M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

SIMPLE BANKING APPLICATION

Submitted in partial fulfillment of requirements for the award of the course

of

CGB1221-DATABASE MANAGEMENT SYSTEMS

Under the guidance of

Ms.A.Nithyasri

Assistant Professor / AI

Submitted By

NARESH D **(927623BAD067)**

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

MAY 2025

M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

This is to certify that this project report on **"SIMPLE BANKING APPLICATION"** is the bonafide work of **NARESH D (927623BAD067)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Ms. A. NITHYASRI, M.E., (P.hd)

SUPERVISOR,

Department of Artificial Intelligence,
M. Kumarasamy College of Engineering,
Thalavapalayam, Karur - 639 113.

Signature

Dr. A. SELVI, M.Tech., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,
M. Kumarasamy College of Engineering,
Thalavapalayam, Karur - 639 113.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To excel in education, innovation, and research in Artificial Intelligence and Data Science to fulfil industrial demands and societal expectations.

MISSION OF THE DEPARTMENT

M1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

M2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

M3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

M4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Graduates will be able to:

- **PEO 1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science.
- **PEO 2:** Provide industry-specific solutions for the society with effective communication and ethics.
- **PEO 3:** Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, give and receive clear instructions.

10. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
11. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Capable of finding the important factors in large datasets, simplify the data, and improve predictive model accuracy.
- **PSO2:** Capable of analyzing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

Simple Banking Application, is a secure and user-friendly web platform designed to simplify and automate core banking operations. It allows users to create an account by submitting necessary personal details and identity proofs, after which a unique account number is automatically generated. Once registered, users can perform essential banking functions such as deposit, withdrawal, fund transfer, and apply for loans, all through an intuitive interface. On the administrative side, the admin panel provides powerful tools for managing user data, verifying documents, monitoring transaction activities, and processing loan approvals. Built using PHP and MySQL, the system ensures robust data encryption, role-based access control, and structured storage of profile images and account details. With its clean architecture and real-time validations, this system delivers a seamless and secure banking experience, reducing manual effort and enhancing operational efficiency for both users and administrators.

Keywords: *Simple Banking Application, Account Number, Personal Details, Identity Proofs, Deposit, Withdrawal, Fund Transfer, Loans, Admin Panel, PHP, MySQL, Data Encryption, Access Control*

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	COs MAPPED	POs MAPPED	PSOs MAPPED
Simple Banking Application, is a secure and user-friendly web platform designed to simplify and automate core banking operations. It allows users to create an account by submitting necessary personal details and identity proofs, after which a unique account number is automatically generated. Once registered, users can perform essential banking functions such as deposit, withdrawal, fund transfer, and apply for loans, all through an intuitive interface. On the administrative side, the admin panel provides powerful tools for managing user data, verifying documents, monitoring transaction activities, and processing loan approvals. Built using PHP and MySQL, the system ensures robust data encryption, role-based access control, and structured storage of profile images and account details. With its clean architecture and real-time validations, this system delivers a seamless and secure banking experience, reducing manual effort and enhancing operational efficiency for both users and administrators.	CO1 CO2 CO3 CO4	PO1(3) PO2(3) PO3(3) PO5(2) PO10(3) PO11(1)	PSO1(2) PSO2(1)

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 PROBLEM STATEMENT

In the current landscape, traditional banking processes remain largely manual, slow, and prone to errors, creating frustration for users and inefficiencies for banks. Customers face long queues, delayed transactions, and limited access to their account information outside working hours. Additionally, existing systems often lack a unified platform that seamlessly handles both customer operations like deposits, withdrawals, transfers, and loans, alongside effective administrative controls for managing user data and transactions. This gap calls for a Simple Banking Application that digitizes and automates core banking functions, delivering real-time, secure, and user-friendly experiences. The solution aims to bridge the divide between customer convenience and robust administrative management, reducing dependency on physical infrastructure and enhancing overall banking efficiency.

1.2 INTRODUCTION

This project presents a Simple Banking Application designed to modernize and streamline everyday banking activities. By harnessing the power of digital technology, it offers a user-centric platform that enables customers to effortlessly manage their finances without the limitations of traditional banking hours or paperwork. Core operations such as deposit, withdrawal, fund transfer, and loan management are integrated into a seamless interface, allowing users to execute transactions swiftly and securely. Beyond customer convenience, the system empowers administrators with efficient tools to oversee user accounts, monitor transactions, and maintain data integrity, ensuring transparency and security. This holistic approach bridges the gap between frontline banking services and backend administration, transforming how banks interact with customers. By replacing manual processes with automation, the application reduces operational overhead, minimizes human error, and enhances accessibility. This project encapsulates the evolution of banking toward digital ecosystems that prioritize speed, security, and simplicity, delivering a robust yet intuitive solution to meet the demands of modern financial management.

1.3 OBJECTIVE

The main goal of this project is to develop a user-friendly Simple Banking Application that supports both customer and admin functions on a single platform. It simplifies key operations like deposits, withdrawals, transfers, and loan processing, enabling users to transact anytime, anywhere. For administrators, it offers tools to manage accounts, verify users, track transactions, and ensure regulatory compliance. Security is reinforced through encrypted passwords and session management. By automating routine tasks, the system reduces manual errors and processing time. Overall, it aims to enhance user experience, promote financial inclusion, and boost banking efficiency.

1.4 DBMS

A Database Management System (DBMS) is software that allows efficient creation, management, and manipulation of databases. It provides a structured way to store, retrieve, update, and delete data while ensuring data integrity, security, and consistency. Acting as a bridge between the database and users or applications, a DBMS enables controlled access using query languages like SQL. It supports key features such as transaction management, concurrency control, and data backup and recovery, which are crucial for maintaining reliable banking operations. Common DBMS examples include MySQL, Oracle, SQL Server, and PostgreSQL, all of which can effectively handle the data needs of a banking application.

1.5 PHP

PHP (Hypertext Preprocessor) is a widely-used, open-source server-side scripting language specifically designed for web development. Embedded within HTML, it enables the creation of dynamic, interactive web pages by processing scripts on the server and sending the resulting HTML to the client's browser. PHP integrates seamlessly with databases like MySQL, making it ideal for developing data-driven web applications such as banking systems. It supports key web functionalities including form processing, session and cookie management, file handling, and user authentication. Platform-independent and compatible with popular web servers like Apache and Nginx, PHP is valued for its simplicity, flexibility, and robust community support. It powers many prominent websites and platforms, including WordPress and parts of Facebook.

CHAPTER 2
PROJECT METHODOLOGY

2.1 PROPOSED WORK

User Workflow:

1. **Visit the Application:** Users land on the homepage of the banking application where they can explore basic info and features.
2. **Check Account Status:** If the user already has an account, they can proceed to login. If not, they are prompted to register before accessing any services.
3. **User Registration:** New users fill out their personal details (Name, Aadhaar, PAN, Contact Info) and upload a photo. Upon successful registration, the system generates a unique account number.
4. **Login:** Registered users log in using their username and password.
5. **Dashboard Access:** Once logged in, users are greeted with a dashboard showing account summary, balance, and navigation options.
6. **Deposit Money:** Select the *Deposit* option, enter the desired amount, and funds are added to the account.
7. **Withdraw Funds:** Choose *Withdraw*, enter amount (validated against available balance), and confirm.
8. **Transfer Money:** Users can transfer money to other users by entering the recipient's account number and the transfer amount.
9. **Apply for Loan:** Users submit income and job details to request a loan. Status can be tracked through the dashboard after admin review.
10. **View Transaction History:** Complete transaction logs of deposits, withdrawals, transfers, and loan activity are available.
11. **Profile Management:** Users can update their personal info, change passwords, or modify settings.

Admin Workflow:

1. **Admin Login:** Admin accesses the system with secure credentials to enter the admin dashboard.
2. **Review User Registrations:** New user accounts are listed for admin to verify and approve before activation.
3. **Dashboard Monitoring:** Admin dashboard displays total users, loan requests, and financial summaries.
4. **Manage Users:** Admin can view, edit, or remove users, and investigate unusual activities if flagged.
5. **Monitor Transactions:** Full visibility of all user transactions to ensure transparency and security.
6. **Loan Management:** Review submitted loan requests and approve/reject them based on user eligibility and criteria.
7. **System Integrity Oversight:** Admin ensures database health, backup status, and general maintenance of the platform.

2.2 ARCHITECTURE

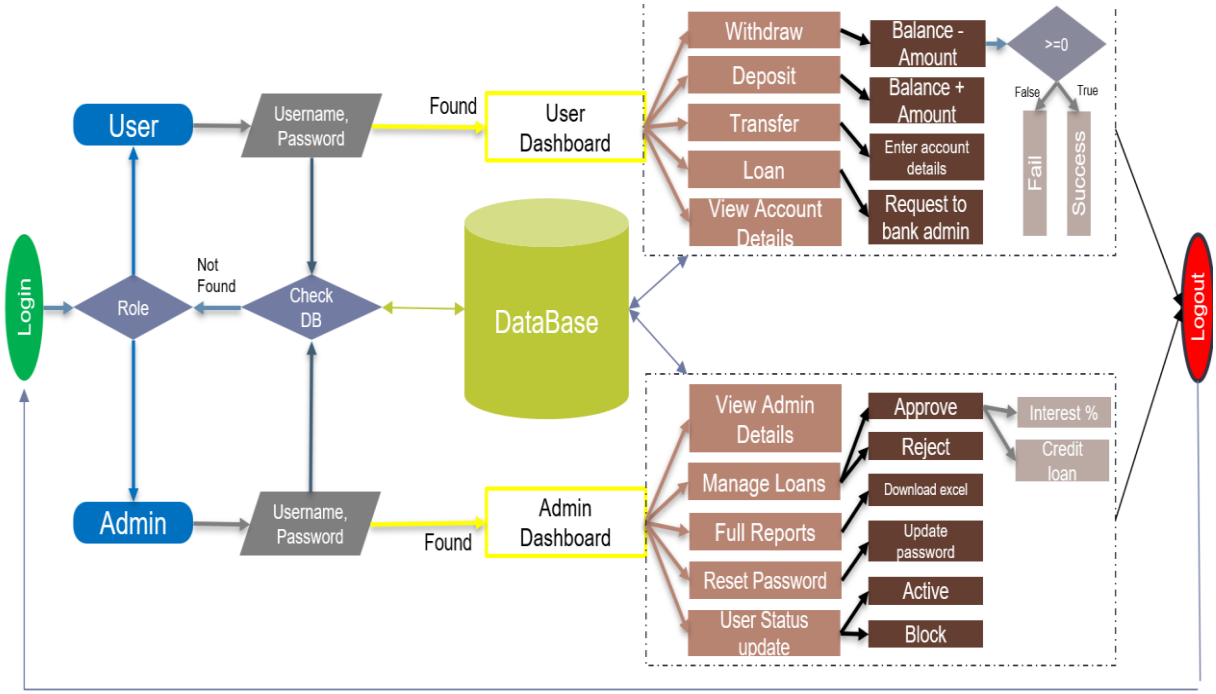


Fig. 2.2.1 Proposed system architecture

2.3 E-R DIAGRAM

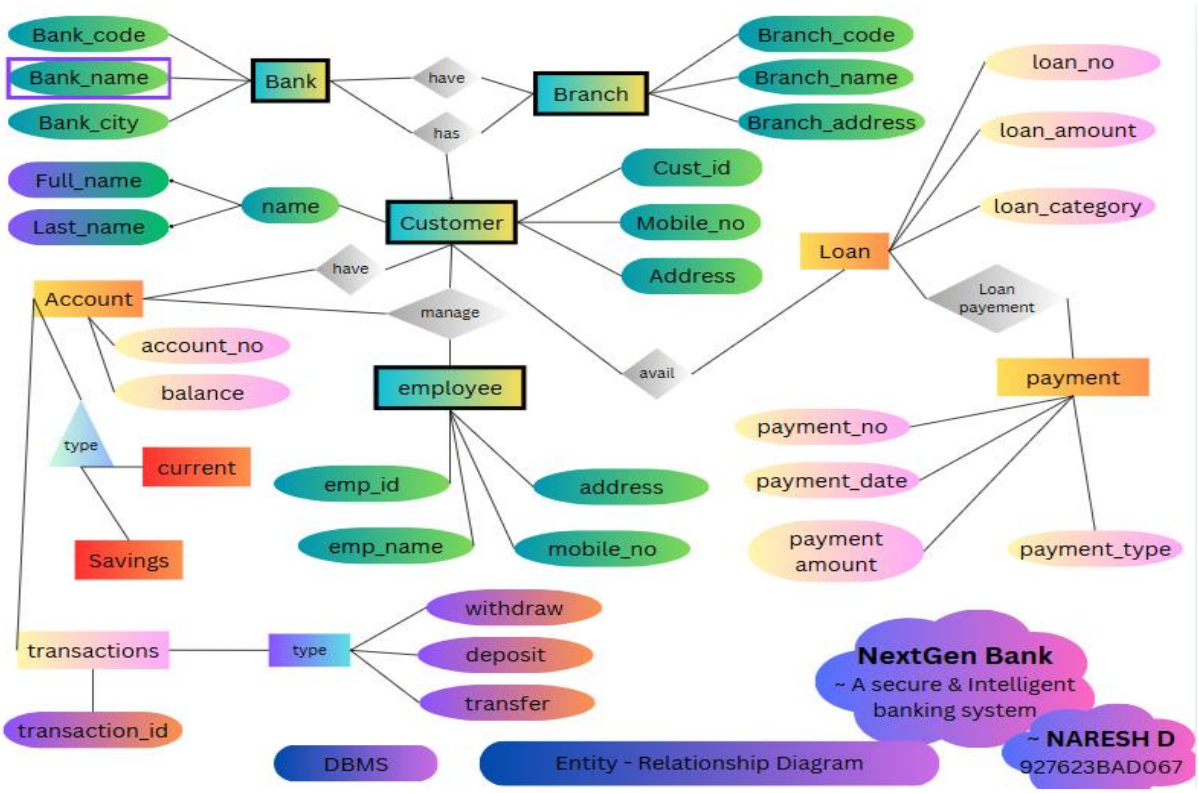


Fig. E-R diagram

2.4 SCHEMA DIAGRAM

Accounts table

id	User_id	Account_number	balance	status
5	12	800000000012	4840.99	active

Loans table

id	User_id	amount	Duration_in_months
7	12	4000	4

Interest_rate	status	Approved_by	Created_at
28.00	Approved	1	2025-05-16 21:05:11

Loan_repayments

id	User_id	Loan_id	amount	Created_at
7	12	7	1059.01	2025-05-16 21:11:11

Transaction table

id	Sender_id	Receiver_id	amount	description	Created_at
1	5	5	12.00	Self Deposit	2025-05-15 16:29:48

Users table

id	Account_number	First_name	Last_name	Father_name	Mother_name
12	800000000012	ram	kumaran	dhamandhren	senbhi

address	Phone1	Phone2	Aadhar_number	Pancard_number	Voter_id
20/2 south, velur	07200754466	7200754565	987654321911	fhidkfhd483	465fnf5y7u

age	dob	gender	Differently_abled	Working_details	Is_salaried
20	2006-07-30	Male	No	general manager in google	no

Salary_amount	Profile_picture	username	email	password	role
300000.00	68275ae233f2e_IMG-20241110-WA0010.jpg	ram	nareshd2086@gmail.com	$2y$10$yE1wIMQ…	user

Created_at	balance
2025-05-16 21:03:54	1900.00

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 Front End

The Front End is the visual interface of the application that users interact with. It has been developed using HTML for structure, CSS for styling, and JavaScript for client-side interactivity. These technologies together ensure a clean, user-friendly, and responsive experience. Users can seamlessly navigate through pages such as login, registration, dashboard, deposit, withdrawal, and loan application forms. Form validation, input handling, and dynamic updates are powered by JavaScript to enhance usability. The design is kept simple, professional, and accessible, ensuring users of all skill levels can comfortably manage their banking needs.

3.2 Back End

The back end of the application is developed using PHP and MySQL, running locally on the XAMPP platform which integrates Apache and phpMyAdmin for seamless development and database management. PHP handles all server-side logic, acting as the backbone that processes user requests, manages sessions, and interacts with the database. It performs essential operations like user registration, login authentication, account management, deposits, withdrawals, transfers, and loan processing. All data is stored securely in a structured MySQL database, ensuring integrity and efficient retrieval. The backend logic also includes validation, error handling, and dynamic content rendering, providing users with real-time feedback and updates. This robust architecture ensures secure communication between the user interface and the database, enabling smooth, reliable, and secure functioning of the entire banking system.

CHAPTER 4

MODULE DESCRIPTION

4.1 The Sentinel of Security

The Sentinel of Security serves as the digital guardian of the system, overseeing user authentication and access control. It manages secure logins for both users and administrators, encrypted password storage using advanced algorithms, and efficient session handling. This module also includes reliable password recovery and reset options, ensuring users regain access without risking data safety. Its layered security approach defends against unauthorized access, protecting sensitive personal and financial information. By maintaining confidentiality, verifying credentials, and enabling uninterrupted access, the Sentinel upholds user trust and system reliability. Overall, it plays a vital role in strengthening the security framework of the banking application.

4.2 The Custodian of Clients

The Custodian of Clients manages all user-related operations with precision, allowing administrators to create, view, update, and search user accounts. It supports profile management, account freezing/unfreezing, and activity tracking to ensure compliance and transparency. Secure storage of user details, credentials, and transaction history enhances data reliability. Serving as the central hub for customer data, this module fosters organized and efficient interactions, strengthening client relationships within the banking ecosystem.

4.3 The River of Wealth

The River of Wealth powers the application’s financial operations, accurately managing deposits, withdrawals, and fund transfers with real-time balance updates. Each transaction is securely logged to ensure transparency, traceability, and integrity. Built-in validations and error-handling protect against fraud and data issues. Whether it's a simple deposit or a cross-account transfer, the process remains smooth and secure. This module reflects the trust, speed, and reliability essential to modern digital banking, streamlining every user’s financial journey.

4.4 The Architect of Aspirations

This module transforms financial dreams into reality by managing all loan-related activities. Users can apply for loans with details like amount, purpose, and duration, while administrators review applications for approval or rejection based on set policies. It records the full loan history, tracks repayment schedules, and calculates outstanding balances with accuracy. With real-time updates and transparency, it fosters trust and prevents discrepancies. By automating calculations and approvals, it reduces processing time and boosts user satisfaction. Whether for personal or business needs, this module ensures loans are accessible, accountable, and efficiently administered — becoming a pillar of opportunity within the system.

4.5 The Oracle of Insights

The Oracle of Insights brings clarity to complexity, offering a powerful dashboard of analytics and reports. This module collects, processes, and visualizes key data points, helping administrators understand the system's overall health. From user behavior trends to financial summaries, it offers dynamic charts, graphs, and reports that reveal real-time patterns. By tracking transactions, account activity, and loan distribution, it enables better decision-making and proactive issue detection. Whether assessing performance or identifying potential risks, this module is the strategic eye of the application. It transforms raw data into valuable intelligence, making it a vital tool for governance, optimization, and growth.

4.6 The Throne of Command

Commanding the entire architecture, the Throne of Command is the master control module for all administrative operations. It offers a centralized interface where admins can monitor system health, manage users and transactions, and respond to alerts instantly. This module provides a real-time overview through metrics, summaries, and actionable insights. With capabilities like quick access to logs, activity history, and control settings, it ensures the system stays aligned with security and performance goals. It empowers administrators with strategic oversight and the agility to maintain order and scalability. Like a vigilant king on the throne, it governs with clarity, authority, and precision.

CHAPTER 5

IMPLEMENTATION

5.1 SOURCE CODE

Register.php

```
<?php
session_start();
require_once '../database/config.php';
include '../header.php';
// Enable error reporting for PDO
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$errors = [];
$success = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$first_name = htmlspecialchars($_POST['first_name']);
$last_name = htmlspecialchars($_POST['last_name']);
$username = htmlspecialchars($_POST['username']);
$father_name = htmlspecialchars($_POST['father_name']);
$mother_name = htmlspecialchars($_POST['mother_name']);
$address = htmlspecialchars($_POST['address']);
$phone1 = htmlspecialchars($_POST['phone1']);
$phone2 = !empty($_POST['phone2']) ? htmlspecialchars($_POST['phone2']) : null;
$email = htmlspecialchars($_POST['email']);
$aadhaar_number = htmlspecialchars($_POST['aadhaar']);
$pancard_number = htmlspecialchars($_POST['pan']);
$voter_id = htmlspecialchars($_POST['voter_id']);
$age = htmlspecialchars($_POST['age']);
$dob = $_POST['dob'];
$gender = $_POST['gender'];
$differently_abled = $_POST['disabled'];
$working_details = htmlspecialchars($_POST['work_details']);
```

```
$salary_amount = !empty($_POST['monthly_salary']) ?
htmlspecialchars($_POST['monthly_salary']) : null;
$password = password_hash($_POST['password'], PASSWORD_DEFAULT);
// Handle profile picture
if (isset($_FILES['profile_picture']) && $_FILES['profile_picture']['error'] == 0) {
$img_name = uniqid() . "_" . basename($_FILES["profile_picture"]["name"]);
$target_path = "uploads/" . $img_name;
move_uploaded_file($_FILES["profile_picture"]["tmp_name"], $target_path);
} else {
$errors[] = "Profile picture is required.";
}
// Proceed only if no form errors
if (empty($errors)) {
try {
$stmt = $pdo->prepare("INSERT INTO users
(first_name, last_name, username, father_name, mother_name, address, phone1, phone2,
email,aadhaar_number, pancard_number, voter_id, age, dob, gender, differently_abled,
working_details, salary_amount,password, profile_picture, role)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 'user')");
$stmt->execute([
$first_name, $last_name, $username, $father_name, $mother_name, $address, $phone1,
$phone2, $email, $aadhaar_number, $pancard_number, $voter_id, $age, $dob, $gender,
$differently_abled, $working_details, $salary_amount, $password, $img_name ]);
// Successfully inserted
$last_id = $pdo->lastInsertId();
$_SESSION['user_id'] = $last_id;
header("Location: generate_account.php?id=$last_id");
exit;
} catch (PDOException $e) {
$errors[] = "Database error: " . $e->getMessage();
}}}
?>
```

User/login.php

```
<?php
// user/login.php
session_start();
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
require_once '../database/config.php';
$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$username = $_POST['username'] ?? '';
$password = $_POST['password'] ?? '';
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ? AND role = 'user'");
$stmt->execute([$username]);
$user = $stmt->fetch();
// Using MD5 to match your DB password encryption
//if ($user && md5($password) === $user['password']) {
if ($user && password_verify($password, $user['password'])) {
$_SESSION['user_id'] = $user['id'];
$_SESSION['role'] = 'user';
header("Location: dashboard.php");
exit;
} else {
$error = "Invalid credentials!";
}
}
?>
```

User/deposit_money.php

```
<?php
// user/deposit_money.php
session_start();
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'user') {
header("Location: ../main.php");
exit;
}
require_once '../database/config.php';
$message = "";
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$amount = floatval($_POST['amount']);
$description = "Self Deposit";
if ($amount > 0) {
$stmt = $pdo->prepare("UPDATE users SET balance = balance + ? WHERE id = ?");
$stmt->execute([$amount, $_SESSION['user_id']]);
$acc = $pdo->prepare("UPDATE accounts SET balance = balance + ? WHERE user_id =
?");
$acc->execute([$amount, $_SESSION['user_id']]);
$stmt = $pdo->prepare("INSERT INTO transaction (sender_id, receiver_id, amount,
description) VALUES (?, ?, ?, ?)");
$stmt->execute([$_SESSION['user_id'], $_SESSION['user_id'], $amount, $description]);
$message = "Successfully deposited ₹$amount to your account!";
} else {
$message = "Invalid amount entered!";
}
}
?>
```

User/withdraw_money.php

```
<?php
// user/withdraw_money.php
session_start();
if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'user') {
header("Location: ../main.php");
exit; }
require_once '../database/config.php';
$message = "";
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$amount = filter_input(INPUT_POST, 'amount', FILTER_VALIDATE_FLOAT);
if ($amount === false || $amount === null || $amount <= 0) {
throw new InvalidArgumentException("Invalid withdrawal amount"); }
$stmt = $pdo->prepare("SELECT balance FROM accounts WHERE user_id = ?");
$stmt->execute([$_SESSION['user_id']]);
$account = $stmt->fetch(PDO::FETCH_ASSOC);
if (!$account) {
throw new Exception("Account not found"); }
if ($amount > 0 && $amount >= 0) {
$stmt = $pdo->prepare("UPDATE users SET balance = balance - ? WHERE id = ?");
$stmt->execute([$amount, $_SESSION['user_id']]);
$acc = $pdo->prepare("UPDATE accounts SET balance = balance - ? WHERE user_id =
?");
$acc->execute([$amount, $_SESSION['user_id']]);
$stmt = $pdo->prepare("INSERT INTO transaction (sender_id, receiver_id, amount,
description) VALUES (?, NULL, ?, ?)");
$stmt->execute([$_SESSION['user_id'], $amount, "Cash Withdrawal"]);
$message = "Successfully withdrew ₹$amount from your account!";
} else {
$message = "Invalid amount entered!";
}}
?>
```

User/view_balance.php

```
<?php
session_start();
require_once '../database/config.php';
include '../header.php';
// Check if user is logged in
if (!isset($_SESSION['user_id'])) {
header('Location: ../login.php');
exit;
}
// Fetch user balance
$stmt = $pdo->prepare("SELECT account_number, balance FROM accounts WHERE user_id = ?");
$stmt->execute([$_SESSION['user_id']]);
$account = $stmt->fetch();
if (!$account) {
die("Account not found.");
}
?>
```

User/apply_loan.php

```
<?php
// user/apply_loan.php
session_start();
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'user') {
header("Location: ../main.php");
exit;
}
include '../header.php';
```

```
require_once '../database/config.php';
$message = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$amount = $_POST['amount'];
$duration = $_POST['duration'];
if ($amount > 0 && $duration > 0) {
$stmt = $pdo->prepare("INSERT INTO loans (user_id, amount, duration_in_months)
VALUES (?, ?, ?)");
$stmt->execute([$_SESSION['user_id'], $amount, $duration]);
$message = " 🎉 Loan application submitted successfully!";
} else {
$message = " ❗ Please enter valid loan details.";
}
}
?>
```

Admin/login.php

```
<?php
// admin/login.php
session_start();
header("Cache-Control: no-store, no-cache, must-revalidate, max-age=0");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
require_once '../database/config.php';
$error = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$username = $_POST['username'] ?? '';
$password = $_POST['password'] ?? '';
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ? AND role =
'admin'");
$stmt->execute([$username]);
$admin = $stmt->fetch();
```

```
if ($admin && password_verify($password, $admin['password'])) {
$_SESSION['user_id'] = $admin['id'];
$_SESSION['role'] = 'admin';
header("Location: dashboard.php");
exit;
} else {
$error = "Invalid credentials!";
}
}
?>
```

Admin/dashboard.php

```
<?php
session_start();
if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'admin') {
header("Location: ../main.php");
exit;
}
include '../header.php';
require_once '../database/config.php';
// Fetch total number of users
$stmt = $pdo->query("SELECT COUNT(*) as total_users FROM users WHERE role = 'user'");
$total_users = $stmt->fetch(PDO::FETCH_ASSOC)['total_users'];
// Fetch total balance across all accounts
$stmt = $pdo->query("SELECT SUM(balance) as total_balance FROM accounts");
$total_balance = $stmt->fetch(PDO::FETCH_ASSOC)['total_balance'] ?? 0;
// Fetch today's transactions count
$stmt = $pdo->prepare("SELECT COUNT(*) as today_transactions FROM transaction WHERE DATE(created_at) = CURDATE()");
$stmt->execute();
$today_transactions = $stmt->fetch(PDO::FETCH_ASSOC)['today_transactions'];
```

```
// Fetch active loans count
$stmt = $pdo->prepare("SELECT COUNT(*) as active_loans FROM loans WHERE status
= 'Approved'");
$stmt->execute();
$active_loans = $stmt->fetch(PDO::FETCH_ASSOC)['active_loans'];
// Function to format currency
function formatCurrency($amount) {
if ($amount >= 10000000) { // Crores
return '₹' . number_format($amount / 10000000, 2) . ' Cr';
} elseif ($amount >= 100000) { // Lakhs
return '₹' . number_format($amount / 100000, 2) . ' L';
} else {
return '₹' . number_format($amount, 2);
}
}
?>
```

Admin/adjust_balance.php

```
<?php
// admin/adjust_balance.php
session_start();
require_once '../database/config.php';
include '../header.php';
$message = '';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$search = $_POST['search'] ?? '';
$type = $_POST['type'] ?? '';
$amount = floatval($_POST['amount']);
// Get user by ID or Account Number
$stmt = $pdo->prepare("SELECT users.id, users.first_name, accounts.id AS acc_id,
accounts.account_number, accounts.balance
FROM users
```

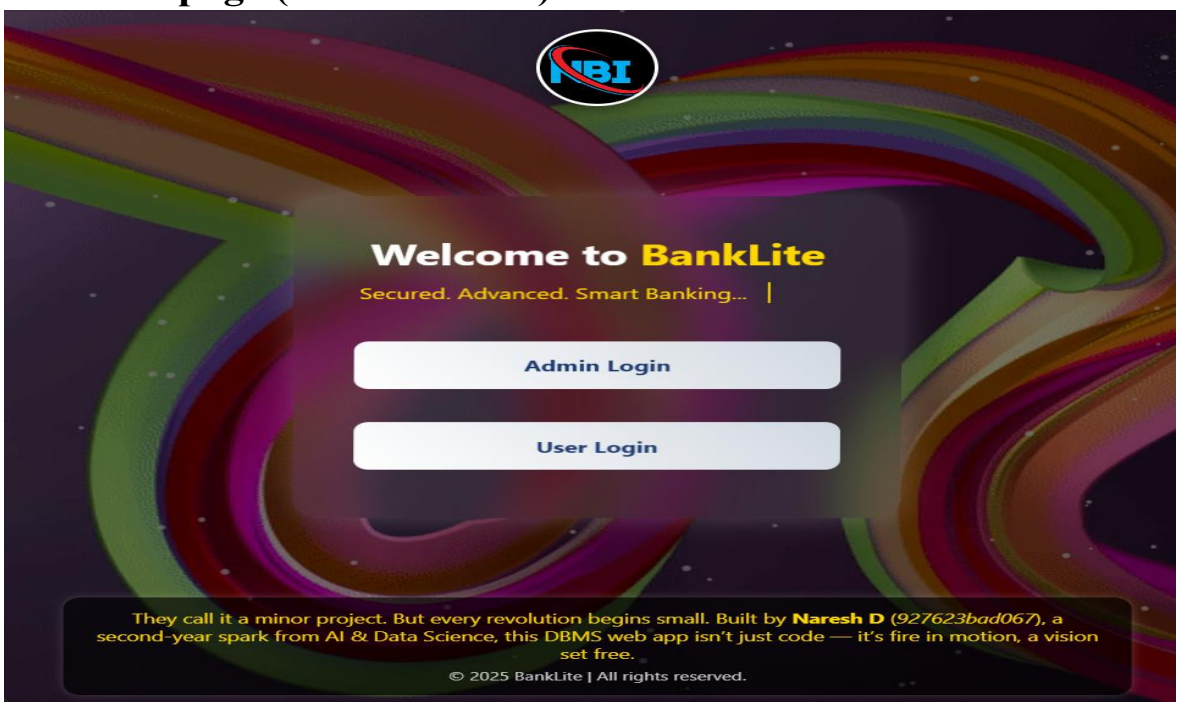
```
JOIN accounts ON users.id = accounts.user_id
WHERE users.id = ? OR accounts.account_number = ?");
$stmt->execute([$search, $search]);
$user = $stmt->fetch();
if ($user && $amount > 0 && in_array($type, ['deposit', 'withdraw'])) {
$new_balance = $type === 'deposit'
? $user['balance'] + $amount
: $user['balance'] - $amount;
if ($type === 'withdraw' && $amount > $user['balance']) {
$message = "❌ Withdrawal failed: Insufficient funds.";
} else {
// Update balance
$pdo->prepare("UPDATE accounts SET balance = ? WHERE id = ?")
->execute([$new_balance, $user['acc_id']]);
// Log transaction (remove 'type' column)
$pdo->prepare("INSERT INTO transaction (sender_id, receiver_id, amount, description)
VALUES (?, ?, ?, ?)")
->execute([
$type === 'deposit' ? null : $user['id'],
$type === 'deposit' ? $user['id'] : null,
$amount,
'Manual ' . ucfirst($type) . ' by Admin'
]);
$message = "☑ Successfully {$type}ed ₹$amount for user {$user['first_name']}
(Account: {$user['account_number']})";
}
} else {
$message = "⚠ Invalid input or user not found.";
}
}
?>
```

Admin/approve_loans.php

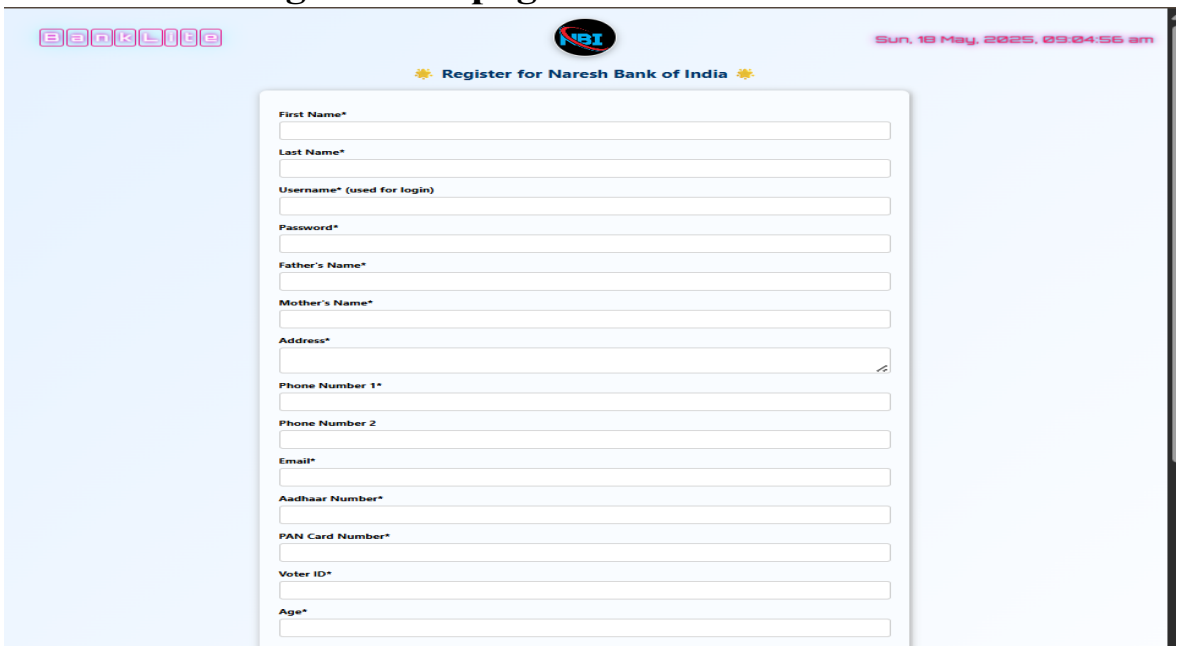
```
<?php
session_start();
if (!isset($_SESSION['user_id']) || $_SESSION['role'] !== 'admin') {
header("Location: ../main.php");
exit;
}
include '../header.php';
require_once '../database/config.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$loan_id = $_POST['loan_id'];
$action = $_POST['action'];
$interest = $_POST['interest_rate'];
if ($action === "Approve") {
$stmt = $pdo->prepare("UPDATE loans SET status='Approved', interest_rate=?, approved_by=? WHERE id=?");
$stmt->execute([$interest, $_SESSION['user_id'], $loan_id]);
} elseif ($action === "Reject") {
$stmt = $pdo->prepare("UPDATE loans SET status='Rejected', approved_by=? WHERE id=?");
$stmt->execute([$_SESSION['user_id'], $loan_id]);
}
}
$pendingLoans = $pdo->query("SELECT loans.*, users.username FROM loans JOIN users ON loans.user_id = users.id WHERE loans.status='Pending'")->fetchAll();
?>
```

CHAPTER 6
SNAPSHOTS

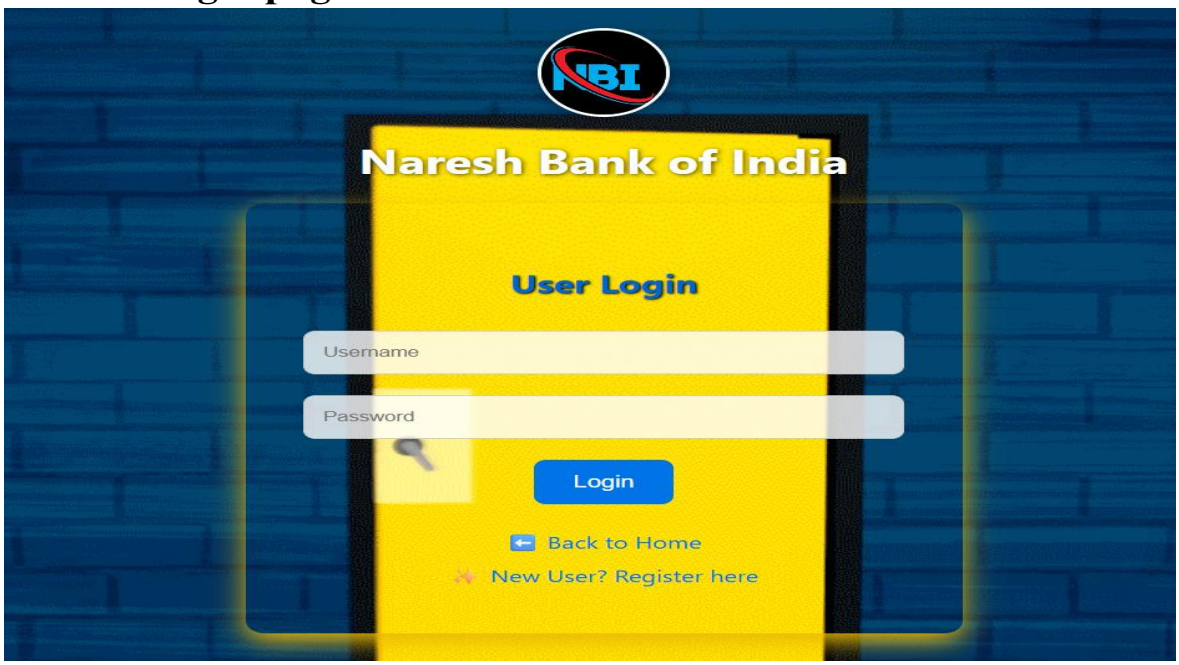
6.1 Main page (Role selection)



6.2 New user registration page

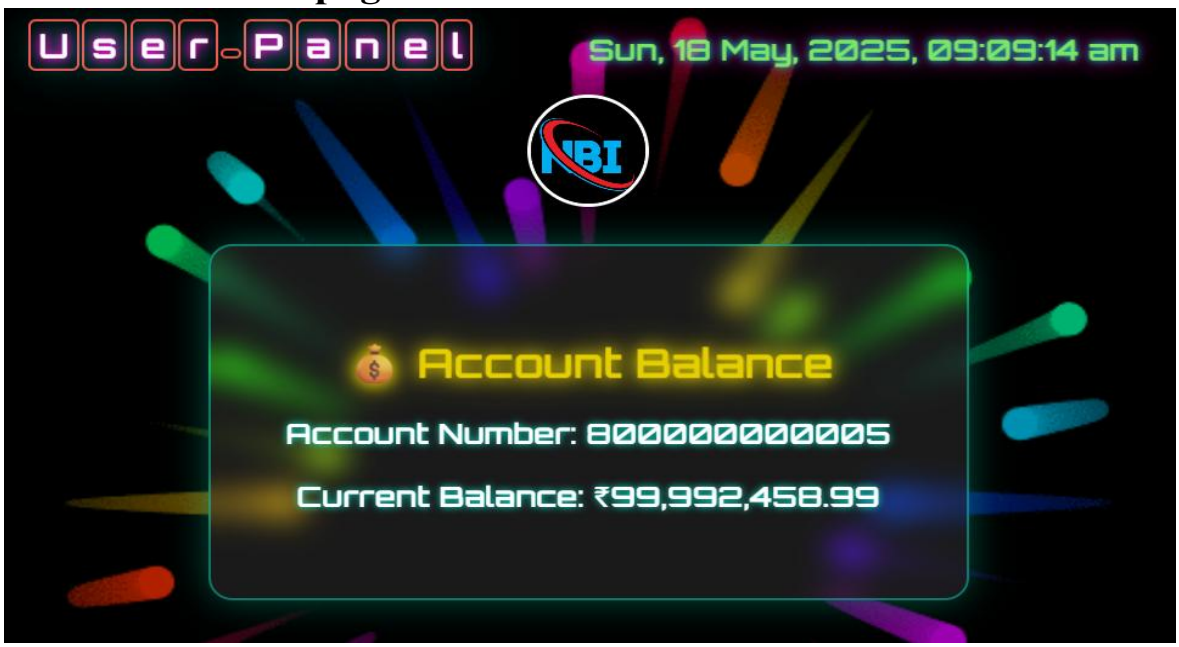


6.3 User login page

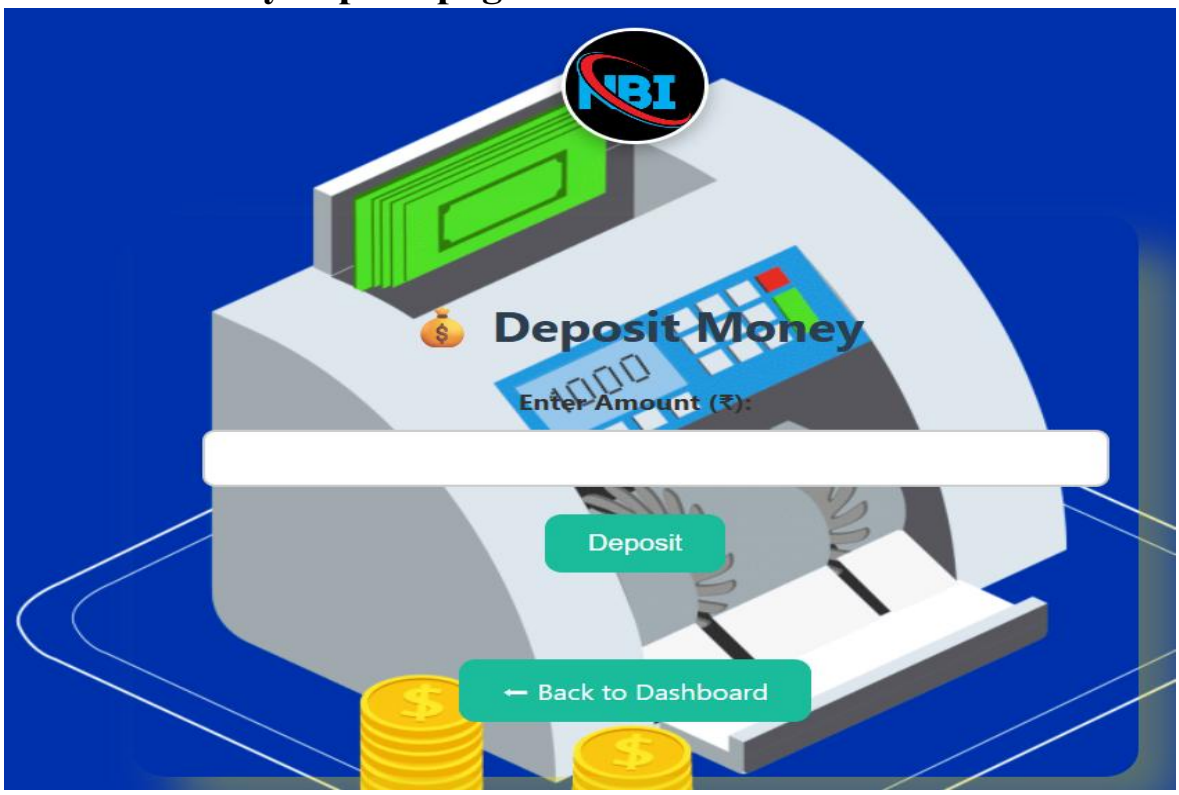


6.4 User dashboard

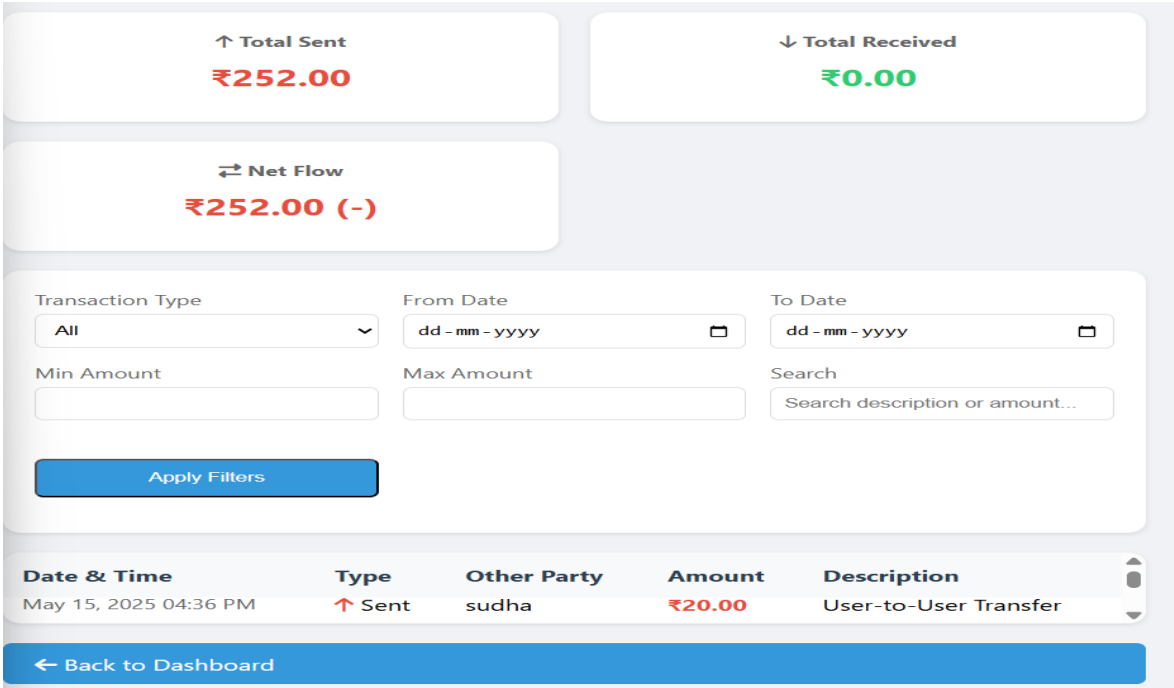
6.5 User balance page



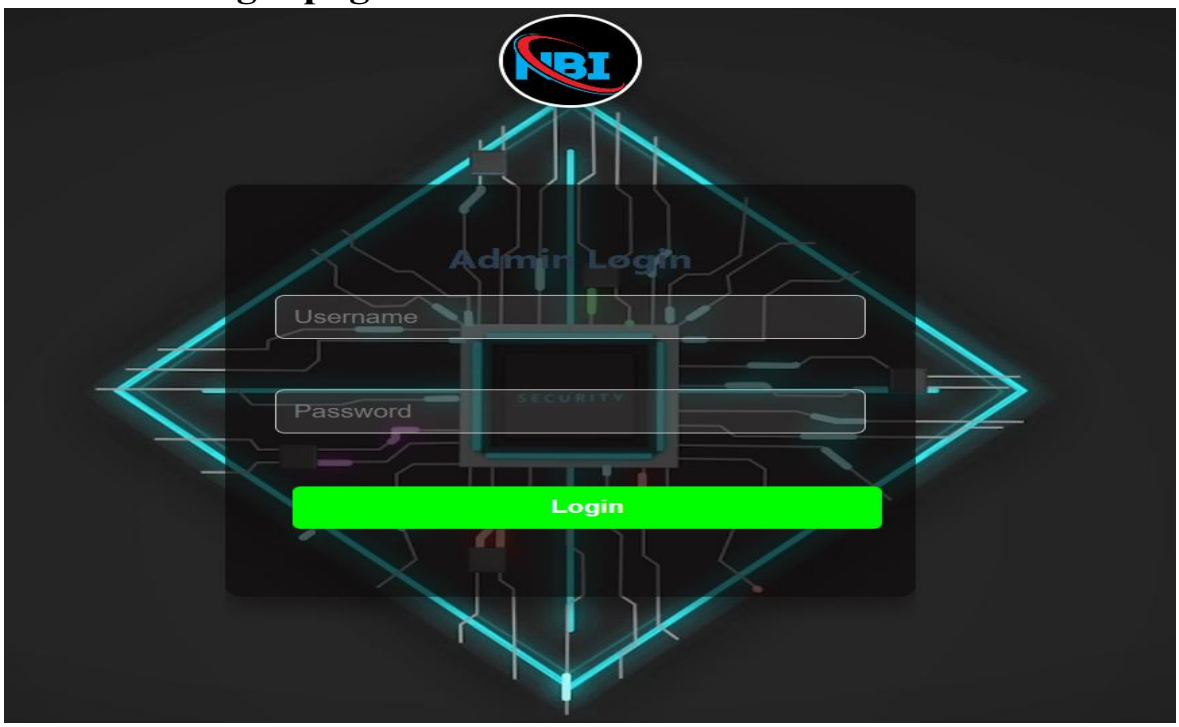
6.6 User money deposit page



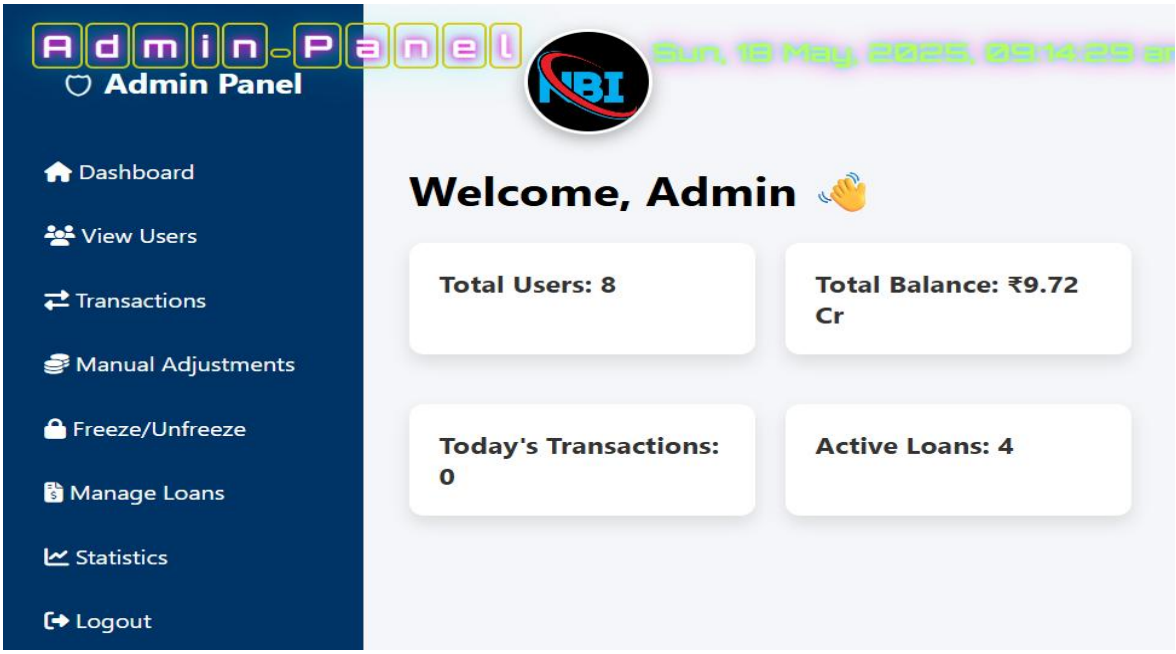
6.7 User transactions page



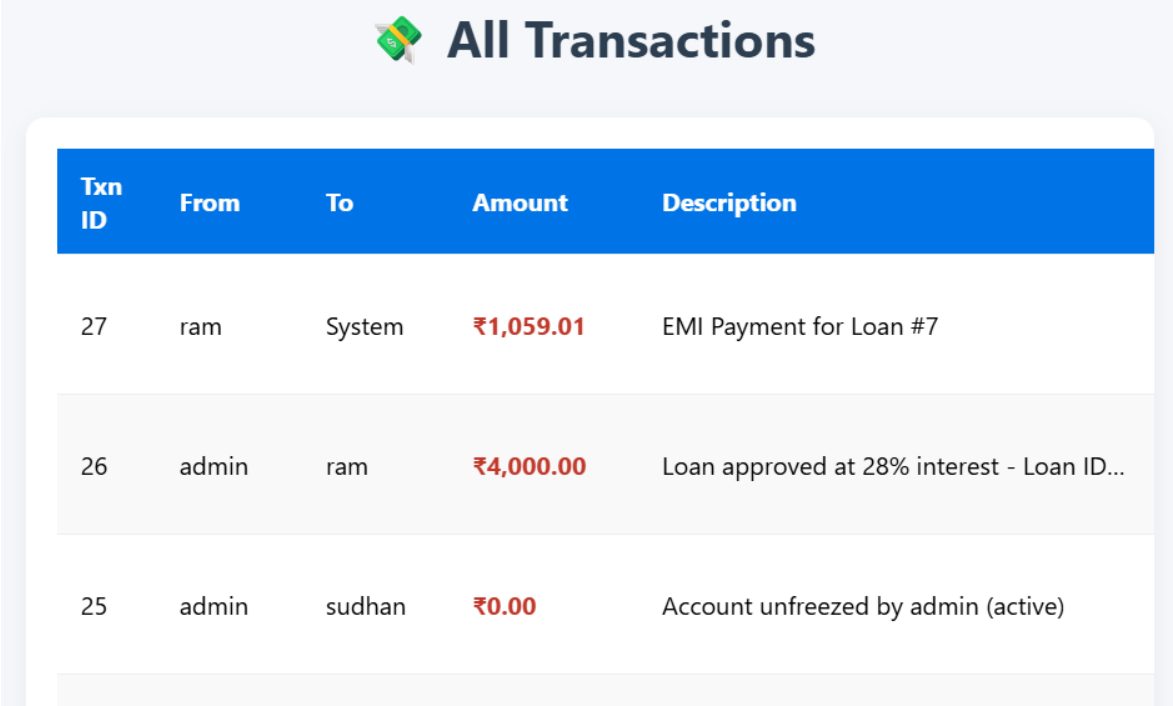
6.8 Admin login page



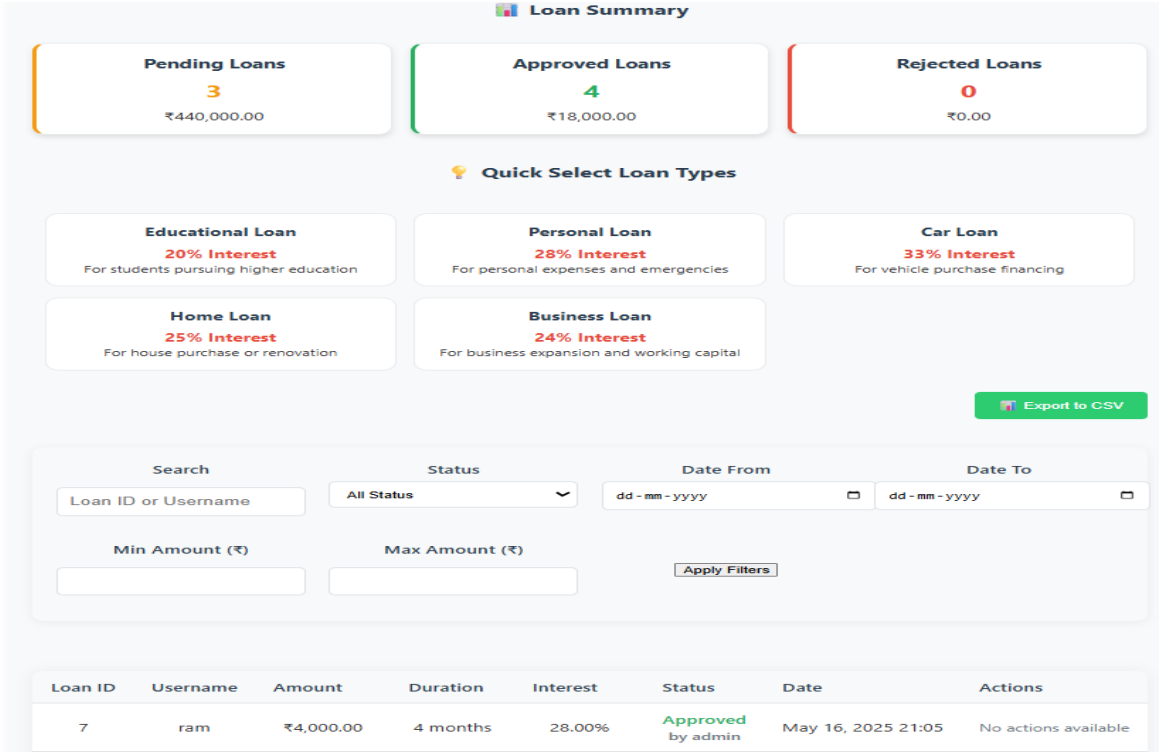
6.9 Admin Dashboard



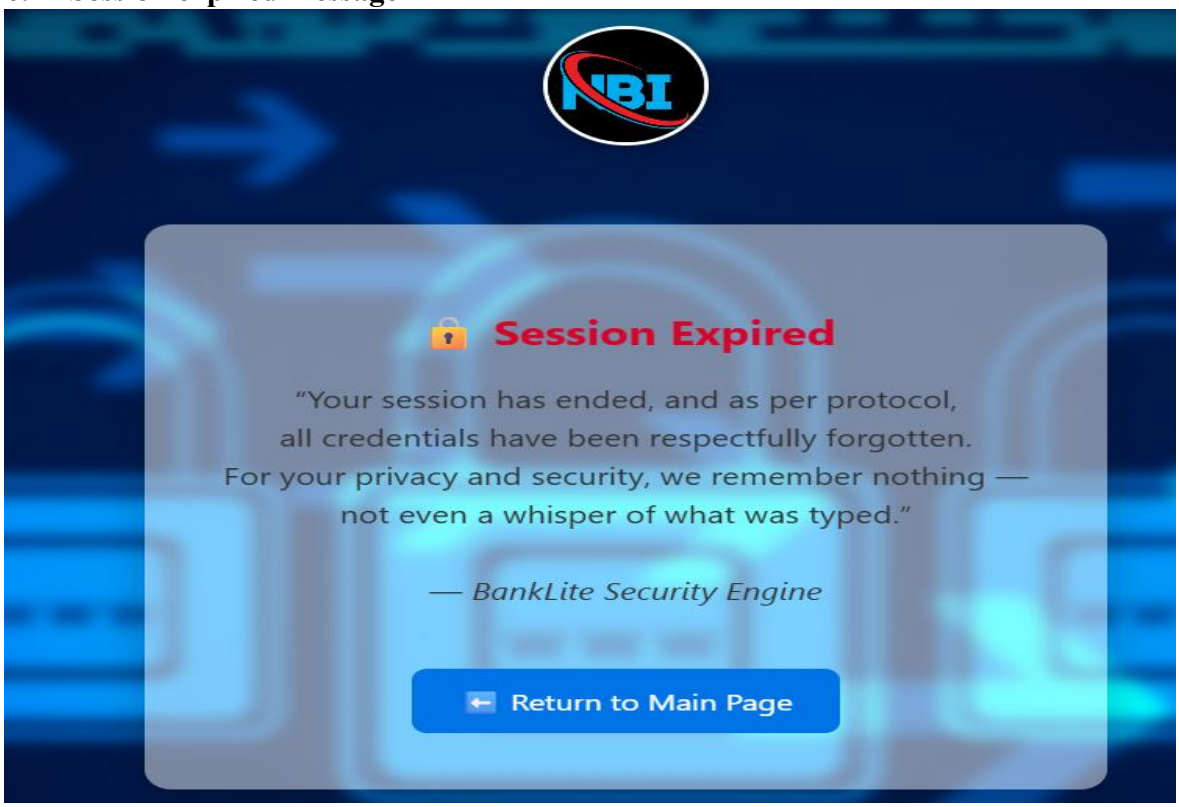
6.10 Admin all transactions page



6.11 Admin loan management page



6.12 Session expired message



CONCLUSION

Simple Banking Application, serves as a powerful testament to the fusion of technology and trust. It was developed with the vision of bringing essential banking services—such as account creation, deposits, withdrawals, fund transfers, and loan management—into a digital, accessible, and secure environment. The application caters to both users and administrators, streamlining operations through intuitive workflows, clean interface design, and a well-structured backend powered by PHP, MySQL, and XAMPP.

Each module is built with clarity and purpose: from the Sentinel of Security, safeguarding user data, to the River of Wealth, managing financial transactions with precision, and the Architect of Aspirations, supporting the vision of responsible lending. The Oracle of Insights empowers administrators with analytics, while the Throne of Command centralizes system control, ensuring order and scalability.

What makes this project unique is not just its functionality but its emphasis on user experience, modularity, and security. It reflects real-world banking models and simulates how users interact with modern financial systems.

As we stand at the crossroads of digital transformation, this project exemplifies how even simple web technologies, when orchestrated thoughtfully, can create powerful, user-friendly applications. It is more than a system—it is a stepping stone toward smarter, faster, and more secure financial solutions for tomorrow.

REFERENCES

1. **Silberschatz, A., Korth, H. F., & Sudarshan, S.** *Database System Concepts (7th Edition)* McGraw-Hill Education, 2020.
2. **Welling, L., & Thomson, L.** *PHP and MySQL Web Development (5th Edition)* Addison-Wesley, 2016.
3. **MDN Web Docs** https://developer.mozilla.org/