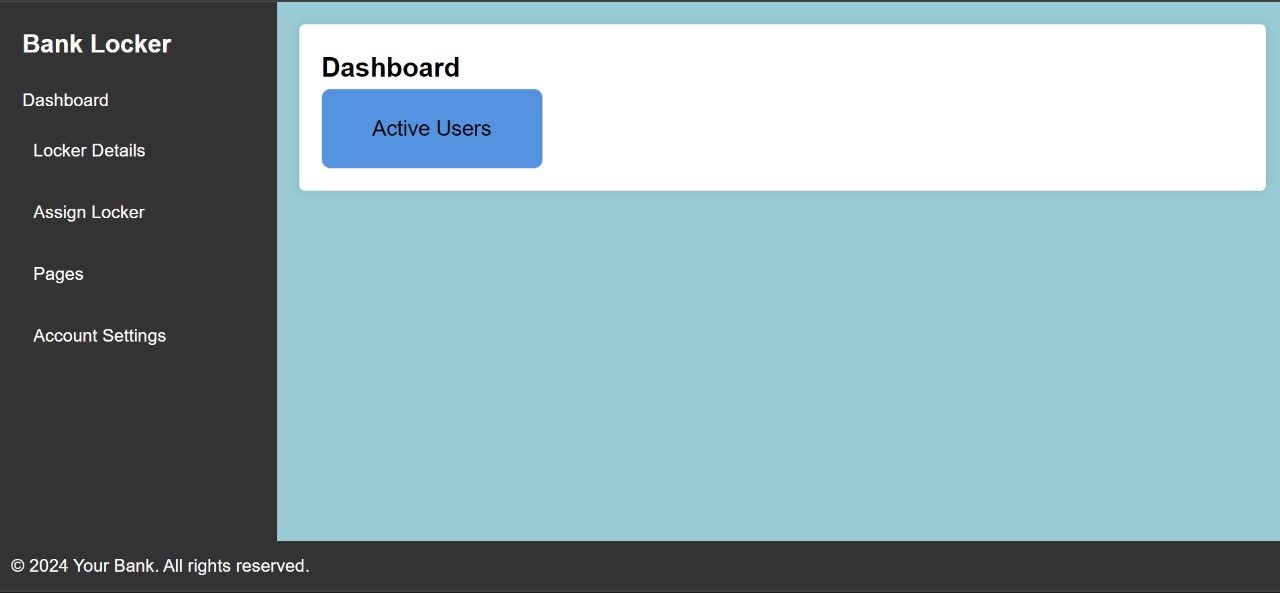
**Login Page:**



**Sign-up Page:**



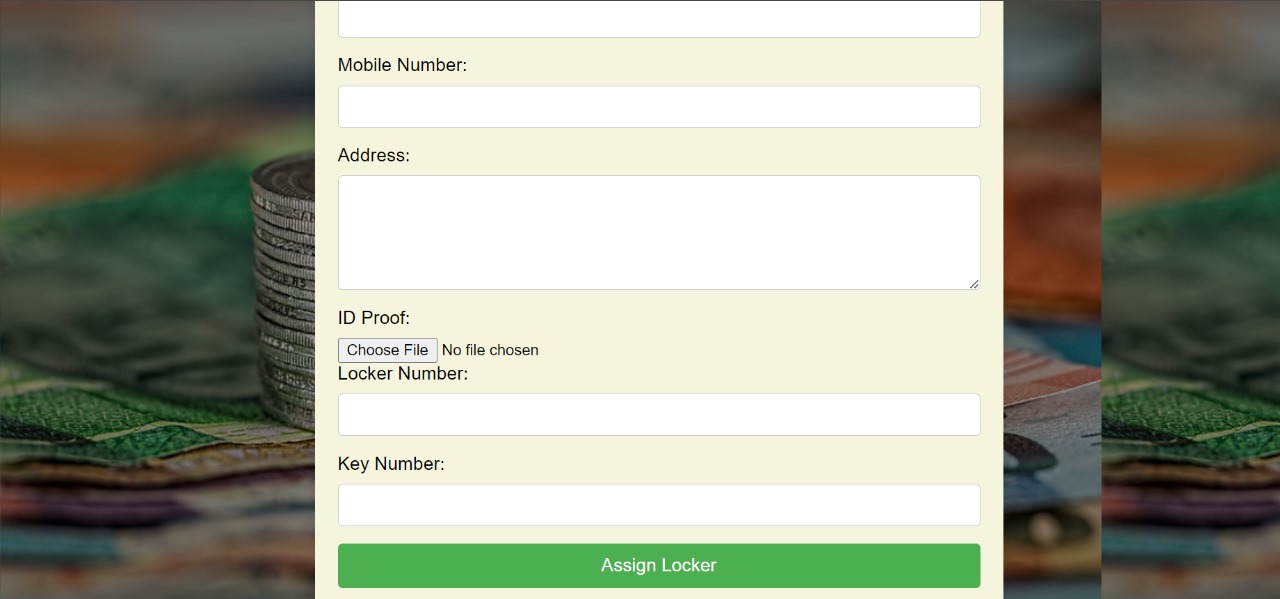
**Dashboard:**



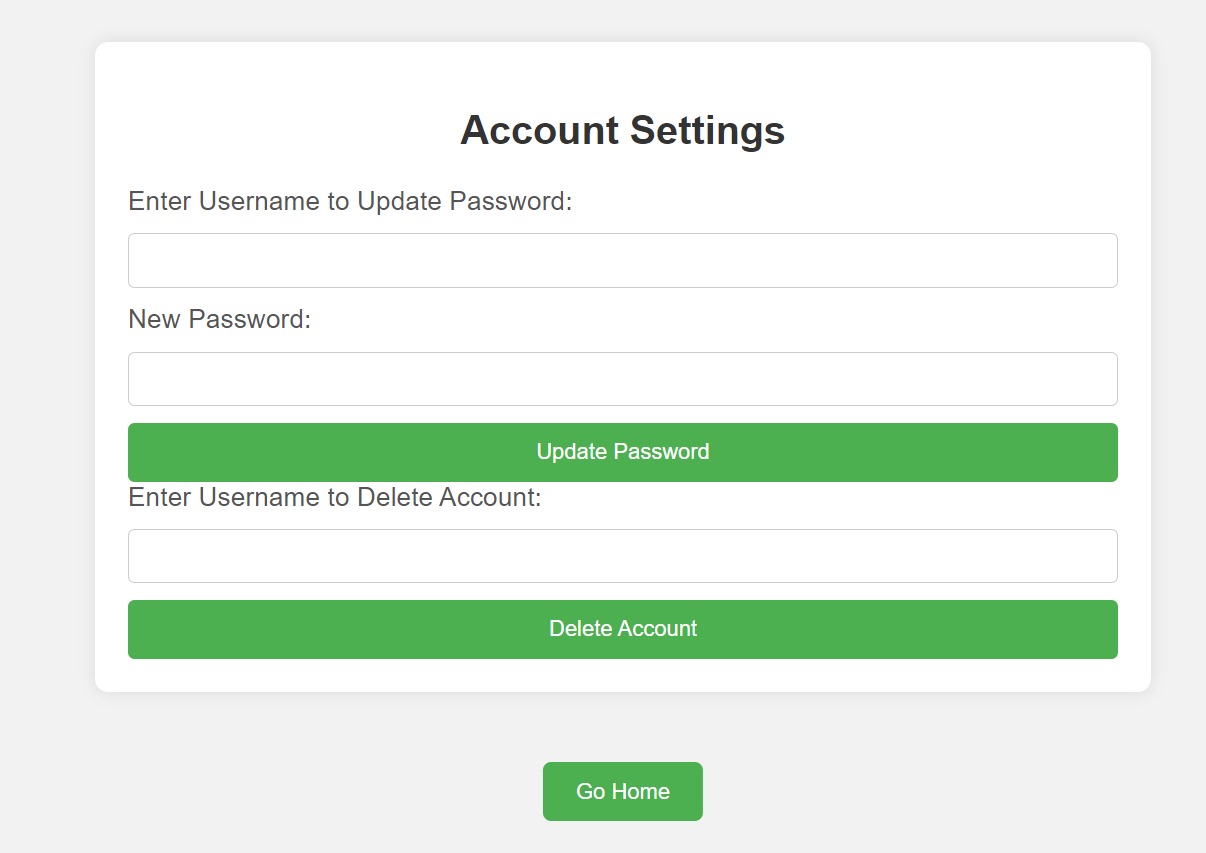
**Search Details**



**Assign Locker:**



**Account Settings**



1. **Conclusion**

The Bank Locker Management System is a comprehensive solution for managing bank lockers efficiently and securely. It automates many of the manual processes involved in locker management, reduces errors, and improves customer service. The system is scalable, secure, and easy to use, making it an ideal solution for banks looking to improve their locker management processes. In conclusion, the Bank Locker Management System represents a significant advancement in the realm of banking operations, particularly concerning the efficient and secure management of bank lockers. By automating key processes and integrating robust security measures, the system addresses the challenges associated with traditional manual methods.

The system's ability to automate locker allocation, monitor transactions in real-time, and manage customer interactions effectively enhances operational efficiency and reduces the likelihood of errors. This automation not only streamlines day-to-day operations but also improves overall service delivery and customer satisfaction. Moreover, the Bank Locker Management System is designed with scalability in mind, allowing banks to expand their locker services seamlessly as their customer base grows. The system's architecture and technology choices ensure it can adapt to evolving banking regulations and technological advancements, thereby future-proofing its capabilities.

By implementing this system, banks can expect tangible benefits such as improved security protocols, streamlined administrative tasks, and enhanced transparency in locker operations. Customers, in turn, benefit from a more convenient and secure locker management experience, reinforcing trust and loyalty to the banking institution. In essence, the Bank Locker Management System stands as a cornerstone in modern banking operations, demonstrating its effectiveness in enhancing operational efficiency, security, and customer service standards. As banks continue to embrace digital transformation, solutions like this play a crucial role in shaping the future landscape of banking services.

**References**

 **PHPGurukul:** PHPGurukul provides various tutorials and resources related to PHP development, including projects like bank locker management systems. You can visit their website for specific tutorials and code examples.

 **Stack Overflow:** Stack Overflow is a popular community-driven platform where programmers ask and answer questions related to programming. You can find discussions, solutions, and best practices related to implementing bank locker management systems using various technologies, including PHP.

 **W3Schools:** W3Schools offers tutorials and references on web development technologies, including PHP, HTML, CSS, JavaScript, and more. It provides practical examples and explanations that can help you understand and implement features of a bank locker management system.