Banking Management System

G. Bal Gopal (6021B032)

Sourav Das (6021B020)

Deepankar Singha (6021B018)

Kiran Kumar Murmu (6021B023)

Department Of Computer Science
Bhadrak Autonomous College, Bhadrak

Certificate

This is to certify that the project entitled “Banking Management System Using PHP” has been submitted to department of Computer Science, Bhadrak Autonomous College, Bhadrak for the fulfillment of the requirement for the award of the degree of Bachelor in Science in “Computer Science” by following students of final year.

G. Bal Gopal (6021B032)

Sourav Das (6021B020)

Deepankar Singha (6021B018)

Kiran Kumar Murmu (6021B023)

Signature of External **Signature of internal**

Declaration

We hereby declare that the project report entitled “Banking management System “ being submitted by us towards the partial fulfillment of the degree of Bsc Computer science, for the department of Computer Science is a project work carried by us under the supervision of Shri Krutikanta Bhattacharya , and has not been submitted anywhere else.

Acknowledgement

We feel to acknowledge our indebtedness and deep sense of gratitude to our guide Shri Krutikanta Bhattacharya for providing us with continuous support. He has always been there to listen to our queries, give advice and has prudently guided us all throughout this project.

We owe a deep sense of gratitude to Mr Krutikanta Bhattacharya for educating us with the language used in this project and for always being there to extend a helping hand.

We would like to convey our sincere thanks to all the members and staff of our department for the generous help and cooperation throughout our study period.

Lastly, we would like to thank our friends, who helped us to gather different information collecting data and despite their busy schedules they gave us different ideas in making this project unique.

Thank you.

G. Bal Gopal

Kiran Kumar Murmu

Sourav Das

Deepankar Singha

Abstract

This report serves as the groundwork for our final project in the BSc in Computer Science program. The project, titled "Bank Management System: Developing an Integrated Banking Solution," is under the supervision of Mr. Kabiratna Behera. The primary objective of this report is to conduct a comprehensive analysis of the banking sector, focusing on the opportunities and prerequisites for constructing a successful bank management system.

To accomplish this, we initially conducted an extensive review of relevant literature and related work within the realm of banking and financial technology. This encompassed not only the technical aspects but also the business and social dimensions of modern banking practices. Given the significant impact of banking technologies on global commerce, it was imperative to understand its broader context. The literature review aimed to extrapolate insights that would inform our project's strategic focus.

We delved into the success and failure factors within the banking sector, examining case studies of institutions that effectively implemented technological solutions and those that encountered challenges. Additionally, we compared various established methods for developing banking systems. This comparative analysis highlighted the strengths and weaknesses of each approach, enabling us to leverage their advantages and circumvent potential pitfalls in our chosen methodology.

The subsequent section of the report outlines the methodology to be employed throughout the project. It delineates the project's overarching objectives and goals, accompanied by a detailed project plan. Furthermore, we conducted research on key aspects of the development methodology, such as system evaluation, market analysis, and suitable business models for the banking sector.

Our bank management system is designed to ensure robust security and data integrity. Leveraging PHP and MySQL database technologies, we mitigate the risk of data loss through regular backups and employ encryption mechanisms to safeguard sensitive information.

Additionally, the use of electronic databases minimizes code complexity, enhancing system reliability and performance.

Overall, this report lays the foundation for the development of a sophisticated bank management system that meets the evolving needs of the financial industry while prioritizing security, reliability, and efficiency.

Overview

Chapter 1: Introduction

This chapter outlines the objectives, purpose, and overview of the banking web application project. It highlights the significance of the project in addressing the needs of users within the banking sector and provides a comprehensive overview of its scope and functionalities.

Chapter 2: Feasibility Analysis

In this chapter, we assess the technical and economic feasibility of the banking web application project. We determine whether the project is technically viable for development and operation, considering factors such as technological requirements and operational feasibility. Additionally, we analyze the economic feasibility of the project, evaluating its potential return on investment and cost-effectiveness.

Chapter 3: Software Engineering Process

This chapter discusses the software engineering paradigm adopted for the development of the banking web application. We outline the software life cycle, encompassing activities such as requirements definition, development, testing, deployment, and maintenance. By following a structured approach, we ensure the systematic and efficient development of the software product.

Chapter 4: Requirements Analysis

In this chapter, we identify and analyze the requirements of the banking web application. We outline the hardware and software prerequisites necessary for the project's implementation. Furthermore, we specify the functional requirements essential for the banking web application to meet user needs effectively.

Chapter 5: System Design

This chapter elaborates on the design of the banking web application system. We present the Data Flow Diagram (DFD), illustrating the flow of data through various modules of the web application. Additionally, we document any assumptions made during the system design phase to ensure its functionality and effectiveness.

Chapter 6: Implementation and Coding

Here, we provide insights into the coding process and offer a sneak preview of the banking web application's user interface. We include screenshots showcasing the execution of the project, demonstrating its functionality and visual appeal.

Chapter 7: Testing Methodologies

This chapter discusses the testing methodologies employed for the banking web application. We detail the unit testing procedures carried out to verify the functionality of individual components. Additionally, we describe the integration testing process and provide an overview of the total system simulation to ensure its reliability and robustness.

Chapter 8: Future Scope

In this concluding chapter, we explore the potential for further improvement and enhancement of the banking web application. We identify creative and feasible avenues for future development, highlighting opportunities to expand functionality, improve user experience, and address evolving industry requirements.

Index

3. Methodology

4. System Requirements

8. Future Enhancements and Recommendations

8.1 Introduction

8.2 Scope for Further Improvement

8.3 Recommendations

9. References

1. Introduction

1.1 Introduction:

The proliferation of online banking management systems represents a pivotal shift in the financial services sector, ushering in an era of unparalleled convenience, efficiency, and security for both customers and financial institutions. As the digital landscape continues to evolve at an exponential pace, the design and implementation of a robust online banking platform necessitate meticulous planning, strategic foresight, and cutting-edge technologies. This introduction serves as a foundational framework for the development of a sophisticated online banking management system, highlighting the multifaceted considerations and intricate intricacies involved in its creation.

1.2 Background of the Study:

The evolution of online banking is a testament to the relentless march of technological progress, tracing its origins from the nascent days of internet banking to the sophisticated digital ecosystems of today. This section delves deep into the historical antecedents of online banking, exploring the catalyzing factors, pivotal milestones, and transformative impacts that have shaped its trajectory over time. From the advent of Automated Teller Machines (ATMs) to the emergence of mobile banking applications and blockchain technology, the journey of online banking is characterized by a continuous quest for innovation, efficiency, and customer-centricity.

Furthermore, this section elucidates the seismic shifts that online banking has engendered within the broader financial landscape, from the democratization of financial services to the disruption of traditional banking models. By elucidating the evolutionary journey of online banking, this section underscores the imperative for financial institutions to adapt and innovate in response to the changing needs and preferences of digital-native consumers.

1.3 Statement of the Problem:

Amidst the backdrop of rapid technological advancement and shifting consumer preferences, traditional banking methods find themselves increasingly challenged to keep pace with the demands of the digital age. The COVID-19 pandemic, in particular, has served as a catalyst for change, underscoring the vulnerabilities of brick-and-mortar banking infrastructure and accelerating the imperative for digital transformation.

This section articulates the multifaceted challenges confronting traditional banking models, ranging from the inefficiencies of manual processes to the vulnerabilities of legacy systems. Moreover, it highlights the pressing need for financial institutions to embrace digitalization and develop robust online banking management systems that can deliver seamless, secure, and personalized banking experiences across digital channels.

1.4 The Solution:

In response to the shortcomings of traditional banking methods, the proposed online banking management system represents a holistic solution designed to address the evolving needs and expectations of modern banking customers. By harnessing the power of advanced technologies such as artificial intelligence, machine learning, and biometric authentication, the system aims to deliver a frictionless, intuitive, and secure banking experience that transcends the constraints of time and space.

Moreover, the system prioritizes interoperability, scalability, and regulatory compliance, enabling financial institutions to adapt and innovate in a rapidly evolving digital landscape. By fostering greater connectivity, transparency, and trust between banks and their customers, the online banking management system seeks to redefine the contours of financial services in the digital age.

1.5 Aims and Objectives:

The overarching aim of this project is to conceptualize, design, and implement a state-of-the-art online banking management system that embodies the principles of innovation, inclusivity, and integrity. Specific objectives include:

- Enhancing Accessibility: To develop a user-friendly interface that caters to the diverse needs and preferences of banking customers, including individuals with disabilities or limited digital literacy.

- Improving User Experience: To streamline banking processes, reduce friction points, and enhance the overall user experience through intuitive design, personalized recommendations, and proactive customer support.

- Optimizing Operational Efficiency: To automate routine tasks, streamline backend processes, and leverage data analytics to drive operational efficiency and cost savings for financial institutions.

- Ensuring Regulatory Compliance: To adhere to stringent regulatory requirements, safeguard customer data, and mitigate risks associated with cybersecurity threats, fraud, and money laundering.

By achieving these objectives, the online banking management system aims to empower financial institutions to thrive in a digital-first world while delivering tangible value and convenience to customers.

1.6 Scope and Limitation:

The scope of the online banking management system encompasses a wide range of functionalities, including account management, funds transfer, bill payments, loan applications, and financial reporting. However, certain limitations must be acknowledged and addressed, including:

- Regulatory Compliance: Ensuring compliance with evolving regulatory frameworks, such as Know Your Customer (KYC) regulations, Anti-Money Laundering (AML) laws, and General Data Protection Regulation (GDPR) requirements.

- Security Concerns: Mitigating risks associated with cybersecurity threats, data breaches, identity theft, and fraudulent activities through robust security protocols, encryption algorithms, and multi-factor authentication mechanisms.

- Technological Dependencies: Navigating the complexities of integrating disparate systems, legacy applications, and third-party services while maintaining compatibility, reliability, and scalability.

- Customer Support: Providing comprehensive customer support services, including troubleshooting assistance, account inquiries, and dispute resolution, to ensure a seamless and satisfactory banking experience for users.

While these limitations pose significant challenges, they also present opportunities for innovation, collaboration, and continuous improvement in the development and deployment of the online banking management system.

1.7 Research Methodology:

The development of the online banking management system will adhere to a structured research methodology comprising several key phases, including:

- Feasibility Study: Conducting a comprehensive analysis of the technical, operational, and economic feasibility of the project, including market research, competitor analysis, and cost-benefit analysis.

- Requirement Gathering: Collaborating with stakeholders, including banks, regulatory authorities, and end-users, to identify and prioritize functional and non-functional requirements for the system.

- System Analysis: Analyzing the current state of banking operations, identifying pain points and bottlenecks, and defining system specifications, use cases, and user stories to guide the development process.

- Design and Development: Creating high-fidelity wireframes, mockups, and prototypes to visualize the user interface and user experience, and leveraging agile development methodologies to iteratively build and refine the system.

- Testing and Quality Assurance: Conducting rigorous testing, including unit testing, integration testing, and user acceptance testing, to validate the functionality, performance, and security of the system.

- Deployment and Maintenance: Deploying the online banking management system in a production environment, monitoring its performance and reliability, and providing ongoing maintenance and support to address issues and implement enhancements.

Throughout the research process, data will be collected from various sources, including academic literature, industry reports, market surveys, and expert interviews, to inform decision-making and ensure alignment with industry best practices and emerging trends.

1.8 Significance of the Study:

The significance of this study lies in its potential to catalyze transformative change within the banking industry, driving innovation, fostering financial inclusion, and enhancing customer satisfaction. By developing a robust online banking management system, financial institutions can unlock new opportunities for growth, differentiation, and value creation, while empowering customers with greater control, flexibility, and convenience over their financial lives.

Furthermore, the study holds broader implications for the broader ecosystem of digital finance, including fintech startups, regulatory authorities, and policymakers, by providing

insights into emerging technologies, regulatory trends, and consumer preferences shaping the future of banking.

1.9 Definition of Terms:

To ensure clarity and understanding, key terms and concepts relevant to online banking management, e-commerce, financial technology, and regulatory compliance will be defined and elucidated throughout the study. This includes terms such as Automated Clearing House (ACH), Application Programming Interface (API), Cryptocurrency, Digital Wallet, Peer-to-Peer (P2P) lending, and more.

1.10 Chapter Layout:

The subsequent chapters will delve into specific aspects of the online banking management system, including literature review, methodology, system analysis, design, implementation, testing, results, discussion, conclusion, and recommendations. Each chapter will contribute to a comprehensive understanding of the project's objectives, methodologies, findings, and implications for the banking industry.

2. Literature Review

2.1 Introduction:

The introduction to the literature review sets the stage for understanding the significance of online banking management systems in today's financial landscape. It underscores the transformative impact of digital technologies on banking practices, emphasizing the need for sophisticated online banking platforms to meet evolving customer expectations and industry standards. By providing context and framing the discussion, this section highlights the relevance and urgency of exploring existing literature in this domain.

2.2 Background of the Study:

The background section delves into the historical evolution of online banking, tracing its origins from early experiments with internet-based financial services to the sophisticated digital ecosystems prevalent today. It explores key milestones, technological advancements, and regulatory developments that have shaped the trajectory of online banking over time. By contextualizing the evolution of online banking within broader historical and technological trends, this section provides valuable insights into its current state and future prospects.

2.3 Status of Online Banking in the Current Business Environment:

This subsection provides a comprehensive overview of the current landscape of online banking, drawing on recent research, industry reports, and market trends. It examines the adoption rates, usage patterns, and emerging technologies driving the growth of online banking services. By synthesizing empirical evidence and expert analysis, this section offers valuable insights into the status quo of online banking and its implications for financial institutions and consumers alike.

2.4 Importance of Online Banking:

Here, the literature review explores the importance of online banking in transforming financial services and enhancing customer experiences. It elucidates the myriad benefits of online banking, including convenience, accessibility, and cost-effectiveness. Through a synthesis of scholarly research and industry best practices, this section highlights the critical role of online banking in driving innovation, fostering financial inclusion, and improving overall efficiency in the banking sector.

2.5 Problems in Online Banking:

This subsection critically examines the challenges and drawbacks associated with online banking, ranging from cybersecurity threats to usability issues and regulatory complexities. It identifies common pain points faced by both financial institutions and consumers, such as data breaches, phishing scams, and compliance burdens. By acknowledging these challenges, this section underscores the importance of proactive measures and robust risk management strategies to mitigate the risks associated with online banking.

2.6 Factors Affecting Online Banking:

Here, the literature review analyzes the various factors influencing the adoption and success of online banking initiatives. It explores technological, regulatory, economic, and social factors shaping the online banking landscape, highlighting the drivers and barriers to adoption. Through a synthesis of empirical research and theoretical frameworks, this section provides valuable insights into the complex interplay of factors shaping the future of online banking and its implications for financial institutions and policymakers.

3. Methodology

3.1 Methodology:

This subsection outlines the systematic approach employed in the development of the online banking management system, ensuring a structured and efficient process from inception to implementation. It emphasizes the importance of meticulous planning, strategic decision-making, and stakeholder collaboration in achieving project objectives. By delineating the methodology, this section provides a roadmap for the project team to navigate through various stages effectively.

3.2 Introduction:

The introduction to the methodology section contextualizes the importance of adopting a robust methodology for the development of the online banking management system. It highlights the significance of systematic planning, rigorous analysis, and iterative design in ensuring the success of complex software projects. By setting the stage for the subsequent discussions, this section underscores the critical role of methodology in achieving project goals and delivering value to stakeholders.

3.3 System Analysis:

This subsection focuses on the initial phase of system analysis, which involves assessing the feasibility, gathering requirements, and defining system specifications. It discusses the methods and techniques employed to understand stakeholder needs, identify functional and non-functional requirements, and establish the foundation for system design. Through a structured analysis process, this section aims to ensure alignment between project objectives and user expectations.

3.3.1 Feasibility Study:

Here, the feasibility study evaluates the technical, operational, and economic feasibility of the online banking management system. It assesses factors such as resource availability, technology readiness, and market demand to determine the viability of the project. By conducting a comprehensive feasibility analysis, this section informs decision-making and risk management strategies for the project.

3.3.2 Requirement Gathering:

This stage focuses on gathering requirements from stakeholders, including banks, regulatory authorities, and end-users. It employs various techniques such as interviews, surveys, and workshops to elicit comprehensive and actionable requirements. By engaging stakeholders early in the process, this section ensures that the online banking management system meets the diverse needs and preferences of its users.

3.3.3 Data Collection Methods:

In this subsection, the methods for collecting relevant data for system analysis are discussed. It emphasizes the importance of using both primary and secondary research methods to gather comprehensive insights into user behaviors, market trends, and technological advancements. By leveraging multiple data sources, this section ensures the robustness and validity of the analysis.

3.3.4 Requirements:

Here, the requirements elicited from stakeholders are documented and prioritized based on their importance and feasibility. This section defines functional and non-functional

requirements, use cases, and user stories to guide the design and development of the online banking management system. By establishing clear and measurable requirements, this section lays the foundation for a successful project outcome.

ID	Requirement Description	Priority	Type
R1	User authentication	High	Functional
R2	Account creation	High	Functional
R3	Deposit funds	High	Functional
R4	Withdraw funds	High	Functional
R5	Transfer funds between accounts	High	Functional
R6	View account balance	High	Functional
R7	View transaction history	Medium	Functional
R8	Apply for loans	Medium	Functional
R9	Manage recurring payments	Medium	Functional
R10	Generate account statements	Medium	Functional
R11	Data encryption	High	Non-Functional
R12	Scalability	High	Non-Functional
R13	Performance	High	Non-Functional
R14	Reliability	High	Non-Functional
R15	Compliance with banking regulations	High	Non-Functional

3.4 Input Specification:

This subsection outlines the specifications for inputs to the online banking management system, including data formats, validation rules, and user inputs. It ensures compatibility and consistency in system inputs, facilitating seamless data processing and user interactions. By defining input specifications, this section enhances the usability and reliability of the system.

Input	Description
Customer Information	Data provided by customers during account registration or updates, including name, address, contact information, identification documents, and tax ID.
Account Details	Information related to bank accounts, such as account type, account number, opening date, balance, interest rate, and associated customer IDs.
Transactions	Data representing financial transactions initiated by customers, including deposits, withdrawals, fund transfers, bill payments, and loan applications.
Loan Applications	Information provided by customers when applying for loans, including loan type, amount requested, repayment term, employment details, and income sources.
Payment Instructions	Instructions for bill payments, fund transfers, or loan repayments, including recipient details, payment amount, payment date, and reference information.
User Credentials	Login credentials provided by users for authentication purposes, typically including username/email and password.
Compliance Documentation	Documentation required for regulatory compliance, such as KYC (Know Your Customer) documents, AML (Anti-Money Laundering) information, and tax compliance forms.
System Configuration Data	Configuration settings for the banking management system, including system parameters, user roles, access permissions, and security policies.
External Data Feeds	External data sources providing market data, exchange rates, interest rates, and other financial information used for account management and decision-making.
Error and Exception Data	Information related to system errors, exceptions, or warnings, including error codes, error messages, timestamps, and stack traces.

3.5 Output Specification:

Here, the specifications for outputs generated by the online banking management system are defined, including reports, alerts, and user interfaces. It ensures that outputs meet user expectations and regulatory requirements, enhancing the overall user experience. By specifying output requirements, this section enables effective communication and decision-making within the system.

Output	Description
Account Statement	A detailed report showing all transactions, deposits, withdrawals, and current balances for a specific account over a defined period.
Transaction History	A chronological list of transactions, including deposits, withdrawals, fund transfers, and bill payments, for a specified account or customer.
Balance Summary	An overview of the current balances for all accounts associated with a particular customer or client.
Account Activity Report	A summary of recent account activities, including deposits, withdrawals, and fund transfers, for a specified account or customer.
Loan Statement	A report detailing the status of a loan account, including the outstanding principal, interest accrued, payment history, and next payment due.
Audit Trail	A comprehensive log of all system activities, including user logins, access attempts, account modifications, and transaction processing.
Daily Transaction Report	A summary of transactions processed by the system within a specified day, including total deposits, withdrawals, and transfers, categorized by transaction type and account.
Monthly Financial Report	An overview of the financial health of the bank for a specific month, including total deposits, withdrawals, interest earned, loan balances, and net profit.
Customer Account Summary	A consolidated summary of all accounts held by a customer, including current balances, interest earned, and any outstanding loans or overdrafts.
Compliance Report	A report demonstrating the bank's compliance with regulatory requirements, including KYC (Know Your Customer), AML (Anti-Money Laundering), and GDPR (General Data Protection Regulation).

3.6 System Specification:

This subsection provides an overview of the system specifications, including hardware and software requirements, architecture design, and integration capabilities. It ensures that the system is scalable, reliable, and secure, meeting the needs of both current and future users. By defining system specifications, this section lays the groundwork for successful system implementation and deployment.

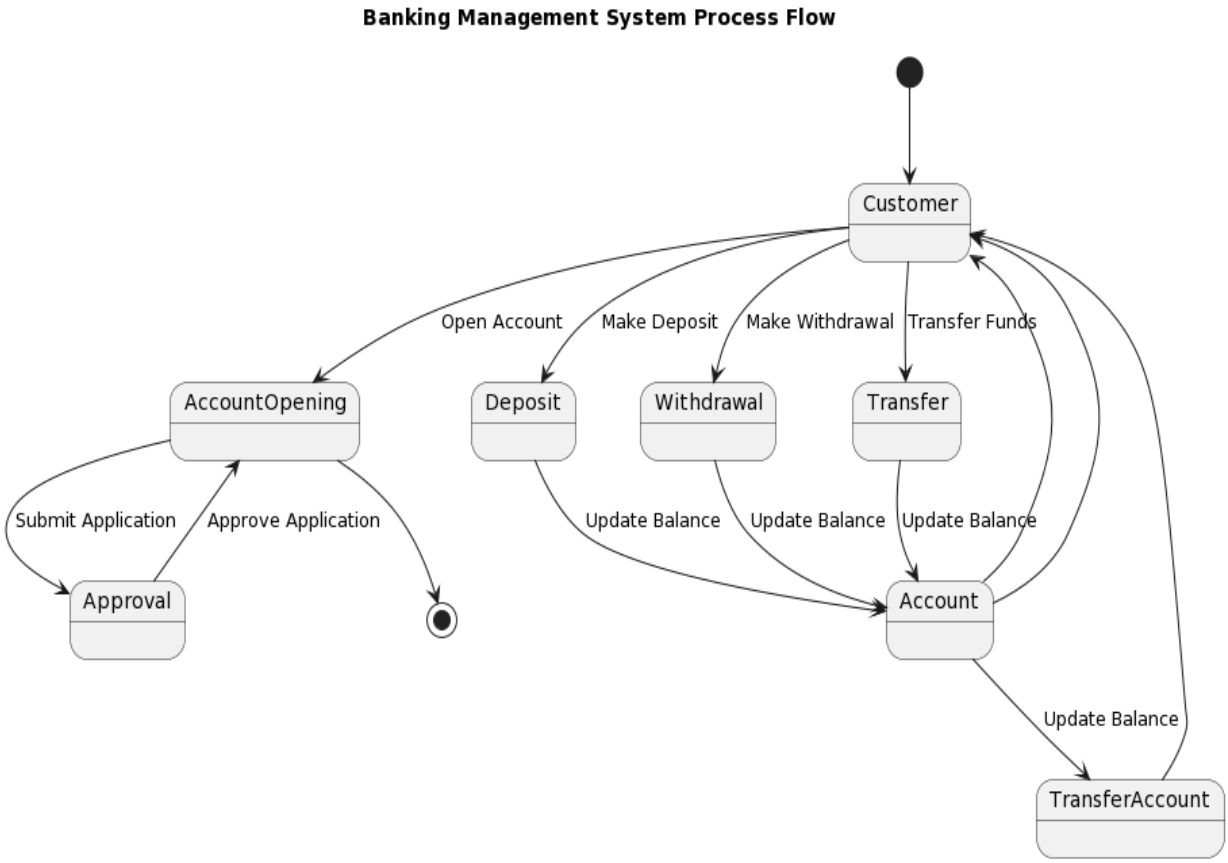
Feature	Description
System Architecture	Three-tier architecture with a web-based front end, PHP-based backend, and MySQL database back end.
Technologies Used	Frontend: HTML5, CSS3, JavaScript Backend: PHP Database: MySQL
Security	User authentication using bcrypt hashing for passwords in PHP, HTTPS encryption for secure data transmission, role-based access control (admin, teller, customer), transaction logging.
Account Management	Ability to create, update, and delete customer accounts using PHP and MySQL. Support for various types of accounts such as savings, checking, and loan accounts.
Transaction Processing	Support for deposit, withdrawal, fund transfers, bill payments, and loan applications using PHP and MySQL. Real-time transaction processing with balance updates.
Reporting	Generation of account statements, transaction history, and balance summaries using PHP and MySQL queries.
Notifications	Automated email or SMS notifications for account activities, payment reminders, and system alerts using PHP's mail function.
Compliance	Compliance with banking regulations such as KYC (Know Your Customer), AML (Anti-Money Laundering), and GDPR (General Data Protection Regulation) using PHP and MySQL.
Scalability	Ability to handle a large number of concurrent users and transactions using PHP's scalable architecture and MySQL's horizontal scaling capabilities.
Performance	Response times for critical operations (e.g., transaction processing, account retrieval) should be within acceptable limits even under peak loads.
Disaster Recovery	Regular backups of the MySQL database with off-site storage. Procedures for data recovery in case of system failures or disasters.
Logging and Auditing	Logging of all system activities including user actions, access attempts, and system errors using PHP and MySQL. Regular auditing of logs for security and compliance purposes.

3.7 System Design:

In this stage, the system requirements and specifications are translated into a detailed design for the online banking management system. It involves developing architectural diagrams, wireframes, and prototypes to visualize system components and interactions. By creating a detailed system design, this section ensures alignment between project objectives and technical implementation.

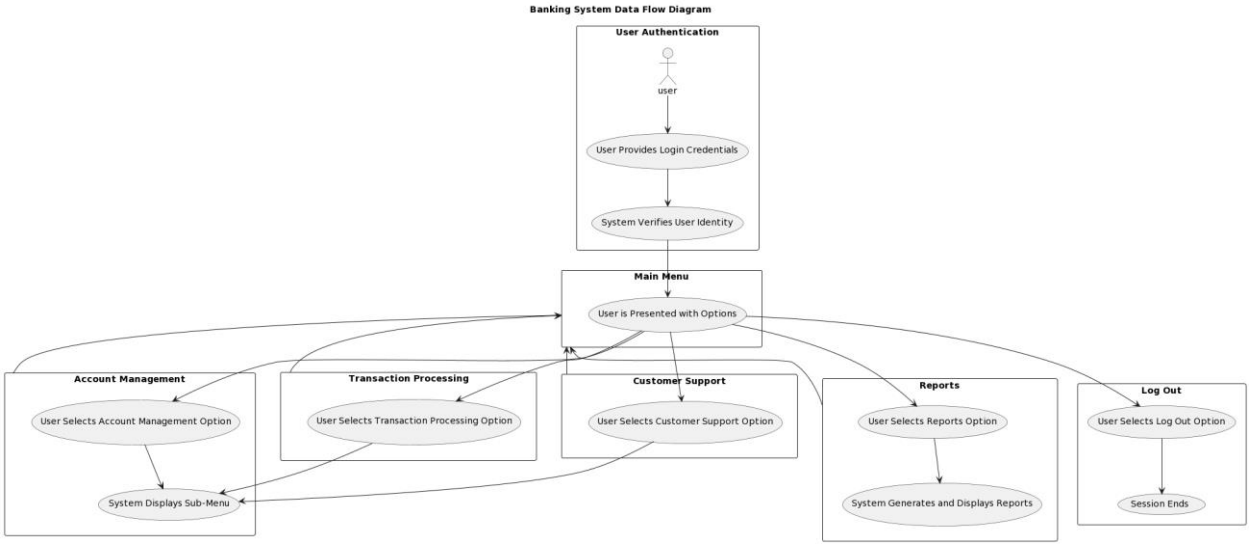
3.8 Process Flow:

Here, process flow diagrams are created to visualize the flow of activities and data within the online banking management system. It maps out the sequential steps involved in various banking processes, identifying dependencies and decision points for effective workflow management. By documenting process flows, this section enhances transparency and efficiency in system operation.



3.9 Data Flow Diagram:

This subsection develops data flow diagrams to illustrate the movement of data within the online banking management system. It identifies sources, processes, storage, and destinations of data, facilitating a clear understanding of data flow and interactions across system components. By visualizing data flows, this section ensures data integrity and consistency within the system.



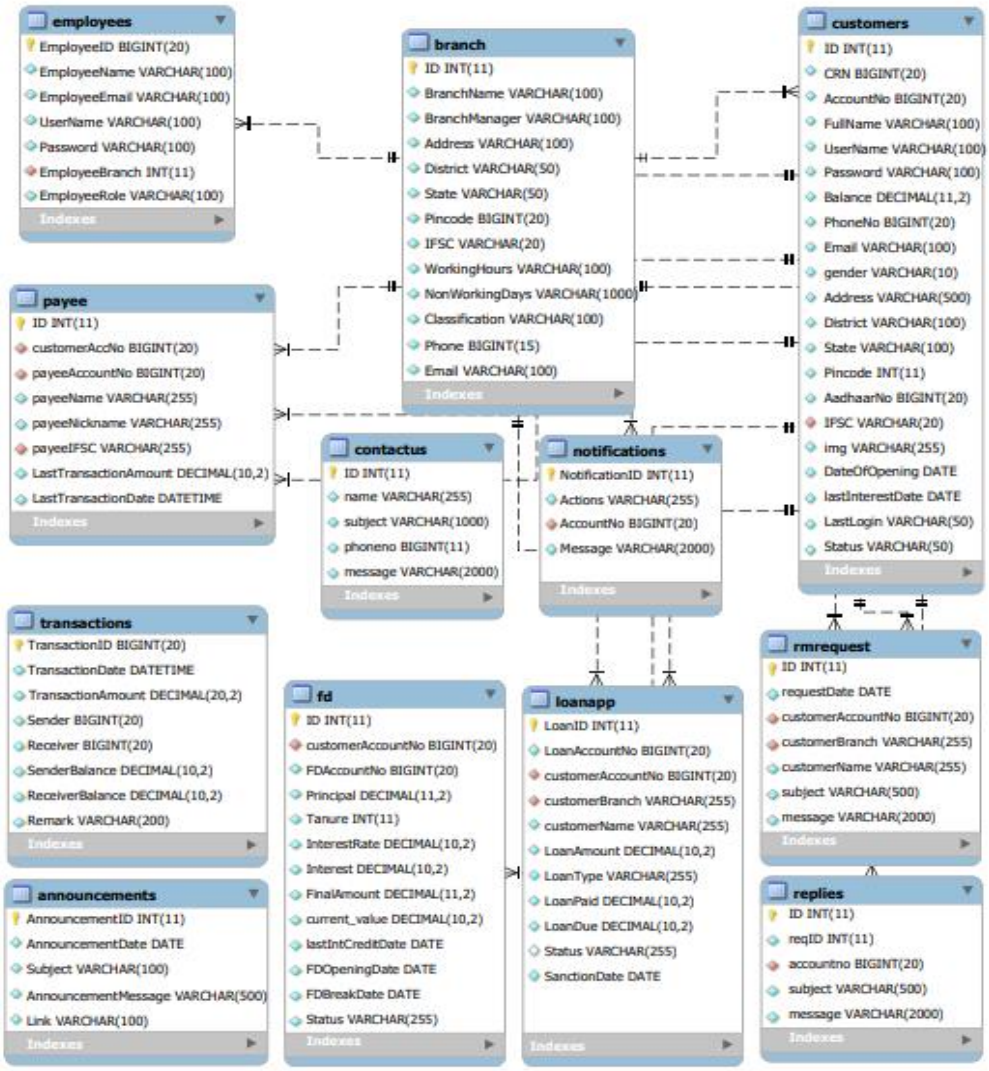
3.10 Flow Chart:

Flowcharts are created in this subsection to visualize the decision-making processes and logic within the online banking management system. It maps out different scenarios and outcomes, clarifying the sequence of actions and conditions that govern system behavior. By creating flowcharts, this section enhances transparency and efficiency in system operation.

3.11 UML Diagram:

Unified Modeling Language (UML) diagrams are utilized in this subsection to represent the system architecture, components, and relationships. It develops class diagrams, sequence diagrams, and use case diagrams to capture different aspects of the online banking management system. By utilizing UML diagrams, this section facilitates communication and collaboration among project stakeholders

BANKING MANAGEMENT SYSTEM EER DIAGRAM



3.12 Data Design:

The data design stage focuses on designing the database schema and data models for the online banking management system. It defines tables, relationships, and constraints to organize and structure the system's data, ensuring efficiency, integrity, and scalability. By designing the data schema, this section enhances data management and retrieval within the system.

3.13 Data Relationship:

Here, the relationships between different data entities within the online banking management system are explored. It defines associations, cardinality, and constraints to establish meaningful connections and dependencies, enabling efficient data retrieval and manipulation. By defining data relationships, this section ensures data consistency and accuracy within the system

4. System Requirements

4.1 Introduction:

Chapter 4 focuses on delineating the comprehensive set of system requirements for the development of the online banking management system. This chapter elucidates the functional and non-functional requirements, hardware and software prerequisites, and user interface specifications essential for the successful implementation of the system. By outlining clear and detailed requirements, stakeholders can align their expectations and priorities, guiding the subsequent phases of system development and testing.

4.2 Functional Requirements:

Functional requirements delineate the specific features, functionalities, and capabilities that the online banking management system must possess to meet the needs and expectations of users and stakeholders. This section provides a detailed breakdown of functional requirements across various system modules, including:

- User Authentication: The system must support secure user authentication mechanisms, such as username/password, biometric authentication, and two-factor authentication, to verify the identity of users accessing the system.

- Account Management: Users should be able to create, update, and delete their accounts, view account balances, transaction history, and account statements, and perform account-related tasks such as fund transfers, bill payments, and account settings customization.

- Transaction Processing: The system must facilitate seamless and secure processing of financial transactions, including fund transfers between accounts, bill payments to external vendors, loan applications, and credit/debit card transactions.

- Reporting and Analytics: The system should provide comprehensive reporting and analytics capabilities, enabling users to generate and analyze financial reports, transaction summaries, and performance metrics to make informed decisions.

- Security and Compliance: The system must adhere to stringent security standards and regulatory requirements, implementing robust data encryption, access controls, audit trails, and compliance checks to safeguard user data and prevent unauthorized access or fraudulent activities.

- Customer Support: The system should offer user-friendly customer support features, including online help documentation, FAQs, chat support, and ticketing systems, to assist users with inquiries, issues, and troubleshooting.

By delineating clear and concise functional requirements, stakeholders can ensure that the online banking management system delivers the desired features and functionalities to meet user needs and business objectives effectively.

4.3 Non-Functional Requirements:

Non-functional requirements specify the quality attributes, performance characteristics, and constraints that govern the behavior and operation of the online banking management system. This section categorizes non-functional requirements into various dimensions, including:

- Performance: The system should exhibit optimal performance under varying loads and concurrency levels, with minimal latency and response times for user interactions and transaction processing.

- Scalability: The system must be scalable to accommodate growing user bases, transaction volumes, and system loads, with the ability to dynamically allocate resources and scale horizontally or vertically as needed.

- Reliability: The system should demonstrate high reliability and availability, with minimal downtime, data loss, or service disruptions, through redundancy, failover mechanisms, and fault tolerance.

- Security: The system must enforce robust security measures, including data encryption, authentication, authorization, and intrusion detection, to protect against cyber threats, data breaches, and unauthorized access.

- Usability: The system should provide a user-friendly and intuitive interface, with clear navigation, informative feedback, and accessibility features, to ensure ease of use and accessibility for users of all backgrounds and abilities.

- Compatibility: The system must be compatible with a diverse range of devices, operating systems, browsers, and network environments, ensuring seamless access and functionality across different platforms and configurations.

- Compliance: The system must comply with relevant regulatory standards, industry best practices, and organizational policies, such as GDPR, PCI DSS, and ISO/IEC 27001, to ensure data privacy, security, and legal compliance.

By articulating non-functional requirements, stakeholders can prioritize and address critical quality attributes and constraints that impact the overall usability, performance, and reliability of the online banking management system.

4.4 Hardware and Software Requirements:

The hardware and software requirements delineate the infrastructure, technologies, and tools necessary for the development, deployment, and operation of the online banking management system. This section specifies the hardware specifications, such as server configurations, storage capacities, and network bandwidth, as well as the software requirements, including operating systems, web servers, database management systems, and development frameworks.

Additionally, this section outlines any third-party dependencies, APIs, libraries, or integration points required for system interoperability and functionality. By defining clear hardware and

software requirements, stakeholders can procure and provision the necessary resources and technologies to support the online banking management system effectively.

Requirement	Description
Hardware	
Server	Minimum: Dual-core processor, 8 GB RAM, 100 GB storage Recommended: Quad-core processor, 16 GB RAM, 200 GB SSD storage
Client Computers	Minimum: Dual-core processor, 4 GB RAM, 250 GB storage Recommended: Quad-core processor, 8 GB RAM, 500 GB HDD storage
Networking	Ethernet connection for server, LAN or WAN connectivity for client computers
Backup System	External storage device for regular backups of database and system files
Software	
Operating System	Server: Linux (e.g., Ubuntu Server, CentOS) or Windows Server Client: Windows 10, macOS, or Linux (Ubuntu, Fedora)
Web Server	Apache HTTP Server, Nginx
Database	MySQL (recommended), PostgreSQL
Programming Language	Backend: PHP (recommended), Node.js, Python Frontend: HTML5, CSS3, JavaScript
Frameworks/Libraries	Backend: Laravel (PHP), Express.js (Node.js), Django (Python) Frontend: Bootstrap, jQuery
Security Software	Firewall (e.g., iptables), Antivirus/Antimalware software
Version Control System	Git (for code versioning and collaboration)
Development Environment	Code editor (e.g., Visual Studio Code, Sublime Text), Terminal or Command Prompt for running commands, Git client
Additional Tools	Cron jobs or task scheduler for automated tasks (e.g., database backups), Email server for notifications (e.g., Postfix), SSH for remote access to server

4.5 User Interface Design:

User interface (UI) design encompasses the visual layout, interaction patterns, and usability features of the online banking management system, ensuring a seamless and intuitive user experience. This section describes the UI design principles, guidelines, and components employed in the system, including:

- Navigation: Clear and intuitive navigation menus, breadcrumbs, and links to facilitate easy traversal and exploration of system features and content.

- Layout: Consistent and visually appealing layout designs, with well-organized content, whitespace, and typography to enhance readability and comprehension.

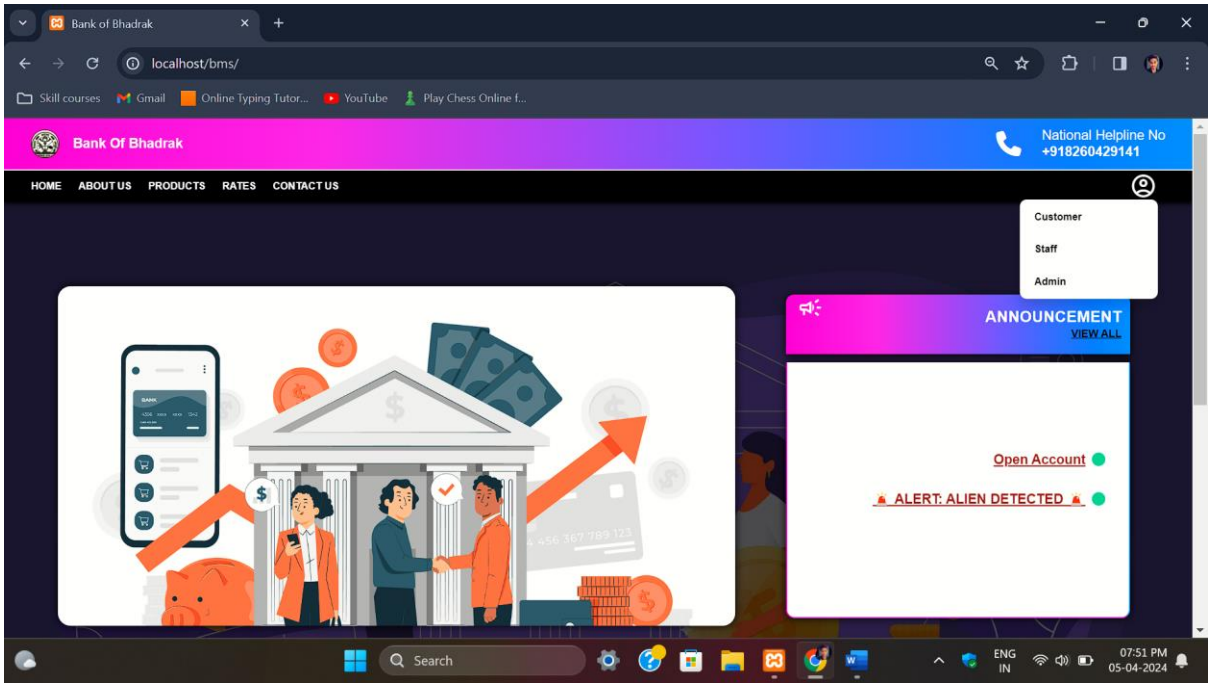
- Interaction: Interactive elements such as buttons, forms, dropdowns, and tooltips to enable user input, feedback, and actions within the system.

- Accessibility: Accessibility features such as keyboard navigation, screen reader compatibility, and text alternatives for images to ensure inclusivity and compliance with accessibility standards.

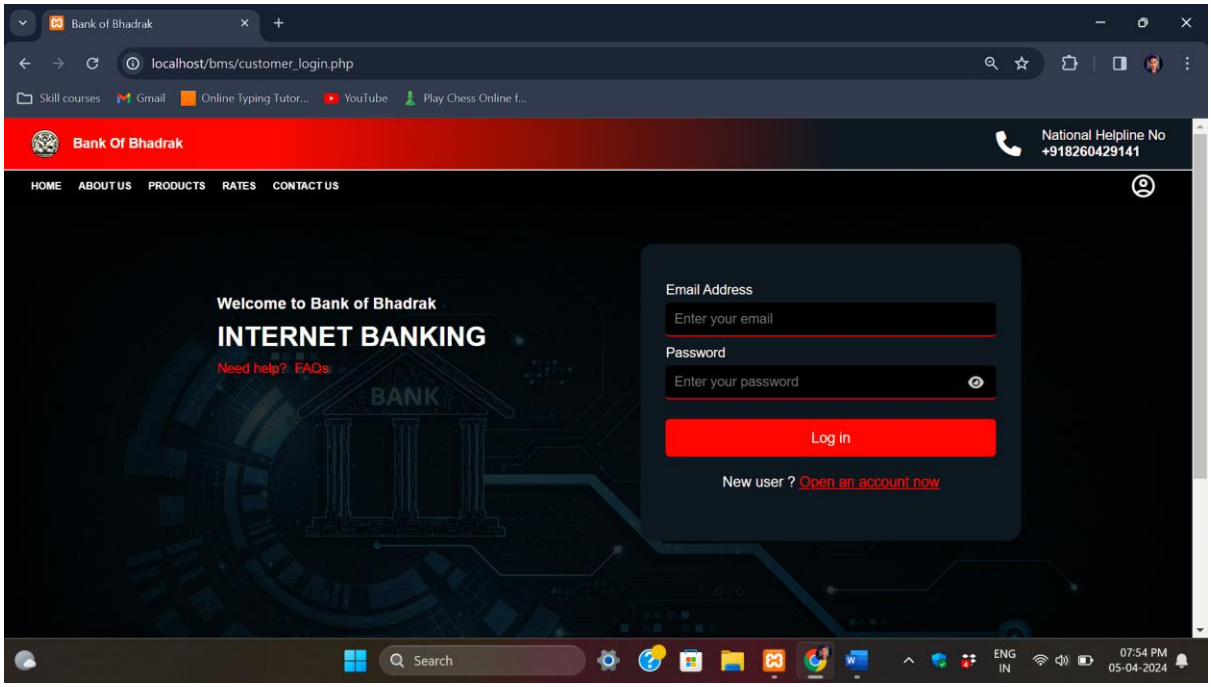
- Responsiveness: Responsive design techniques to adapt the user interface to different screen sizes, resolutions, and device orientations, providing a consistent experience across desktop, mobile, and tablet devices.

By prioritizing user interface design considerations, stakeholders can create an engaging and user-centric interface that enhances usability, satisfaction, and adoption of the online banking management system.

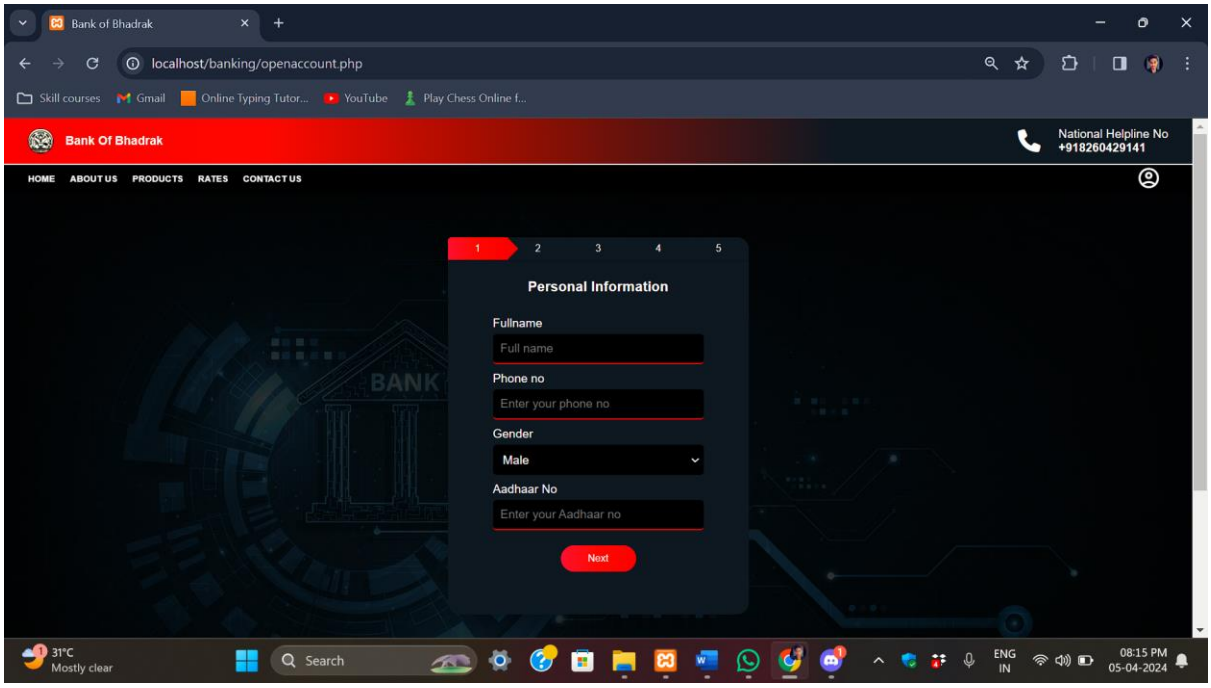
HOMEPAGE:



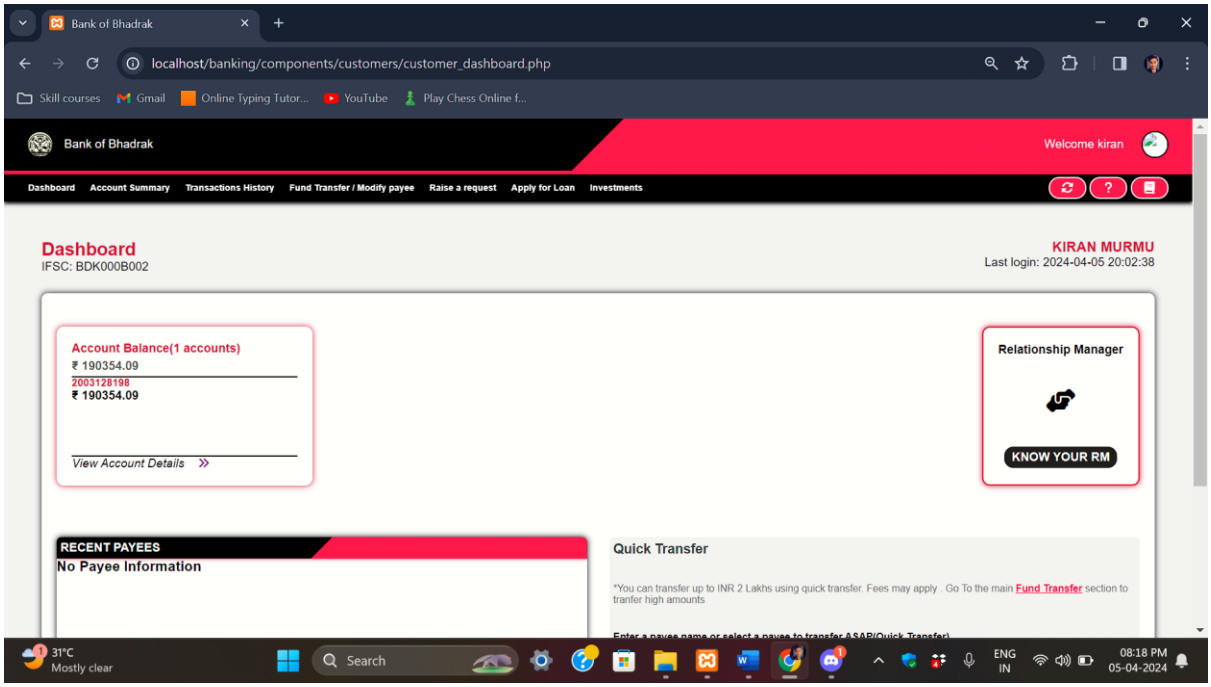
LOGIN PAGE :



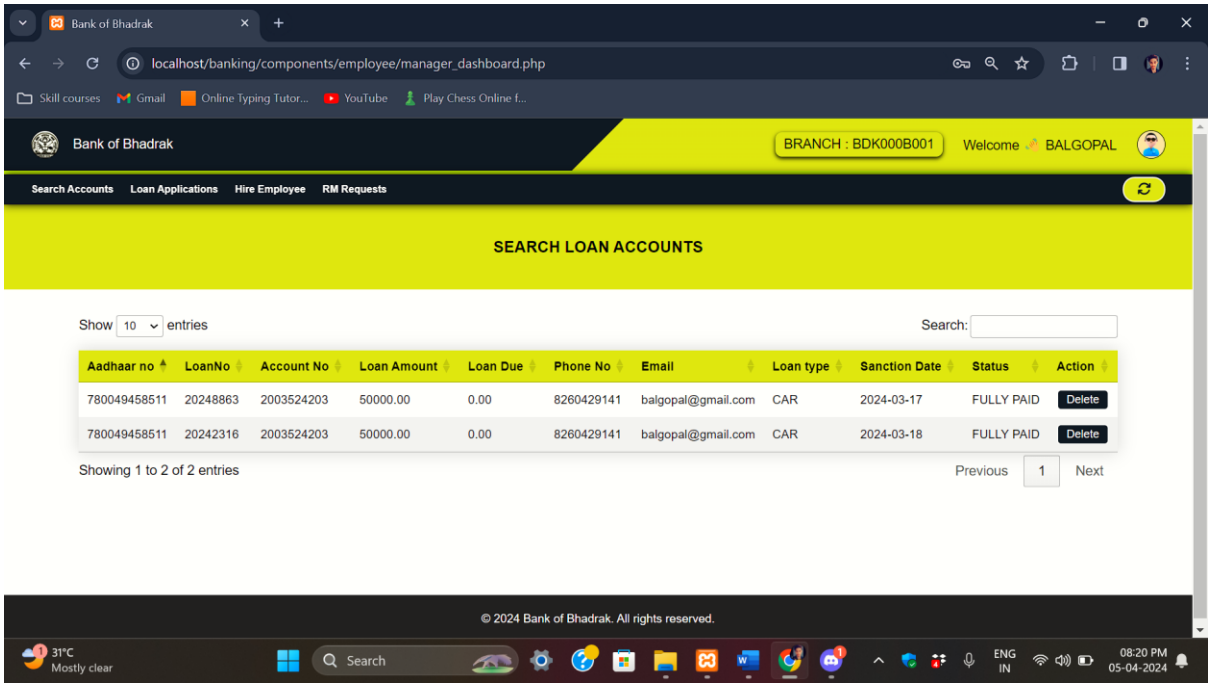
REGISTER PAGE:



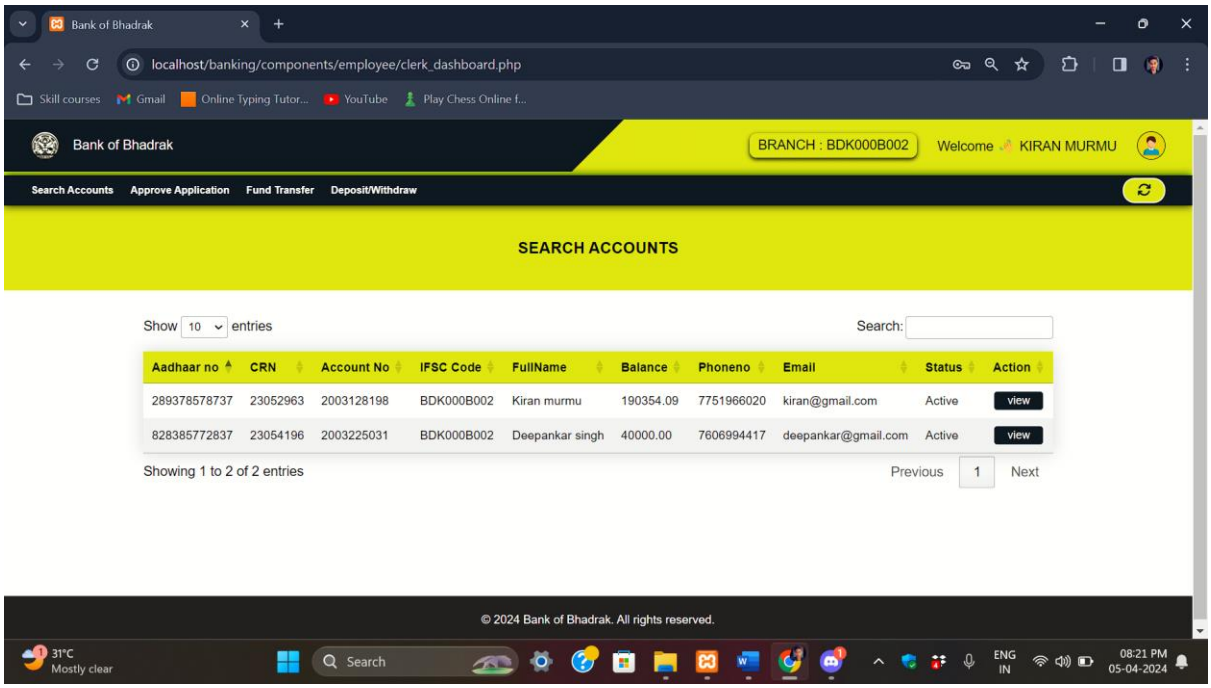
CUSTOMER DASHBOARD:



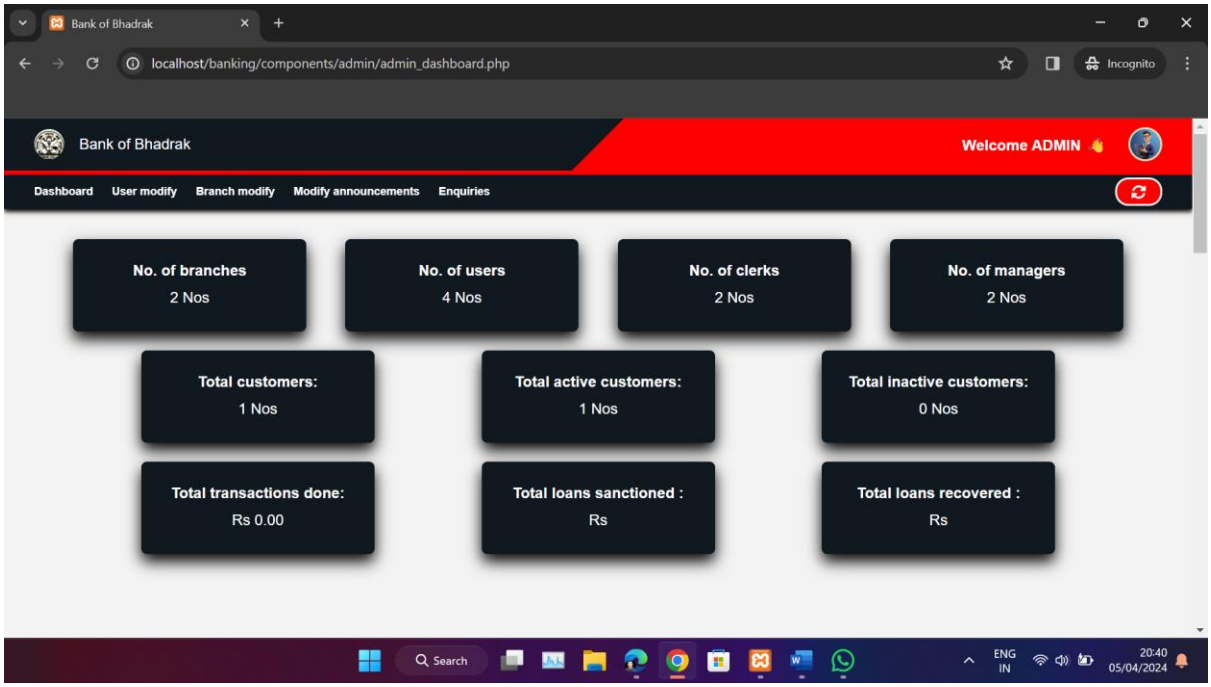
MANAGER DASHBOARD:



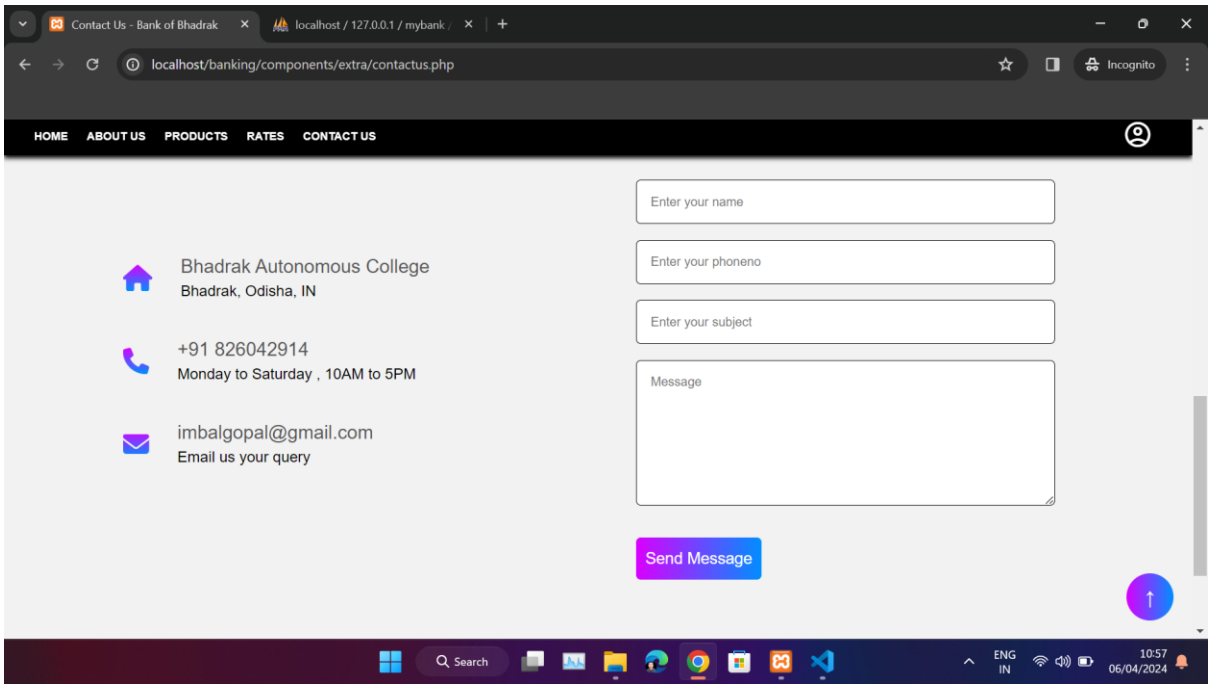
CLERK DASHBOARD:



ADMIN DASHBOARD:



CONTACT US PAGE:



5. System Design and Architecture

5.1 Introduction:

Chapter 5 delves into the design and architecture of the online banking management system, providing insights into the structural organization, component interactions, and technological frameworks underpinning its development. This chapter outlines the systematic approach to system design, encompassing conceptual modeling, architectural patterns, and component specifications, to translate the identified requirements into a robust and scalable system architecture.

5.2 System Architecture:

The system architecture defines the high-level structure and organization of the online banking management system, outlining the distribution of components, communication protocols, and deployment topology. This section presents the architectural design decisions, including:

- Client-Server Architecture: The system adopts a client-server architecture, with distinct client-side interfaces and a centralized server-side infrastructure. Client applications interact with server endpoints via secure APIs and protocols, facilitating seamless data exchange and transaction processing.

- Three-Tier Architecture: The system employs a three-tier architecture, comprising presentation, application, and data tiers. The presentation tier handles user interactions and interface rendering, the application tier implements business logic and process orchestration, and the data tier manages data storage, retrieval, and manipulation.

- Microservices Architecture: The system embraces a microservices architecture, decomposing complex functionalities into smaller, independent services that can be developed, deployed,

and scaled autonomously. Microservices communicate via lightweight APIs and messaging protocols, enabling agility, resilience, and scalability in system operations.

By adopting a robust system architecture, the online banking management system can achieve modularity, flexibility, and maintainability, facilitating future enhancements and updates.

5.3 Component Design:

Component design entails the specification and modeling of individual system components, modules, and services, detailing their interfaces, behaviors, and dependencies. This section delineates the design of key system components, including:

- User Interface Components: The system's user interface components encompass web pages, forms, widgets, and navigation elements that enable users to interact with the system effectively. UI components adhere to design principles such as consistency, responsiveness, and accessibility, ensuring a seamless and intuitive user experience.

- Business Logic Components: The system's business logic components encapsulate the core functionalities and rules governing transaction processing, account management, and security enforcement. Business logic components implement algorithms, workflows, and validation rules to orchestrate system operations and enforce business policies.

- Data Access Components: The system's data access components facilitate interaction with underlying data stores, such as relational databases, NoSQL repositories, or external APIs. Data access components implement data access patterns such as CRUD operations, querying, and caching to retrieve, store, and manipulate data efficiently.

By designing modular and cohesive components, the online banking management system can achieve encapsulation, reusability, and testability, enabling iterative development and evolution.

5.4 Database Design:

Database design encompasses the structuring and organization of data within the online banking management system, defining the schemas, tables, and relationships that govern data storage and retrieval. This section outlines the database design considerations, including:

- Entity-Relationship Model: The system adopts an entity-relationship model to represent the entities, attributes, and relationships within the domain of online banking. Entities such as users, accounts, transactions, and security roles are modeled along with their respective attributes and associations.

- Normalization: The system applies normalization techniques to eliminate data redundancy and ensure data integrity within the database. Normalized database schemas adhere to normalization forms such as 1NF, 2NF, and 3NF, reducing update anomalies and improving data consistency.

- Indexing and Optimization: The system employs indexing and optimization strategies to enhance database performance and query efficiency. Indexes are created on frequently queried columns, and optimization techniques such as query tuning and caching are applied to minimize response times and resource utilization.

By designing a robust and scalable database schema, the online banking management system can ensure data integrity, consistency, and performance across various operations and workloads.

5.5 System Integration:

System integration involves the seamless integration of disparate system components, third-party services, and external data sources to enable end-to-end functionality and interoperability. This section discusses the integration points, protocols, and mechanisms employed in system integration, including:

- API Integration: The system integrates with external APIs and web services to facilitate data exchange, payment processing, and authentication with third-party providers such as payment gateways, credit bureaus, and identity verification services.

- Legacy System Integration: The system integrates with legacy systems and backend databases to migrate existing data, processes, and functionalities seamlessly. Integration adapters, middleware, or data migration tools are employed to bridge the gap between modern and legacy systems.

- Event-Driven Architecture: The system adopts an event-driven architecture to enable real-time communication and event propagation between system components. Events such as user actions, system notifications, and data updates trigger asynchronous workflows and message exchanges, enhancing system responsiveness and scalability.

By orchestrating seamless system integration, the online banking management system can leverage the capabilities of external services and legacy systems, enriching its functionality and enhancing the overall user experience.

5.6 Security and Compliance:

Security and compliance considerations are paramount in the design and architecture of the online banking management system, ensuring the confidentiality, integrity, and availability of sensitive user data and transactions. This section discusses the security measures,

protocols, and best practices employed to safeguard the system against security threats and regulatory violations, including:

- Encryption and Hashing: The system employs encryption algorithms and hashing techniques to secure data transmission, storage, and authentication processes. Strong encryption standards such as AES and TLS are utilized to protect sensitive information from unauthorized access and interception.

- Access Control and Authentication: The system implements robust access control mechanisms and authentication protocols to verify the identity of users and regulate their access to system resources. Role-based access control (RBAC), multi-factor authentication (MFA), and biometric authentication are employed to enforce granular access permissions and strengthen user authentication.

- Audit Logging and Monitoring: The system incorporates audit logging and monitoring features to track user activities, system events, and security incidents in real-time. Audit logs capture detailed information such as user login attempts, transaction history, and system configurations, enabling forensic analysis, compliance auditing, and incident response.

By prioritizing security and compliance in system design and architecture, the online banking management system can mitigate security risks, ensure regulatory compliance, and foster trust and confidence among users and stakeholders.

6. Implementation and Development

6.1 Introduction:

Chapter 6 focuses on the practical implementation and development of the online banking management system. This chapter provides insights into the coding process, software development methodologies, tools, and technologies employed to translate the system design into a functional and operational solution. By detailing the implementation approach and showcasing the system's features and functionalities, stakeholders can gain a deeper understanding of the development process and its outcomes.

6.2 Software Development Methodology:

The software development methodology outlines the systematic approach and workflow followed during the implementation of the online banking management system. This section discusses the chosen development methodology, whether it be agile, waterfall, or iterative, and elucidates how it aligns with the project objectives, team dynamics, and timeline constraints. Additionally, it highlights the key phases, activities, and deliverables associated with the chosen methodology, providing insights into the development process and its iterations.

6.3 Coding:

The coding phase involves the actual implementation of system functionalities and features using programming languages, frameworks, and libraries. This section provides an overview of the coding process, including:

- Programming Languages: The system is implemented using programming languages such as PHP, JavaScript, HTML, CSS, and SQL, leveraging their respective strengths in backend, frontend, and database development.

- Frameworks and Libraries: The development process utilizes frameworks and libraries such as Spring Boot, Angular, React, and Hibernate to expedite development, streamline code organization, and enhance code reusability.

- Coding Standards and Conventions: The coding process adheres to established coding standards, conventions, and best practices to ensure consistency, readability, and maintainability of the codebase. Code reviews, linting tools, and automated testing are employed to enforce coding standards and identify potential issues early in the development cycle.

By following a systematic and disciplined approach to coding, the development team can produce clean, efficient, and maintainable code that aligns with the system's requirements and design specifications.

Customer login page:

customer_login.php :

```
<?php
require $_SERVER['DOCUMENT_ROOT'] . "/banking/assets/php/config.php";
session_start();
require_once $_SERVER['DOCUMENT_ROOT'] . "/banking/assets/php/prac.php";
if (isset($_POST['submit'])) {
    $email = strtolower(mysqli_real_escape_string($conn,
$_POST['email']));
    $password = mysqli_real_escape_string($conn, $_POST['password']);

    if (!empty($email) && !empty($password)) {
        $sql = "SELECT * FROM customers WHERE Email = '$email' AND
Status='ACTIVE'";
        $result = mysqli_query($conn, $sql);

        if ($result && mysqli_num_rows($result) > 0) {
            $row = mysqli_fetch_assoc($result);
            $enc_pass = $row['Password'];
```

```
            if (md5($password) === $enc_pass) {
                $_SESSION['isLoggedin'] = true;
                $_SESSION['email'] = $email;
                $_SESSION['LastLogin'] = $row['LastLogin'];
                $_SESSION['currentLogin'] = time();

                $balance = $row['Balance'];
                $accountNo = $row['AccountNo'];
                $LastLogin = $row['LastLogin'];
                $LastIntCreditDate = $row['lastInterestDate'];
                $lastIntDatee = DateTime::createFromFormat('Y-m-d',
$LastIntCreditDate);
                $today = new DateTime('today');
                $diff = $lastIntDatee->diff($today);
                $formattedDate = $today->format('Y-m-d');

                if ($diff->days > 0) {
                    $dailyInterest = ($balance * $diff->days * 0.010958)
/ 100;
                    $newBalance = $balance + $dailyInterest;
                    mysqli_query($conn, "UPDATE `customers` SET
`Balance`=$newBalance,lastInterestDate='$formattedDate'  WHERE `Email` =
'$email';");
                    mysqli_query($conn, "INSERT INTO
`transactions`(`TransactionAmount`,`Receiver`,`Actions`,`ReceiverBalance`
) VALUES ('$dailyInterest','$accountNo','int._credited',$newBalance);");
                }

                if (mysqli_num_rows($get_fd = mysqli_query($conn, "SELECT
* FROM fd WHERE customerAccountNo = $accountNo")) > 0) {
                    while ($fd_data = mysqli_fetch_assoc($get_fd)) {
                        $fdInterest = $fd_data['InterestRate'];
                        $currBalance = $fd_data['current_value'];
                        $fdOpenDate = $fd_data['FDOpeningDate'];
                        $fdBreakDate = $fd_data['FDBreakDate'];
                        $fdLastIntDate = $fd_data['lastIntCreditDate'];
                        $FDIntDatee = DateTime::createFromFormat('Y-m-d',
$fdLastIntDate);
                        $FDBDatee = DateTime::createFromFormat('Y-m-d',
$fdBreakDate);
                        $today = new DateTime();
                        $diffFD = $FDIntDatee->diff($today);
                        if ($diffFD->days > 0) {
                            $dailyFDint = $fdInterest / 365;
                            $dailyReturn = ($currBalance * $diffFD->days
* $dailyFDint) / 100;
```

```
                            $newCurrBalance = $currBalance +
$dailyReturn;
                            mysqli_query($conn, "UPDATE `fd` SET
`current_value` = '$newCurrBalance',
lastIntCreditDate='$formattedDate'  WHERE customerAccountNo =
$accountNo;");
                        }
                    }
                }

                header('Location:
./components/customers/customer_dashboard.php');
                exit;
            } else {
                $error_message = 'Password is incorrect';
            }
        } else {
            $error_message = 'This email does not exist / Inactive
account';
        }
    } else {
        $error_message = 'All input fields are required';
    }
}
?>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Bank of Bhadrak</title>
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:
opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.1/css/all.min.css" integrity="sha512-
DTOQO9RWCH3ppGqcWaEA1BIZOC6xxalwEsw9c2QQeAIftl+Vegovlnee1c9QX4TctnWMn13TZ
ye+giMm8e2LwA==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <script src="/banking/assets/javascript/app.js"></script>
    <link rel="stylesheet" href="style.css">
</head>
<body style="flex-direction: column;">
    <nav class="navbar login-page">
        <div class="top-nav">
            <div>
```

```
                <a href="#">
                    <img src="./images/logo.jpg" alt="">
                    <p>Bank Of Bhadrak</p>
                </a>
            </div>
            <div class="help-line">
                <i class="fa-solid fa-phone"></i>
                <div>
                    National Helpline No
                    <br>
                    <a href="">
                        +918260429141
                    </a>
                </div>
            </div>
        </div>
    </nav>
    <div class="bottom-nav">
        <div class="signin s1">
            <ul>
                <li><a href="/banking/index.php">HOME</a></li>
                <li><a href="/banking/components/extra/about-
us.php">ABOUT US</a></li>
                <li><a
href="/banking/components/extra/products.php">PRODUCTS</a></li>
                <li><a
href="/banking/components/extra/rates.php">RATES</a></li>
                <li><a
href="/banking/components/extra/contactus.php">CONTACT US</a></li>
            </ul>
        </div>
        <div class="dropdown login-page">
            <span class="material-symbols-outlined
users">account_circle</span>
            <div class="items">
                <a href="customer_login.php">Customer</a>
                <a href="employee_login.php">Staff</a>
                <a href="admin_login.php">Admin</a>
            </div>
        </div>
    </div>
    <main class="clogin-main" style="height:355px;">
        <div class="bank-title">
            <h3>Welcome to Bank of Bhadrak</h3>
            <h1>INTERNET BANKING</h1>
            <div class="link">
```

```
                <a href="/banking/components/extra/contactus.php">Need
help?</a>
                <a href="/banking/components/extra/faqs.php">FAQs</a>
            </div>
        </div>

        <div class="wrapper">
            <section class="form login">
                <form action="" method="POST" enctype="multipart/form-
data" autocomplete="off">
                    <div class="error-text"></div>
                    <div class="field input">
                        <label>Email Address</label>
                        <input type="text" name="email"
placeholder="Enter your email" required>
                    </div>
                    <div class="field input">
                        <label>Password</label>
                        <input type="password" name="password"
placeholder="Enter your password" required>
                        <i  class="fas fa-eye"></i>
                    </div>
                    <div class="field button">
                        <input type="submit" name="submit" value="Log
in">
                    </div>
                </form>
                <div class="link">New user ? <a
href="openaccount.php">Open an account now</a></div>
            </section>
        </div>
    </main>
    <script src="/banking/assets/javascript/pass-show-hide.js"></script>
    <footer class="login-page">
        <p>&copy; 2024 Bank of Bhadrak. All rights reserved. |
            <a href="/banking/components/extra/about-us.php">About Us</a>
|
            <a href="/banking/components/extra/products.php">Services</a>
|
            <a href="/banking/components/extra/faqs.php">FAQs</a> |
            <a href="/banking/components/extra/terms-
conditions.php">Terms and Conditions</a> |
            <a href="/banking/components/extra/privacy-
policy.php">Privacy Policy</a>
        </p>
    </footer>
</body>
```

```
</html>

<?php
if (isset($error_message)) {
    echo "<script>
    var err = document.querySelector('.error-text');
    err.style.display='block';
    err.innerText='$error_message';
    </script>";
}
?>
```

CUSTOMER MODULE FUNCTIONS:

functions.php:

```
<?php
function updateTransaction($sendersAccountNo, $receiversAccountNo,
$amount, $remark)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $senders_info = mysqli_query($conn, "SELECT * FROM customers WHERE
AccountNo='$sendersAccountNo'");
    $receivers_info = mysqli_query($conn, "SELECT * FROM customers WHERE
AccountNo='$receiversAccountNo'");
    $sender_data = mysqli_fetch_assoc($senders_info);
    $receiver_data = mysqli_fetch_assoc($receivers_info);
    if (mysqli_num_rows($receivers_info) > 0) {
        $new_senders_balance = $sender_data['Balance'] - $amount;
        $new_receivers_balance = $receiver_data['Balance'] + $amount;
        mysqli_query($conn, "UPDATE customers SET
Balance='$new_senders_balance' WHERE AccountNo='$sendersAccountNo'");
        mysqli_query($conn, "UPDATE customers SET
Balance='$new_receivers_balance' WHERE AccountNo='$receiversAccountNo'");
        mysqli_query($conn, "INSERT INTO
`transactions`(`TransactionAmount`, `Sender`, `Receiver`,`Actions`,
`Remark`,`SenderBalance`, `ReceiverBalance`) VALUES
('$amount','$sendersAccountNo','$receiversAccountNo','Transfer','$remark'
,$new_senders_balance,$new_receivers_balance);");
        $today = date('Y-m-d H:i:s');
        $senderNotification = "A/c debited Rs.$amount on $today to
$receiversAccountNo. Total Bal : Rs.$new_senders_balance";
        $receiverNotification = "Your A/c is credited by Rs.$amount ,
Total Bal : Rs.$new_receivers_balance as on: $today";
```

```
        mysqli_query($conn, "INSERT INTO `notifications`(`Actions`,
`AccountNo`, `Message`) VALUES ('debit', $sendersAccountNo,
'$senderNotification')");
        mysqli_query($conn, "INSERT INTO `notifications`(`Actions`,
`AccountNo`, `Message`) VALUES ('credit', $receiversAccountNo,
'$receiverNotification')");
        echo "<script>alert('Fund transfer successful');
window.location.href = '" . $_SERVER['HTTP_REFERER'] . "';</script>";
    } else {
        echo "<script>var err = document.getElementsByClassName('error-
text')[0];
        err.innerText = 'Receiver account not found';
        err.style.display='block';
        </script>";
    }
}


function calculateInterest($principal, $days, $tanure)
{
    if ($days >= 7 && $days <= 14) {
        $interestRate = 2.80;
    } elseif ($days >= 15 && $days <= 29) {
        $interestRate = 2.80;
    } elseif ($days >= 30 && $days <= 45) {
        $interestRate = 3.00;
    } elseif ($days >= 46 && $days <= 90) {
        $interestRate = 3.25;
    } elseif ($days >= 91 && $days <= 120) {
        $interestRate = 3.50;
    } elseif ($days >= 121 && $days <= 180) {
        $interestRate = 3.85;
    } elseif ($days >= 181 && $days < 270) {
        $interestRate = 4.50;
    } elseif ($days >= 270 && $days < 365) {
        $interestRate = 4.75;
    } elseif ($days == 365) {
        $interestRate = 6.10;
    } elseif ($days > 365 && $days < 730) {
        $interestRate = 6.30;
    } elseif ($days >= 730 && $days < 1095) {
        $interestRate = 6.70;
    } elseif ($days >= 1095 && $days < 1825) {
        $interestRate = 6.25;
    } elseif ($days == 1825) {
        $interestRate = 6.25;
```

```
    } elseif ($days > 1825) {
        $interestRate = 6.10;
    } else {
        return "Invalid number of days.";
    }
    $tanure = $tanure / 12;
    $interest = ($principal * $interestRate * $tanure) / 100;
    $totalAmount = $principal + $interest;
    $data = array("principal" => $principal, "interestRate" =>
$interestRate, "interest" => $interest, "totalAmount" => $totalAmount);
    return $data;
}

function fdCreate($accountNo, $principal,$currentDate, $tanure, $endDate,
$interestRate, $interest, $finalAmount, $balance)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $strFDNo = '9124' . rand(100000, 999999);
    $FDno = (int)$strFDNo;
    mysqli_query($conn, "INSERT INTO `fd`(`customerAccountNo`,
`FDAccountNo`, `Principal`, `Tanure`, `InterestRate`,`Interest`,
`FinalAmount`,`current_value`,`FDOpeningDate`, `FDBreakDate`, `Status`)
VALUES
($accountNo,$FDno,$principal,$tanure,$interestRate,$interest,$finalAmount
,$principal,'" . $currentDate->format('Y-m-d') . "','" . $endDate-
>format('Y-m-d') . "','ONGOING')");
    $today = $currentDate->format('Y-m-d');
    $years = (int)($tanure / 12);
    $months = $tanure % 12;
    $notification = "Rs $principal of FD is created on $today for $years
years and $months months";
    mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES
('FDCreated',$accountNo,'$notification')");

    $bankBalance = $balance - $principal;
    mysqli_query($conn, "UPDATE customers SET Balance=$bankBalance WHERE
AccountNo=$accountNo");

    mysqli_query($conn, "INSERT INTO
`transactions`(`TransactionAmount`,`Sender`,
`Actions`,`Remark`,`SenderBalance`) VALUES
('$principal','$accountNo','fd_booked','$FDno',$bankBalance);");
}
```

```
function fdBreak($accountNo, $fdNo, $balance, $fdAmount)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $bankBalance = $fdAmount + $balance;
    mysqli_query($conn, "DELETE FROM fd WHERE FDAccountNo=$fdNo");
    mysqli_query($conn, "UPDATE customers SET Balance=$bankBalance WHERE
AccountNo=$accountNo");

    $notification = "Rs $fdAmount of FD is breaked and money transferred
to your accountno $accountNo successfully";
    mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES
('FDBreaked',$accountNo,'$notification')");

    mysqli_query($conn, "INSERT INTO
`transactions`(`TransactionAmount`,`Receiver`,`Actions`,`Remark`,`Receive
rBalance`) VALUES
('$fdAmount','$accountNo','fd_breaked','$fdNo',$bankBalance);");
}


function applyLoan($accountNo, $amount, $loanType, $ifsc,$name)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $strLoanNo = '2024' . rand(1000, 9999);
    $loanNo = (int)$strLoanNo;

    mysqli_query($conn, "INSERT INTO `loanapp`(`LoanAccountNo`,
`customerAccountNo`,`customerBranch`,`customerName`, `LoanAmount`,
`LoanType`) VALUES
($loanNo,$accountNo,'$ifsc','$name',$amount,'$loanType')");
    echo "<script>alert('Your loan application is received. wait for
approval.'); window.location.href = '" . $_SERVER['HTTP_REFERER'] .
"';</script>";
}

function paydue($accountno,$loanNo,$payamount){
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $account=mysqli_query($conn,"SELECT * FROM customers WHERE
AccountNo=$accountno");
    $loan=mysqli_query($conn,"SELECT * FROM loanapp WHERE
LoanAccountNo=$loanNo");
    $account_data=mysqli_fetch_assoc($account);
    $loan_data=mysqli_fetch_assoc($loan);
    if($payamount>$loan_data['LoanDue']){
        echo "<script>alert('You are over paying your due. Transaction
aborted !!');</script>";
```

```
    }else{
        $bankBalance=$account_data['Balance']-$payamount;
        $remainingDue=$loan_data['LoanDue']-$payamount;
        $totalPaid=$loan_data['LoanPaid']+$payamount;
        mysqli_query($conn,"UPDATE customers SET Balance=$bankBalance
WHERE AccountNo=$accountno");
        if($remainingDue == 0){
            mysqli_query($conn,"UPDATE loanapp SET
LoanPaid=$totalPaid,LoanDue=$remainingDue,Status='FULLY PAID' WHERE
LoanAccountNo=$loanNo");
            echo "<script>alert('Fully paid your due for Loan A/c:
$loanNo'); window.location.href = '" . $_SERVER['HTTP_REFERER'] .
"';</script>";
        }else{
            mysqli_query($conn,"UPDATE loanapp SET
LoanPaid=$totalPaid,LoanDue=$remainingDue,Status='PARTIAL PAYMENT DONE'
WHERE LoanAccountNo=$loanNo");
            echo "<script>alert('Payment successful . remaining amount.
$remainingDue'); window.location.href = '" . $_SERVER['HTTP_REFERER'] .
"';</script>";
        }

        mysqli_query($conn, "INSERT INTO
`transactions`(`TransactionAmount`,`Sender`,
`Actions`,`Remark`,`SenderBalance`) VALUES
('$payamount','$accountno','loan_due_paid','$loanNo',$bankBalance);");

        $notification = "you paid Rs $payamount and remaining due is : Rs
$remainingDue";
        mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES ('loanPaid',$accountno,'$notification')");

        echo "<script>alert('Payment Successful') window.location.href =
'" . $_SERVER['HTTP_REFERER'] . "';</script>";
    }

}

?>
```

EMPLOYEE MODULE FUNCTIONS:

functions.php (EMPLOYEE MODULE) :

```
<?php

function deposit($accountno, $amount, $balance)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $balance = $balance + $amount;
    mysqli_query($conn, "UPDATE customers SET Balance=$balance WHERE
AccountNo=$accountno");
    echo "<script>alert('New Account Balance : " . $balance .
"');</script>";
    $notification = "Amount of Rs. $amount has been deposited to your
A/c: $accountno. Total Bal : Rs.$balance";
    mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES ('credit',$accountno,'$notification')");

    mysqli_query($conn, "INSERT INTO `transactions`(`TransactionAmount`,
`Receiver`, `Actions`,`ReceiverBalance`) VALUES
('$amount','$accountno','deposit',$balance);");
}
function withdraw($accountno, $amount, $balance)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $balance = $balance - $amount;
    mysqli_query($conn, "UPDATE customers SET Balance=$balance WHERE
AccountNo=$accountno");
    echo "<script>alert('New Account Balance : " . $balance .
"');</script>";
    $notification = "Amount of Rs. $amount has been withdrawn from your
A/c: $accountno. Total Bal : Rs. $balance";
    mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES ('debit',$accountno,'$notification')");

    mysqli_query($conn, "INSERT INTO `transactions`(`TransactionAmount`,
`Sender`, `Actions`,`SenderBalance`) VALUES
('$amount','$accountno','withdraw',$balance);");
}

function fundTransfer($sendersAccountNo, $receiversAccountNo,
$senderBalance, $receiverBalance, $amount, $remark)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $senderBalance = $senderBalance - $amount;
    $receiverBalance = $receiverBalance + $amount;
```

```
    mysqli_query($conn, "UPDATE customers SET Balance=$senderBalance
WHERE AccountNo=$sendersAccountNo;");
    mysqli_query($conn, "UPDATE customers SET Balance=$receiverBalance
WHERE AccountNo=$receiversAccountNo;");
    mysqli_query($conn, "INSERT INTO `transactions`(`TransactionAmount`,
`Sender`, `Receiver`,`Actions`,
`Remark`,`SenderBalance`,`ReceiverBalance`) VALUES
('$amount','$sendersAccountNo','$receiversAccountNo','Transfer','$remark'
,$senderBalance,$receiverBalance);");

    $today = Date('Y-m-d H:i:s');
    $senderNotification = "A/c debited Rs.$amount on $today to
$receiversAccountNo. Total Bal : Rs.$senderBalance";
    $receiverNotification = "Your A/c is credited by Rs.$amount from
$sendersAccountNo, Total Bal : Rs.$receiverBalance as on: $today";

    mysqli_query($conn, "INSERT INTO `notifications`(`Actions`,
`AccountNo`, `Message`) VALUES ('debit', $sendersAccountNo,
'$senderNotification')");
    mysqli_query($conn, "INSERT INTO `notifications`(`Actions`,
`AccountNo`, `Message`) VALUES ('credit', $receiversAccountNo,
'$receiverNotification')");
}

function approve($email, $crn, $accountno, $ifsc)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    $currentDate = date('Y:m:d');
    $time = date("Y-m-d H:i:s");
    if (mysqli_query($conn, "UPDATE customers SET CRN = $crn , AccountNo
= $accountno , DateOfOpening='$currentDate', Status = 'Active' , IFSC =
'$ifsc', LastLogin='$time' WHERE Email='$email'")) {
        echo "<script>alert('Account Created');</script>";
        if ($result = mysqli_fetch_assoc(mysqli_query($conn, "SELECT *
FROM customers WHERE Email='$email'"))) {
            echo "<script>alert('Account no : " . $result['AccountNo'] .
"');
            window.location.href = '" . $_SERVER['HTTP_REFERER'] .
"';</script>";
        }
    }
}

function loanApprove($loanNo, $loanAmount, $accountno)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
```

```
    $loan_data = mysqli_fetch_assoc(mysqli_query($conn, "SELECT * FROM
loanapp WHERE LoanAccountNo=$loanNo"));
    $loanType = $loan_data['LoanType'];
    $tanure = 5;
    if ($loanType == 'personal') {
        $interestRate = 12.75;
    } elseif ($loanType == 'home') {
        $interestRate = 8.35;
    } elseif ($loanType == 'study') {
        $interestRate = 6.50;
    } elseif ($loanType == 'business') {
        $interestRate = 14.25;
    } elseif ($loanType == 'car') {
        $interestRate = 9.75;
    }
    $interest = ($loanAmount * $interestRate * $tanure) / 100;
    $totalAmount = $loanAmount + $interest;

    $sanctionDate = date('Y-m-d');
    mysqli_query($conn, "UPDATE loanapp SET Status='SANCTIONED',
LoanDue=$totalAmount , SanctionDate='$sanctionDate' WHERE
LoanAccountNo=$loanNo");

    $result = mysqli_fetch_assoc(mysqli_query($conn, "SELECT * FROM
customers WHERE AccountNo=$accountno"));
    $newBalance = $loanAmount + $result['Balance'];
    mysqli_query($conn, "UPDATE customers SET Balance=$newBalance WHERE
AccountNo=$accountno");

    $notification = "You loan for Rs $loanAmount vide LoanAccountNo.
$loanNo is sanctioned. ";
    mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES
('loanSanctioned',$accountno,'$notification')");

    mysqli_query($conn, "INSERT INTO
`transactions`(`TransactionAmount`,`Receiver`,
`Actions`,`Remark`,`ReceiverBalance`) VALUES
('$loanAmount','$accountno','loan_sanctioned','$loanNo',$newBalance);");
}
function loanReject($loanNo, $accountno)
{
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    mysqli_query($conn, "DELETE FROM loanApp WHERE
LoanAccountNo=$loanNo");
    $notification = "You loan application vide LoanAccountNo. $loanNo is
rejected. ";
```

```
    mysqli_query($conn, "INSERT INTO `notifications`( `Actions`,
`AccountNo`, `Message`) VALUES
('loanRejected',$accountno,'$notification')");
}
```

CUSTOMER DETAILS UPDATE :

account_profile.php:

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Bank of Bhadrak</title>
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:
opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.1/css/all.min.css" integrity="sha512-
DTOQO9RWCH3ppGqcWaEA1BIZOC6xxalwEsw9c2QQeAIftl+Vegovlnee1c9QX4TctnWMn13TZ
ye+giMm8e2LwA==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <script src="./assets/app.js"></script>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet"
href="https://cdn.datatables.net/1.13.5/css/jquery.dataTables.min.css" />
    <script src="https://code.jquery.com/jquery-3.7.0.js"></script>
    <script
src="https://cdn.datatables.net/1.13.5/js/jquery.dataTables.min.js"></scr
ipt>
    <script>
        $(document).ready(function() {
            $('#myTable').DataTable();
        });
    </script>

    <?php
    require $_SERVER['DOCUMENT_ROOT'] ."/banking/assets/php/config.php";
    require_once $_SERVER['DOCUMENT_ROOT'] .
"/banking/assets/php/prac.php";
    session_start();
    if ($_SESSION["isLoggedin"] == false) {
        header('location:/banking/index.php');
```

```
    }
    $email = $_SESSION['email'];
    $get_employee = mysqli_query($conn, "SELECT * FROM employees,branch
WHERE employees.EmployeeEmail = '$email'");
    $emp = mysqli_fetch_assoc($get_employee);

    if (isset($_POST['search-account'])) {
        $accountno = $_POST['accountno'];
        $_SESSION['customerAccountNo'] = $accountno;
        $sql = mysqli_query($conn, "SELECT * FROM customers WHERE
AccountNo=$accountno;");
        $row = mysqli_fetch_assoc($sql);
        $transaction_Details = mysqli_query($conn, "SELECT * FROM
transactions WHERE Sender=$accountno OR Receiver=$accountno");
    } else {
        $accountno = $_SESSION['customerAccountNo'];
        $sql = mysqli_query($conn, "SELECT * FROM customers WHERE
AccountNo=$accountno;");
        $row = mysqli_fetch_assoc($sql);
        $transaction_Details = mysqli_query($conn, "SELECT * FROM
transactions WHERE Sender=$accountno OR Receiver=$accountno");
    }
    ?>
</head>

<body>
    <nav class="navbar">
        <div class="profile-nav">
            <div class="logo">
                <div>
                    <img src="/banking/images/logo.jpg" alt="">
                </div>
                <p>Bank of Bhadrak</p>
            </div>
            <div class="profile">
                <div class="branchIFSC">
                    BRANCH : <?php echo  $row['IFSC']; ?>
                </div>
                Welcome 👌 <?php echo strtoupper($emp['EmployeeName']);
?>
                <div class="dropdown">
                    <img src="/banking/images/clerk.jpg" alt="">
                    <div class="items">
                        <a
href="/banking/assets/php/logout.php">Logout</a>
                    </div>
                </div>
```

```
            </div>
        </div>
    </nav>
    <div class="bottom-nav">
        <div>
            <ul>
                <li><a href="clerk_dashboard.php">Search
Accounts</a></li>
                <li><a href="approveApp.php">Approve Application</a></li>
                <li><a href="fund_transfer.php">Fund Transfer</a></li>
                <li><a
href="deposit_withdraw.php">Deposit/Withdraw</a></li>
            </ul>
        </div>
        <div>
            <div class="tooltip">
                <a href=""> <button>
                        <i class="fa-solid fa-arrows-rotate"></i>
                    </button>
                </a>
            </div>
        </div>
    </div>
    <main class="profile-main">
        <section class="profile-gg">
            <div class="pic">
                <div style="flex-direction: column;">
                    <img src="/banking/images/<?php echo $row['img']; ?>"
class="profile-pic" alt="">
                    <h6 style="text-align: center; margin:20px;">Status :
<?php echo $row['Status'] ?></h6>
                </div>
                <div class="gg">
                    <form action="update_details.php" method="post"
style="flex-direction:row;">
                        <h5>Change Status :</h5>
                        <input type="submit" id="btn" value="Mark Active"
name="active">
                        <input type="submit" id="btn" value="Mark
Inactive" name="inactive">
                    </form>
                    <form action="update_details.php" method="post">
                        <input type="password" name="password"
id="password" placeholder="Create a Password" required>
                        <input type="submit" id="btn" value="Update
password" name="changepass">
                    </form>
```

```
                    <form action="update_details.php" method="post"
enctype="multipart/form-data">
                        <div class="field image">
                            <label>Select Image</label>
                            <input type="file" name="image"
accept="image/x-png,image/gif,image/jpeg,image/jpg" required>
                        </div>
                        <input type="submit" id="btn" value="Update
photo" name="updatepic">
                    </form>
                    <form action="update_details.php" method="post">
                        <input type="submit" id="btn" value="Delete
Account" name="deleteAccount">
                    </form>
                </div>
            </div>
            <div class="wrapper">
                <section class="form signup">
                    <header>Profile</header>
                    <form action="update_details.php" method="POST"
enctype="multipart/form-data" autocomplete="off">
                        <input type="number" name="accountno"
value="<?php echo $row['AccountNo']; ?>" hidden>
                        <div class="error-text"></div>
                        <div class="field input">
                            <label>IFSC</label>
                            <input type="text" name="ifsc"
placeholder="<?php echo $row['IFSC'] ?>" value="<?php echo $row['IFSC']
?>" required>
                        </div>
                        <div class="field input">
                            <label>Fullname</label>
                            <input type="text" name="fname"
placeholder="<?php echo $row['FullName'] ?>" value="<?php echo
$row['FullName'] ?>" required>
                        </div>
                        <div class="field input">
                            <label>Username</label>
                            <input type="text" name="uname"
placeholder="<?php echo $row['UserName'] ?>" value="<?php echo
$row['UserName'] ?>" required>
                        </div>
                        <div class="field input">
                            <label>Email Address</label>
                            <input type="text" name="email"
placeholder="<?php echo $row['Email'] ?>" value="<?php echo $row['Email']
?>" required>
```

```
                        </div>
                        <div class="field input">
                            <label>Phone no</label>
                            <input type="text" name="phoneno"
placeholder="<?php echo $row['PhoneNo'] ?>" value="<?php echo
$row['PhoneNo'] ?>" required>
                        </div>
                        <div class="field input">
                            <label>Gender</label>
                            <input type="text" name="gender"
placeholder="<?php echo $row['gender'] ?>" value="<?php echo
strtoupper($row['gender']) ?>" required>
                        </div>
                        <div class="field input">
                            <label>Address</label>
                            <input type="text" name="address"
placeholder="<?php echo $row['Address'] ?>" value="<?php echo
$row['Address'] ?>" required>
                        </div>
                        <div class="field input">
                            <label>District</label>
                            <input type="text" name="district"
placeholder="<?php echo $row['District'] ?>" value="<?php echo
$row['District'] ?>" required>
                        </div>
                        <div class="field input">
                            <label>State</label>
                            <input type="text" name="state"
placeholder="<?php echo $row['State'] ?>" value="<?php echo $row['State']
?>" required>
                        </div>
                        <div class="field input">
                            <label>Pincode</label>
                            <input type="text" name="pincode"
placeholder="<?php echo $row['Pincode'] ?>" value="<?php echo
$row['Pincode'] ?>" required>
                        </div>
                        <div class="field input">
                            <label>Aadhaar No</label>
                            <input type="text" name="aadhaar"
placeholder="<?php echo $row['AadhaarNo'] ?>" value="<?php echo
$row['AadhaarNo'] ?>" required>
                        </div>
                        <div class="field button">
                            <input type="submit" id="btn" name="submit"
value="Update">
                        </div>
```

```
                    </form>
                </section>
            </div>
        </section>
        <style>
            .transactions {
                display: flex;
                flex-direction: column;
                align-items: center;
                padding: 30px;
                min-height: 300px;
                gap: 25px;
            }

            .dataTables_filter {
                margin-bottom: 15px;
            }
        </style>
        <section class="transactions">
            <h3>TRANSACTIONS</h3>
            <br>
            <table id="myTable" class="styled-table">
                <thead>
                    <tr>
                        <th>Transaction Date</th>
                        <th>Description</th>
                        <th>Debit</th>
                        <th>Credit</th>
                        <th>Balance</th>
                    </tr>
                </thead>
                <tbody>
                    <?php
                    while ($fetch_transactions =
mysqli_fetch_assoc($transaction_Details)) {
                        $tdate = $fetch_transactions['TransactionDate'];
                        $tamount =
$fetch_transactions['TransactionAmount'];
                        $sender = $fetch_transactions['Sender'];
                        $receiver = $fetch_transactions['Receiver'];
                        $actions = $fetch_transactions['Actions'];
                        $remark=$fetch_transactions['Remark'];
                        if ($accountno == $receiver) {
                            if ($actions == 'loan_sanctioned') {
                                $debit = 0;
                                $credit = $tamount;
```

```
                                $balance =
$fetch_transactions['ReceiverBalance'];
                                $description = "LOAN SANCTIONED/CR/" .
$fetch_transactions['Receiver'];
                            } else if ($actions == 'fd_breaked') {
                                $debit = 0;
                                $credit = $tamount;
                                $balance =
$fetch_transactions['ReceiverBalance'];
                                $description = "FD BREAKED/CR/" .
$fetch_transactions['Receiver'];
                            } else if ($actions == 'int._credited') {
                                $debit = 0;
                                $credit = $tamount;
                                $balance =
$fetch_transactions['ReceiverBalance'];
                                $description = "INT. /CR/" .
$fetch_transactions['Receiver'];
                            } else if ($actions == 'deposit') {
                                $debit = 0;
                                $credit = $tamount;
                                $balance =
$fetch_transactions['ReceiverBalance'];
                                $description = "SELF/DEPOSIT/CR/" .
$fetch_transactions['Receiver'];
                            } else {
                                $debit = 0;
                                $credit = $tamount;
                                $balance =
$fetch_transactions['ReceiverBalance'];
                                $description = "P2A/E TRANSFER/CR/" .
$fetch_transactions['Sender'] . "/" . $remark;
                            }
                        } else {
                            if ($actions == 'fd_booked') {
                                $credit = 0;
                                $debit = $tamount;
                                $balance =
$fetch_transactions['SenderBalance'];
                                $description = "FD BOOKED/DR/" .
$fetch_transactions['Sender'];
                            } else if ($actions == 'loan_due_paid') {
                                $credit = 0;
                                $debit = $tamount;
                                $balance =
$fetch_transactions['SenderBalance'];
```

```
                                $description = "LOAN PAYMENT/DR/" .
$fetch_transactions['Sender'];
                            } else if ($actions == 'withdraw') {
                                $credit = 0;
                                $debit = $tamount;
                                $balance =
$fetch_transactions['SenderBalance'];
                                $description = "SELF/WITHDRAWAL/DR/" .
$fetch_transactions['Sender'];
                            } else {
                                $credit = 0;
                                $debit = $tamount;
                                $balance =
$fetch_transactions['SenderBalance'];
                                $description = "P2A/E TRANSFER/DR/" .
$fetch_transactions['Receiver'] . "/" . $remark;
                            }
                        }
                        echo "
                <tr>
                <td>" . substr($tdate, 0, 10) . "</td>
                <td>" . $description . "</td>
                <td>" . $debit . "</td>
                <td>" . $credit . "</td>
                <td>" . $balance . "</td>
                </tr>
                ";
                    } ?>
            </table>
        </section>
    </main>
    <footer>
        <p>&copy; 2024 Bank of Bhadrak. All rights reserved.
        </p>
    </footer>
</body>

</html>
```

6.4 Implementation:

The implementation phase involves integrating and deploying the developed codebase into a production environment, configuring system components, and conducting initial testing and validation. This section discusses the implementation process, including:

- Deployment Environment: The system is deployed in a production environment, utilizing cloud infrastructure providers such as AWS, Azure, or Google Cloud Platform to ensure scalability, availability, and reliability of the system.

- Configuration Management: Configuration files, environment variables, and deployment scripts are used to configure system parameters, dependencies, and runtime environments across different deployment environments, such as development, staging, and production.

- Continuous Integration and Deployment (CI/CD): CI/CD pipelines automate the build, testing, and deployment processes, enabling rapid and reliable delivery of code changes to production environments. Continuous integration tools such as Jenkins, Travis CI, or GitLab CI are employed to streamline code integration and deployment workflows.

By leveraging automated deployment pipelines and configuration management techniques, the implementation phase ensures seamless and consistent deployment of the online banking management system across different environments.

6.5 User Interface and User Experience (UI/UX):

The user interface and user experience design play a crucial role in the success and adoption of the online banking management system. This section discusses the UI/UX design considerations, including:

- Responsive Design: The system's user interface is designed to be responsive and adaptive, ensuring optimal user experience across various devices and screen sizes, including desktops, laptops, tablets, and smartphones.

- Intuitive Navigation: User-friendly navigation menus, breadcrumbs, and tooltips are incorporated to facilitate easy traversal and exploration of system features and functionalities, enhancing user engagement and satisfaction.

- Accessibility Features: Accessibility features such as keyboard shortcuts, screen reader compatibility, and color contrast adjustments are implemented to ensure inclusivity and compliance with accessibility standards, enabling users of all abilities to access and navigate the system effectively.

By prioritizing UI/UX design considerations, the implementation phase enhances the usability, accessibility, and satisfaction of the online banking management system, fostering positive user experiences and interactions.

6.6 Quality Assurance and Testing:

Quality assurance and testing are integral parts of the implementation process, ensuring the reliability, functionality, and performance of the online banking management system. This section discusses the testing methodologies, techniques, and tools employed, including:

- Unit Testing: Unit tests are conducted to validate individual components, functions, and modules of the system, verifying their behavior and functionality in isolation from other components. Testing frameworks such as JUnit, Mockito, and Jasmine are utilized to automate unit tests and streamline test execution.

- Integration Testing: Integration tests verify the interactions and interoperability between different system components, modules, and services, ensuring seamless integration and communication across the system. Integration testing frameworks such as Spring Test, Rest

Assured, and Protractor are employed to automate integration tests and validate end-to-end system workflows.

- User Acceptance Testing (UAT): UAT involves testing the system's functionality and usability from the perspective of end-users, stakeholders, and domain experts, ensuring alignment with user needs, expectations, and business requirements. UAT scenarios, test cases, and acceptance criteria are defined and executed to validate system behavior and gather feedback from stakeholders.

By conducting comprehensive quality assurance and testing activities, the implementation phase validates the correctness, reliability, and performance of the online banking management system, identifying and addressing defects and issues early in the development cycle.

7. Testing Methodologies and Quality Assurance

7.1 Introduction:

Chapter 7 focuses on the testing methodologies and quality assurance practices employed during the development and implementation of the online banking management system. This chapter elucidates the systematic approach to testing, including the types of testing, testing strategies, and tools utilized to ensure the reliability, functionality, and performance of the system. By prioritizing rigorous testing and quality assurance measures, stakeholders can mitigate risks, identify defects, and deliver a robust and high-quality system to end-users.

7.2 Testing Methodologies:

Testing methodologies delineate the systematic approach and techniques employed to verify and validate the functionality, usability, and performance of the online banking management system. This section discusses the following testing methodologies utilized in the testing process:

- Unit Testing: Unit testing focuses on validating individual components, functions, and modules of the system in isolation from other components. Developers write unit tests to verify the correctness of code logic, handling edge cases, and ensuring proper function behavior.

- Integration Testing: Integration testing verifies the interactions and interoperability between different system components, modules, and services. Integration tests validate data flow, communication protocols, and integration points to ensure seamless integration and end-to-end system functionality.

- System Testing: System testing evaluates the system as a whole, validating its compliance with specified requirements, business rules, and user expectations. System tests encompass

functional testing, usability testing, performance testing, security testing, and compatibility testing to ensure the overall reliability and quality of the system.

- User Acceptance Testing (UAT): UAT involves testing the system from the perspective of end-users, stakeholders, and domain experts to ensure that it meets their needs, expectations, and business requirements. UAT scenarios, test cases, and acceptance criteria are executed to validate system behavior and gather feedback for improvement.

By employing a diverse range of testing methodologies, the online banking management system undergoes comprehensive validation and verification, ensuring its readiness for deployment and use in production environments.

7.3 Testing Strategies:

Testing strategies outline the overarching approach and principles governing the testing process, including test planning, execution, and evaluation. This section discusses the following testing strategies employed in the testing process:

- Risk-Based Testing: Risk-based testing prioritizes testing efforts based on the likelihood and impact of potential risks and failures. High-risk areas of the system, such as critical functionalities, security vulnerabilities, and performance bottlenecks, receive greater testing emphasis to mitigate risks effectively.

- Regression Testing: Regression testing ensures that code changes and system enhancements do not introduce unintended side effects or regressions in existing functionality. Regression test suites are executed after each code change or system update to validate the continued correctness and stability of the system.

- Exploratory Testing: Exploratory testing involves ad-hoc, exploratory test sessions conducted by testers to uncover defects, usability issues, and corner cases that may not be

addressed by scripted test cases. Exploratory testing encourages creativity, intuition, and real-world usage scenarios to discover hidden defects and improve test coverage.

- Continuous Testing: Continuous testing integrates testing activities seamlessly into the software development lifecycle, automating test execution, and validation at every stage of the development process. Continuous integration (CI) and continuous deployment (CD) pipelines automate build, test, and deployment processes, enabling rapid feedback and iteration cycles.

By adopting robust testing strategies, the online banking management system undergoes thorough validation and verification, ensuring its reliability, functionality, and performance across different usage scenarios and environments.

7.4 Testing Tools:

Testing tools play a critical role in facilitating test automation, test management, and defect tracking throughout the testing process. This section discusses the following testing tools utilized in the testing process:

- Test Automation Frameworks: Test automation frameworks such as Selenium, Appium, and JUnit are employed to automate the execution of test cases, simulate user interactions, and validate system functionality across different browsers, devices, and platforms.

- Load Testing Tools: Load testing tools such as Apache JMeter, LoadRunner, and Gatling are utilized to simulate high loads, concurrent users, and stress conditions on the system, assessing its scalability, performance, and response times under varying workload scenarios.

- Test Management Platforms: Test management platforms such as TestRail, Zephyr, and PractiTest are used to plan, organize, and track testing activities, including test case management, test execution, and defect reporting. Test management platforms provide

visibility into testing progress, test coverage, and defect metrics, enabling effective test planning and execution.

- Defect Tracking Systems: Defect tracking systems such as JIRA, Bugzilla, and Redmine are employed to capture, prioritize, and manage software defects identified during testing. Defect tracking systems facilitate collaboration between development and testing teams, enabling timely resolution of issues and continuous improvement of the system.

By leveraging testing tools effectively, the testing team can streamline testing activities, improve test coverage, and ensure the timely identification and resolution of defects throughout the testing process.

7.5 Testing Execution:

Testing execution involves the systematic execution of test cases, scenarios, and scripts to validate the functionality, usability, and performance of the online banking management system. This section discusses the following aspects of testing execution:

- Test Case Execution: Test cases are executed according to predefined test plans, scenarios, and acceptance criteria, verifying the behavior and functionality of the system against specified requirements and user expectations.

- Test Result Analysis: Test results are analyzed to identify defects, failures, and deviations from expected behavior. Test reports, logs, and metrics are generated to provide insights into testing progress, test coverage, and defect trends, enabling stakeholders to make informed decisions and prioritize remediation efforts.

- Defect Management: Defects identified during testing are logged, prioritized, and tracked using defect tracking systems. Defects are categorized based on severity, impact, and priority, and assigned to development teams for resolution and validation. Defect management

processes ensure timely resolution of issues and continuous improvement of the system quality.

By executing tests systematically and rigorously, the testing team can validate the correctness, reliability, and performance of the online banking management system, ensuring its readiness for deployment and use in production environments.

8. Future Enhancements and Recommendations

8.1 Introduction:

Chapter 8 outlines the scope for future enhancements and provides recommendations for optimizing and expanding the functionality, usability, and performance of the online banking management system. This chapter explores creative and feasible avenues for improving the system's capabilities, addressing user feedback, and adapting to evolving technology trends and user needs. By identifying opportunities for enhancement and offering actionable recommendations, stakeholders can drive continuous improvement and innovation in the online banking management system.

8.2 Scope for Further Improvement:

The online banking management system presents several opportunities for further enhancement and refinement, including:

- Enhanced Security Measures: Strengthening security measures such as implementing multi-factor authentication, biometric authentication, and advanced encryption algorithms to safeguard user accounts, transactions, and sensitive data from unauthorized access and security breaches.

- Improved User Experience: Enhancing the user interface design, navigation flow, and accessibility features to provide a seamless and intuitive banking experience for users across different devices, browsers, and demographics.

- Advanced Personalization Features: Introducing advanced personalization features such as customizable dashboards, transaction categorization, spending insights, and personalized recommendations to empower users with tailored financial insights and recommendations.

- Integration with Emerging Technologies: Exploring integration opportunities with emerging technologies such as artificial intelligence (AI), machine learning (ML), blockchain, and Internet of Things (IoT) to enable innovative banking services, predictive analytics, and automated processes.

- Expansion of Banking Services: Expanding the range of banking services and offerings, including loan management, investment management, insurance services, bill payments, and peer-to-peer (P2P) transfers, to cater to diverse user needs and preferences.

By prioritizing these areas for improvement, the online banking management system can enhance its value proposition, user engagement, and competitive differentiation in the market, driving continued growth and success.

8.3 Recommendations:

Based on the analysis of current system capabilities and future opportunities, the following recommendations are proposed for optimizing and enhancing the online banking management system:

- Conduct User Feedback Surveys: Regularly solicit feedback from users, stakeholders, and domain experts through surveys, interviews, and usability testing sessions to gather insights into user preferences, pain points, and feature requests. Use this feedback to prioritize future development efforts and address user needs effectively.

- Implement Agile Development Practices: Adopt agile development methodologies such as Scrum or Kanban to promote iterative development, continuous feedback, and adaptive planning. Embrace cross-functional teams, short development cycles, and frequent releases to accelerate innovation and responsiveness to changing requirements.

- Foster Collaboration and Innovation: Foster a culture of collaboration, creativity, and innovation within the development team by encouraging knowledge sharing,

experimentation, and continuous learning. Establish innovation hubs, hackathons, and ideation workshops to spark new ideas and initiatives for enhancing the system.

- Monitor Technology Trends: Stay abreast of emerging technology trends, market developments, and regulatory changes in the banking industry to anticipate future opportunities and challenges. Leverage insights from industry reports, conferences, and thought leadership to inform strategic decision-making and technology investments.

By implementing these recommendations, the online banking management system can evolve into a dynamic, adaptive, and future-proof platform that delivers exceptional value and experiences for users and stakeholders alike.

9. References

References:

1. Sklar, D., & Trachtenberg, A. (2016). PHP Cookbook: Solutions & Examples for PHP Programmers. O'Reilly Media.

2. Lerdorf, R., Tatroe, K., & MacIntyre, P. (2017). Programming PHP: Creating Dynamic Web Pages. O'Reilly Media.

3. Ullman, L. (2018). PHP and MySQL for Dynamic Web Sites. Peachpit Press.

4. Welling, L., & Thomson, L. (2016). PHP and MySQL Web Development. Addison-Wesley Professional.

5. Dambacher, J., & Jones, E. (2019). PHP Objects, Patterns, and Practice. Apress.

6. Zandstra, M. (2016). PHP 7 Zend Certification Study Guide: Ace the ZCE 2017-PHP Exam. Apress.

7. Coggeshall, J., & Tadlock, J. (2016). PHP 7 Programming Cookbook. Packt Publishing.

8. Lockhart, M. (2016). Modern PHP: New Features and Good Practices. O'Reilly Media.

9. Yank, K. (2017). PHP & MySQL: Novice to Ninja. SitePoint.

10. Nixon, M. (2018). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media.

11. Akrabat, R. (2014). Understanding PHP: A Quickstart for PHP Developers. Leanpub.

12. Snyder, J. (2015). PHP Solutions: Dynamic Web Design Made Easy. Apress.

These references provide valuable insights, examples, and best practices for developing dynamic web applications using PHP. They cover topics ranging from basic PHP programming concepts to advanced techniques for building robust and secure web applications. Whether you're a novice PHP developer or an experienced programmer, these resources offer a wealth of knowledge to help you leverage PHP effectively in your projects.