# Finance Managment

MINOR PROJECT REPORT

By

**Vishishta Kavadiya -(RA2211026010564) Sahil Salunkhe– (RA2211026010538) Robin Mandal – (RA2211026010545)**

Under the guidance of

### Dr. Anousouya Devi

**ASSISTANT PROFESSOR , DEPARTMENT OF CINTEL**

*In partial fulfilment for the Course of*

### 21CSC203P – ADVANCED PROGRAMMING PRACTICE

in

### COMPUTER SCIENCE ENGINEERING

**with CSE CORE**



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING SRM INSTITUTE OF SCIENCE AND**

**TECHNOLOGY KATTANKULATHUR**

**NOVEMBER 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203PADVANCED PROGRAMMING PRACTICE** entitled in "**Finance Management** " is the bonafide work of **Vishishta Kavadiya -(RA2211026010564), Sahil Salunkhe– (RA2211026010538)** and **Robin Mandal – (RA2211026010545)** who carried out the work undermy supervision.

**SIGNATURE SIGNATURE**

**Dr. Anousouya Devi Dr. R. Annie Uthra**

**Assistant Professor HOD,Professor**

**CINTEL CINTEL**

SRMIST SRMIST

**Kattankulathur Kattankulathur**

## ABSTRACT

This project report presents a comprehensive overview of a Banking Management System developed using Python programming language. The system aims to revolutionize the banking industry by providing efficient and reliable services to customers, streamlining internal operations, and improving overall management of a bank. The objectives of this project are to develop a user-friendly interface for customers to perform banking transactions, implement a secure authentication system to protect customer data, provide features like account management, fund transfers, balance inquiries, and transaction history, automate internal processes such as account creation, interest calculation, and loan management, and generate reports for administrative purposes, such as customer analysis, transaction summaries, and financial statements. The methodology employed in this project follows a modular approach to ensure scalability and maintainability. Key modules incorporated in the system include a user interface module to design and develop an intuitive graphical user interface (GUI) for customers, an authentication module to implement secure login mechanisms using encryption techniques, an account management module to allow customers to create and manage their accounts, a transaction processing module to handle transactions securely, update account balances, and generate transaction history, and an administrative features module to provide functionalities such as user management, report generation, and system configuration. The expected outputs of the Banking Management System include a user-friendly interface for customers to perform banking transactions, secure authentication mechanisms to protect customer data, account creation, management, and transaction processing capabilities, automatic interest calculation and loan management, and generation of reports for administrative purposes. In conclusion, the Banking Management System developed using Python aims to improve banking services, enhance security, and automate internal processes. By implementing this system, banks can enhance customer experience, streamline operations, and improve overall efficiency, ultimately transforming the banking industry.

**ACKNOWLEDGEMENT**

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our

**Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr. Malarselvi G. , Assistant professor ,Department of Computing Technologies,** for her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. Pushpalatha M. , Professor, Department of Computing Technologies** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

# TABLE OF CONTENTS

**1**

**2**

**4**

**10**

**11**

**12**

**16**

**17**

|  |  |
| --- | --- |
| **S. NO.** | **CONTENTS** |
| **1** | **INTRODUCTION** |
| **2** | **LITERATURE**  **SURVEY** |
| **3** | **REQUIREMENT**  **ANALYSIS** |
| **4** | **FLOWCHART** |
| **5** | **IMPLEMENTATION** |
| **6** | **EXPERIMENT**  **RESULTS & ANALYSIS** |
| **7** | **CONCLUSION** |
| **8** | **REFERENCES** |

CHAPTER 1

INTRODUCTION

The Banking Management System is a crucial component of the modern banking industry, enabling efficient and secure management of various banking operations. This project report presents an overview of a Banking Management System developed using the Python programming language. The system aims to revolutionize the banking industry by providing enhanced services to customers, streamlining internal processes, and improving overall management of a bank. In today's fast-paced world, customers expect seamless and user-friendly banking experiences. The Banking Management System developed in Python aims to meet these expectations by offering a comprehensive and intuitive interface for customers to perform various banking transactions. From account management to fund transfers, balance inquiries, and transaction history, the system ensures that customers can conveniently and securely manage their financial activities. Security is a top priority in the banking sector, given the sensitive nature of customer data. Therefore, this system incorporates robust authentication mechanisms to protect customer information from unauthorized access. By implementing secure login mechanisms using encryption techniques, the system ensures that customer data remains confidential and protected against potential threats. Efficient internal operations are vital for the smooth functioning of any bank. The Banking Management System automates key processes such as account creation, interest calculation, and loan management. This automation not only reduces the risk of errors but also saves time and effort for bank personnel, enabling them to focus on more critical tasks. Moreover, the system generates reports for administrative purposes, such as customer analysis, transaction summaries, and financial statements, providing valuable insights for decision-making and regulatory compliance. The project follows a modular approach, ensuring scalability and maintainability. Different modules, including user interface, authentication, account management, transaction processing, and administrative features, are incorporated into the system to provide a comprehensive solution for banking management. By implementing the Banking Management System, banks can enhance customer experience, streamline operations, and improve overall efficiency. Customers will benefit from user-friendly interfaces, secure transactions, and prompt services, while banks will experience improved productivity, reduced manual errors, and increased customer satisfaction.

CHAPTER 2

LITERATURE SURVEY

1. Online Banking Systems: Explore existing online banking systems and their features. Analyze their user interfaces, security measures, transaction processing capabilities, and customer feedback.
2. Authentication Mechanisms: Investigate different authentication mechanisms used in banking systems, such as username and password, two-factor authentication, biometric authentication, and token-based authentication. Compare their strengths and weaknesses in terms of security and user experience.
3. Account Management: Study various approaches to account management in banking systems, including account creation, account types (savings, checking, etc.), account linking, and account closure procedures. Evaluate the effectiveness of these approaches in providing a seamless user experience.
4. Transaction Processing: Examine different methods of transaction processing, including fund transfers, bill payments, and balance inquiries. Analyze the efficiency, reliability, and security of these processes in existing banking systems.
5. Security Measures: Explore the security measures implemented in banking systems, such as encryption techniques, secure socket layer (SSL) protocols, and firewalls. Investigate the effectiveness of these measures in protecting customer data and preventing unauthorized access.
6. Automated Processes: Investigate automation in banking systems, particularly in areas like interest calculation, loan management, and account reconciliation. Analyze the impact of automation on accuracy, efficiency, and customer satisfaction.
7. Reporting and Analytics: Study the reporting and analytics capabilities of existing banking systems, including generation of customer reports, transaction summaries, and financial statements. Evaluate the usefulness of these reports for administrative purposes.
8. Mobile Banking: Explore the rise of mobile banking applications and their impact on the industry. Analyze the features, security measures, and user experience of these applications. Investigate the adoption rate and customer satisfaction levels associated with mobile banking
9. Regulatory Compliance: Investigate the regulatory requirements and compliance standards that banking systems must adhere to. Analyze how existing systems ensure compliance with regulations like Know Your Customer (KYC), Anti-Money Laundering (AML), and data protection laws.
10. Customer Experience: Study the importance of customer experience in banking systems and the strategies employed by existing systems to enhance customer satisfaction. Analyze user feedback, user interface design, ease of use, and customer support services.
11. Open Banking: Research the concept of open banking and its implications for the banking industry. Analyze the benefits, challenges, and security considerations associated with open banking systems.
12. Emerging Technologies: Explore the integration of emerging technologies like artificial intelligence (AI), machine learning (ML), and blockchain in banking systems. Investigate their potential to revolutionize banking operations, security, and customer experience.

CHAPTER 3

REQUIREMENTS

### User Requirements:

* 1. User-friendly interface
  2. Secure authentication mechanisms
  3. Easy account management (creation, linking, updates, closure)
  4. Convenient fund transfers
  5. Access to account balance information
  6. Transaction history tracking
  7. Bill payment options
  8. Loan management features (application, tracking, repayment)
  9. Strong security measures (encryption, SSL, firewalls) 10)Efficient customer support services

11)Reporting and analytics capabilities 12)Mobile accessibility for on-the-go banking

### Functional Requirements:

1. User Registration: The system should allow users to register by providing necessary information such as name, contact details, and identification documents.
2. Account Creation: Users should be able to create different types of accounts, such as savings, checking, or investment accounts, based on their requirements.
3. Login and Authentication: The system should provide a secure login mechanism, verifying user credentials (username and password) or using additional authentication methods like two-factor authentication or biometric authentication.
4. Account Management: Users should be able to view and update their account information, including personal details, contact information, and preferences.
5. Fund Transfers: Users should be able to transfer funds between their own accounts or to other accounts within the same bank. The system should validate the transaction and update the account balances accordingly.
6. Balance Inquiry: Users should have the ability to check their account balances, including available funds, pending transactions, and any accrued interest.
7. Transaction History: The system should maintain a comprehensive transaction history for each user, allowing them to view and search for past transactions, including details like dates, amounts, and transaction types.
8. Bill Payments: Users should be able to make payments for bills, such as utilities, loans, or credit card bills, by providing the necessary details and initiating the payment process.
9. Loan Application and Management: If the bank offers loan services, users should be able to apply for loans, track the status of their loan applications, and manage loan repayments, including viewing outstanding balances and payment schedules.
10. Account Closure: Users should have the option to close their accounts if they no longer require banking services. The system should facilitate a smooth account closure process while ensuring proper validation and security measures.
11. Security Measures: The system should implement stringent security measures to protect user data and transactions, including encryption techniques, secure communication protocols, and regular security updates.
12. Customer Support: The system should provide channels for users to seek customer support, such as live chat, email support, or a helpline, enabling them to get assistance for any issues or queries.
13. Reporting and Analytics: The system should generate reports for administrative purposes, such as customer analysis, transaction summaries, and financial statements, assisting in decision-making and regulatory compliance.
14. Mobile Accessibility: The system should have a mobile-friendly interface or a dedicated mobile application, allowing users to access their accounts and perform transactions conveniently from their smartphones or tablets.

### 3. Non-Functional Requirements:

1. Performance: The system should be highly responsive to user actions, with minimal latency, ensuring quick and efficient processing of transactions and account updates.
2. Security: The system should adhere to industry-standard security protocols and encryption techniques to safeguard user data and protect against unauthorized access or breaches.
3. Reliability: The system should be reliable and available for use at all times, with minimal downtime for maintenance or upgrades. It should have backup and recovery mechanisms to ensure data integrity and continuity of service.
4. Scalability: The system should be designed to handle a growing number of users and increasing transaction volumes without compromising performance or security. It should be able to scale up or down as needed.
5. Usability: The system should have an intuitive and user-friendly interface, allowing users to easily navigate, perform actions, and access information without the need for extensive training or technical expertise.
6. Accessibility: The system should be accessible to users with disabilities, adhering to accessibility guidelines and providing features such as text-to-speech, screen reader compatibility, and keyboard navigation options.
7. Compliance: The system should comply with relevant regulatory requirements, such as data protection laws, financial regulations, and privacy policies, ensuring the security and confidentiality of user information.
8. Interoperability: The system should be able to integrate with other banking systems or third-party services, such as payment gateways or credit bureaus, facilitating seamless data exchange and interoperability.
9. Maintainability: The system should be designed and developed using modular and well- documented code, making it easier to maintain, troubleshoot, and enhance in the future. 10.Performance Monitoring: The system should have performance monitoring and logging

capabilities, allowing administrators to track system performance, identify bottlenecks, and optimize system resources.

1. Data Backup and Recovery: The system should regularly back up user data and have mechanisms in place for disaster recovery, ensuring minimal data loss in case of hardware failures or system crashes.
2. Internationalization and Localization: The system should support multiple languages, currencies, and date/time formats to cater to users from different regions and comply with international banking standards

### 4. Technical Requirements:

1. Platform Compatibility: The system should be compatible with various platforms, including web browsers (Chrome, Firefox, Safari, etc.), operating systems (Windows, macOS, Linux, etc.), and mobile devices (iOS, Android).
2. Programming Language: The system should be developed using a suitable programming language, such as Java, Python, or .NET, considering factors like performance, scalability, and the availability of libraries and frameworks.
3. Database Management: The system should utilize a robust database management system (DBMS) to store and manage user data securely. Common options include MySQL, PostgreSQL, or Oracle.
4. Web Server: The system should run on a reliable and scalable web server, such as Apache HTTP Server or Nginx, to handle HTTP requests and serve web pages efficiently.
5. Security Measures: The system should implement appropriate security measures, including secure protocols (HTTPS), encryption algorithms (SSL/TLS), and secure coding practices to protect sensitive user data and prevent unauthorized access.
6. API Integration: The system may require integration with external APIs, such as payment gateways, credit bureaus, or third-party services, to facilitate seamless data exchange and provide extended functionalities.
7. Performance Optimization: The system should be optimized for performance to ensure fast response times and efficient resource utilization. Techniques like caching, database indexing, and load balancing may be employed.
8. Scalability and High Availability: The system should be designed to handle increasing user loads and ensure high availability. It may utilize technologies like load balancing, horizontal scaling, or cloud infrastructure to meet these requirements.
9. Mobile Application Development: If a mobile application is required, it should be developed using suitable frameworks like React Native, Flutter, or Xamarin, ensuring compatibility across different mobile platforms.
10. Testing and Quality Assurance: The system should undergo comprehensive testing, including unit testing, integration testing, and performance testing, to ensure functionality, reliability, and adherence to requirements.
11. Documentation: The system should be well-documented, including technical specifications, system architecture, APIs, and installation instructions, facilitating future maintenance, troubleshooting, and knowledge transfer.
12. Version Control: The system code should be managed using a version control system like Git, allowing for collaborative development, code branching, and easy tracking of changes.
13. Deployment and DevOps: The system should have a streamlined deployment process, leveraging DevOps practices, continuous integration, and automated deployment tools for efficient release management.
14. Monitoring and Logging: The system should have monitoring and logging mechanisms in place to track system health, performance metrics, and error logs, aiding in troubleshooting and performance optimization.

### 5. Legal and Regulatory Requirements:

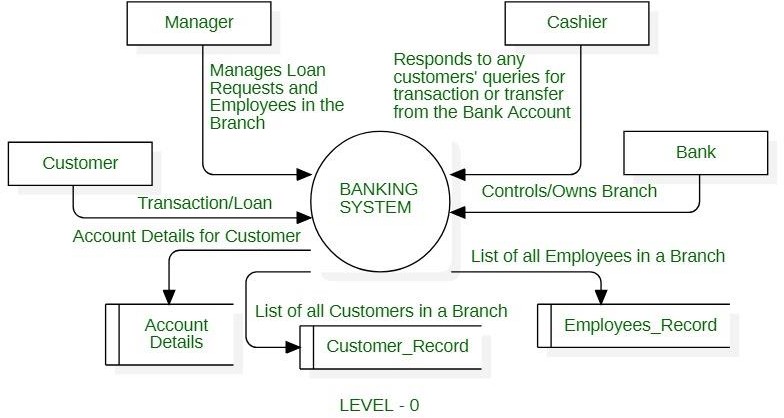
1. Privacy and Data Protection: The system must comply with applicable data protection laws and regulations, such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA). It should ensure the secure handling, storage, and transmission of customer data, provide options for data deletion or consent management, and obtain necessary user consent for data processing.
2. Know Your Customer (KYC) and Anti-Money Laundering (AML): The system should incorporate features to verify customer identities, perform due diligence checks, and monitor transactions for suspicious activities, complying with KYC and AML regulations. It should also retain customer identity and transaction records as required by law.
3. Financial Regulations: The system must adhere to relevant financial regulations, such as the Dodd-Frank Act, Basel III, or the Payment Services Directive (PSD2). It should support regulatory reporting, provide accurate financial data, and ensure compliance with capital adequacy, liquidity, and risk management requirements.
4. Consumer Protection: The system should comply with consumer protection laws and regulations, providing clear and transparent terms and conditions, disclosure of fees and charges, and fair treatment of customers. It should also handle customer complaints and dispute resolution processes in accordance with regulatory guidelines.
5. Electronic Transactions: The system must support electronic transactions and electronic signatures, complying with laws such as the Electronic Signatures in Global and National Commerce Act (ESIGN Act) or the eIDAS Regulation in the European Union. It should ensure the integrity, authenticity, and non-repudiation of electronic transactions.
6. Accessibility: The system should meet accessibility standards, such as the Web Content Accessibility Guidelines (WCAG), ensuring that individuals with disabilities can access and use the system without barriers. This includes providing alternative text for images, keyboard navigation options, and compatibility with assistive technologies.
7. Jurisdiction-specific Regulations: The system should comply with jurisdiction-specific banking and financial regulations, such as the Securities Exchange Commission (SEC) regulations in the United States or the Financial Conduct Authority (FCA) regulations in the United Kingdom. It should consider requirements related to licensing, reporting, and compliance specific to the targeted region.
8. Contractual Obligations: The system should adhere to contractual obligations between the bank and its customers or third-party service providers. This includes obligations related to service level agreements, confidentiality, data protection, and intellectual property rights.
9. Audit and Reporting: The system should support audit trails and logging mechanisms, enabling the tracking and recording of system activities for compliance audits and

regulatory reporting purposes.

1. Records Retention: The system should retain required records, such as customer transaction records, for the specified duration as mandated by regulatory requirements or legal obligation

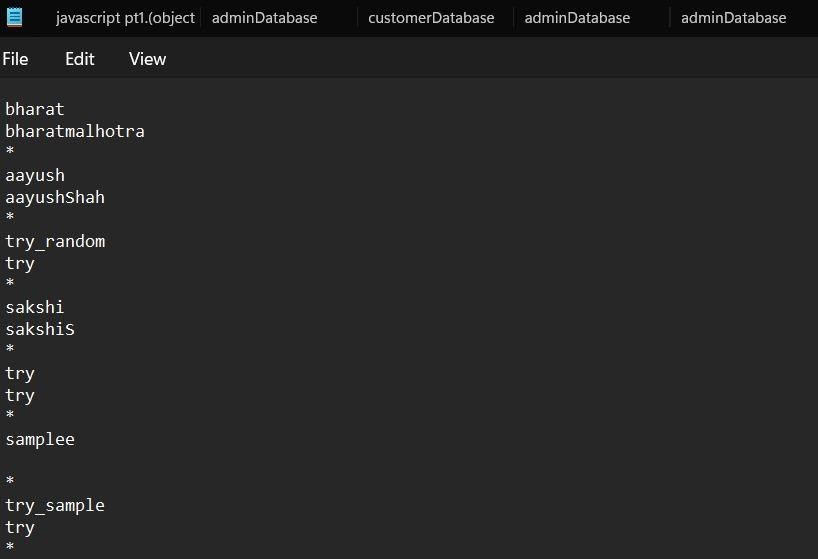
CHAPTER 4

Flowchart



CHAPTER 5

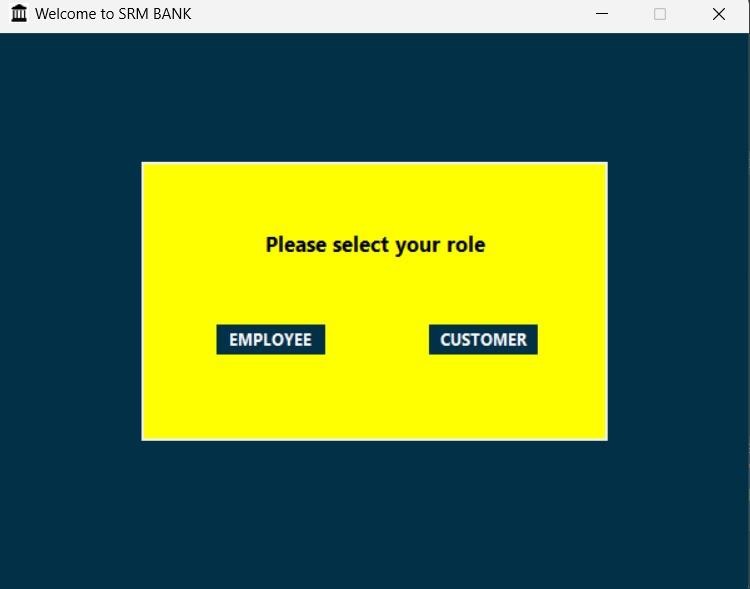
RESULTS AND OUTPUT

**Admin Database:**

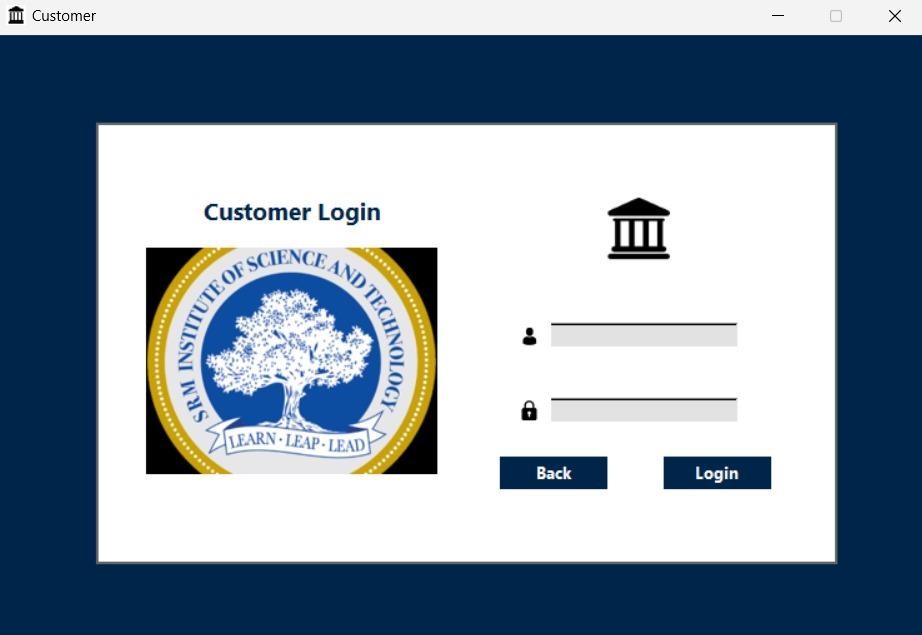
**Customer Database:**



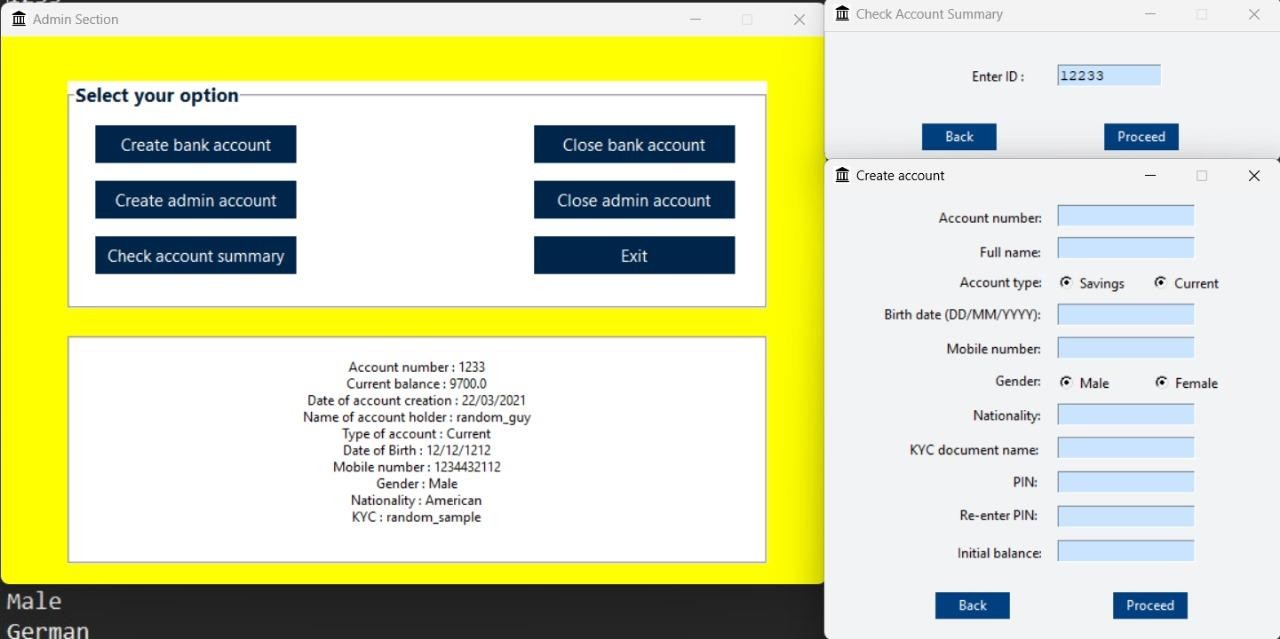
**Main Screen:**



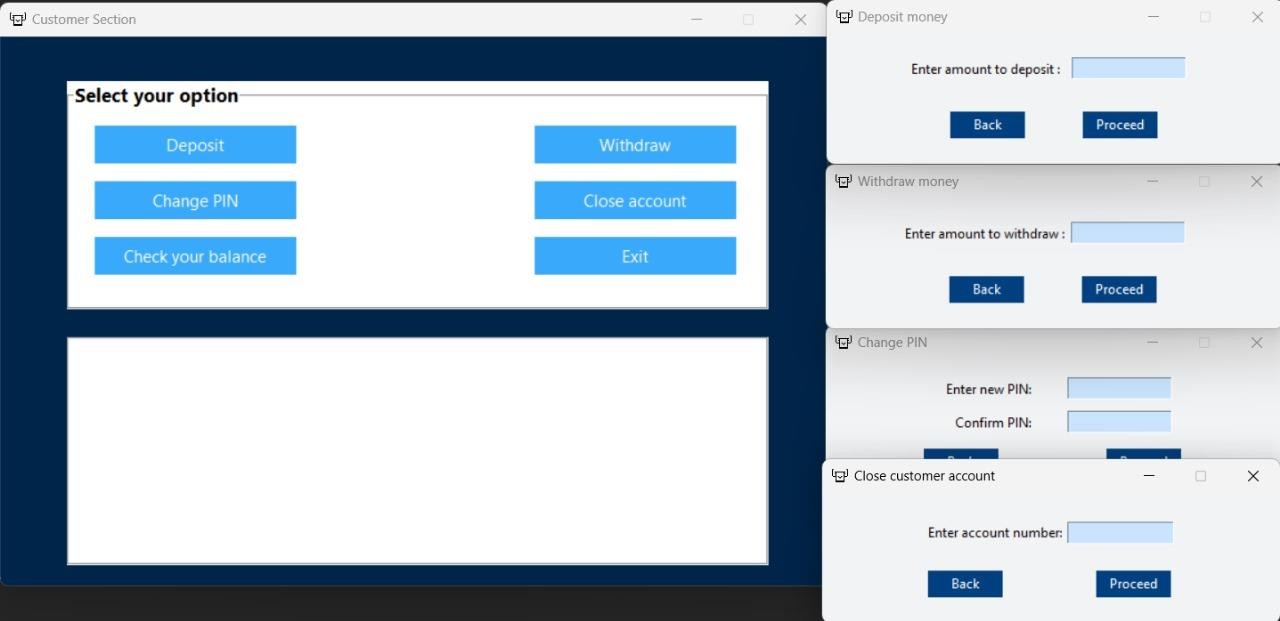
**Customer Login:**



**Control Panel for Admin Login:**



**Login Frame of Customer:**



**Account statement GUI customer:**



CHAPTER 6

CONCLUSION

In conclusion, developing a banking management system requires addressing both technical and legal/regulatory requirements. The technical requirements involve considerations such as platform compatibility, database management, security measures, scalability, and performance optimization. On the other hand, the legal and regulatory requirements encompass aspects like privacy and data protection, KYC/AML compliance, financial regulations, consumer protection, electronic transactions, accessibility, jurisdiction-specific regulations, contractual obligations, audit and reporting, and records retention.

By carefully addressing these requirements, a banking management system can be developed to ensure compatibility, security, compliance, and reliability. It is crucial to consult legal experts and stay updated with relevant laws and regulations to ensure full compliance and to protect the interests of both the bank and its customers.

CHAPTER 7

REFERENCES

[https://dev.mysql.com/doc/sakila/en/sakila-structure-tables-](https://dev.mysql.com/doc/sakila/en/sakila-structure-tables-address.html#%3A~%3Atext%3DThe%20address%20table%20contains%20address%2C%2C%20staff%20%2C%20and%20store%20tables) [address.html#:~:text=The%20address%20table%20contains%20address,%2C%20staff%20%2C%20and%20s](https://dev.mysql.com/doc/sakila/en/sakila-structure-tables-address.html#%3A~%3Atext%3DThe%20address%20table%20contains%20address%2C%2C%20staff%20%2C%20and%20store%20tables) [tore%20tables.](https://dev.mysql.com/doc/sakila/en/sakila-structure-tables-address.html#%3A~%3Atext%3DThe%20address%20table%20contains%20address%2C%2C%20staff%20%2C%20and%20store%20tables)

<https://bootcamp.uxdesign.cc/expense-manager-application-ui-ux-case-study-the-dignitas-a6714900c60f>