# SMALL BUSINESS NETWORK DESIGN WITH SECURE E-COMMERCE SERVER

MINOR PROJECT REPORT

By

**AMAN ANAND (RA2211003010130)**

**AADI TIWARI (RA2211003010127)**

**KRISHNA MAHAJAN (RA2211003010103)**

Under the guidance of   
**J. RamaPrabha Ma’am***In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in <**C.Tech**>



**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 **21CSC203P** **ADVANCED PROGRAMMING PRACTICE** entitled in "**Small Business Network Design with Secure E-commerce server**" is the bonafide work of **Aman Anand (RA2211003010130), Aadi Tiwari (RA2211003010127)** and **Krishna Mahajan (RA2211003010103)** who carried out the work under my supervision.

# SIGNATURE

<Faculty Name>

# <Designation>

**<Department>**

SRM Institute of Science and Technology

Kattankulathur

# ABSTRACT

The core of this project lies in the creation of an extensive Basic Bank System. This multifaceted system encompasses not only savings and checking accounts but also boasts a sophisticated and intuitive Graphical User Interface (GUI), meticulously crafted through the utilization of Java's versatile Swing library.

The project's primary objective is to provide a comprehensive and user-friendly platform for account management and transactions. This not only addresses the fundamental banking requirements of account holders but also emphasizes the need for a robust, seamless, and accessible banking solution.

In essence, this project serves as a testament to the power of Java's Swing library, showcasing its capability in designing efficient and user-centric banking systems that cater to a broad spectrum of financial needs and interactions, all while prioritizing the user experience and convenience.

# 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 **<Faculty Name> , <Designation> , <Department>,** for his/herassistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD <Name> <Designation>, <Department>** 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHAPTER NO** | | | **CONTENTS** | **PAGE NO** |
| **1** | | **INTRODUCTION** | |  |
|  | | 1.1 Motivation | |  |
|  | | 1.2 Objective | |  |
|  | | 1.3 Problem Statement | |  |
|  | | 1.4 Challenges | |  |
| **2** | | **PROJECT SCOPE** | |  |
| **3** | | **PROJECT STRUCTURE** | |  |
| **4** | | **USER INTERFACE** | |  |
| **5** | | **IMPLEMENTATION** | |  |
| **6** | | **RESULTS & DISCUSSUIONS** | |  |
| **7** | | **CONCLUSION** | |  |
|  |

1. **INTRODUCTION**

In the vast landscape of software development, Java programming emerges as a powerhouse for creating versatile and user-friendly applications. This endeavor, while intriguingly unnamed within the code itself, unfolds as a remarkable project that harmonizes the intricate world of coding with the practical needs of modern banking systems. The Java program at the core of this project is no ordinary piece of code; it ingeniously simulates a comprehensive basic bank system, complete with the complex functionalities of savings and checking accounts. A remarkable facet of this project is its elegant graphical user interface (GUI) development, a visual symphony crafted with the prowess of Java's dynamic Swing library.

This multifaceted code snippet enables users to embark on a journey within the realm of banking, where they can seamlessly create accounts, engage in a myriad of financial transactions, and vigilantly monitor their account balances. While the project's title may be discreetly veiled, it serves as a testament to the Java programming language's capacity to craft efficient and user-centric solutions for the financial world, prioritizing the intricacies of the user experience. The overarching project's name, if part of a larger initiative, would invariably find its place outside the confines of this code, leaving it as an intriguing glimpse into the fascinating realm of software development.

1. **PROJECT SCOPE**

The scope of this project encompasses the development of a Basic Bank System with a focus on providing users with two primary types of accounts: savings and checking. The project aims to offer a functional and user-friendly platform for account management and financial transactions.

Specifically, the project includes the following key components:

**1. BankAccount Class:**

- The creation of a core BankAccount class that serves as the foundation for both savings and checking accounts.

- Implementation of essential attributes and methods to manage account details, deposits, withdrawals, and balance tracking.

**2. SavingsAccount and CheckingAccount Classes:**

- Extension of the BankAccount class to create distinct savings and checking account classes.

- Inclusion of specialized functionalities, such as interest rate calculation for savings accounts and overdraft handling for checking accounts.

**3. Graphical User Interface (GUI):**

- Development of a graphical user interface (GUI) using Java's Swing library to enhance user interaction and facilitate account management.

- Inclusion of user-friendly components to create accounts, perform transactions, and check balances.

**4. Bank Class:**

- Implementation of the Bank class to manage accounts, allowing users to add accounts and retrieve account details.

**5. Banking Operations:**

- Provision of functionalities for account holders to deposit funds, withdraw funds, and check their account balances.

**6. Comprehensive Testing:**

- Rigorous testing of the project to ensure the accuracy and reliability of account management and transaction operations.

The project's primary goal is to provide users with an accessible and efficient platform for basic banking operations. While the project is comprehensive, it remains within the scope of fundamental account management and does not extend to more complex financial services. It lays the groundwork for potential future enhancements and serves as an educational tool to showcase the power of Java programming in creating practical, user-centric applications.

1. **PROJECT STRUCTURE**

The project is organized around a well-defined structure with an emphasis on modularity and object-oriented programming principles. Here is an overview of the project's structure:

**1. BankAccount Class:**

- The central class in the project, representing a generic bank account.

- Contains attributes for the account number, account holder's name, and balance.

- Provides methods for depositing and withdrawing funds, as well as checking the account balance.

- Offers a `toString` method for displaying account information.

**2. SavingsAccount Class:**

- An extension of the BankAccount class, representing a savings account.

- Inherits the attributes and methods of the BankAccount class.

- Introduces an interest rate attribute and the `applyInterest` method to calculate and apply interest to the account balance.

**3. CheckingAccount Class:**

- Another extension of the BankAccount class, representing a checking account.

- Inherits the attributes and methods of the BankAccount class.

- Includes an overdraft limit attribute and overrides the `withdraw` method to handle overdrafts.

**4. Bank Class:**

- Manages a collection of bank accounts using a HashMap data structure.

- Provides methods to add new accounts and find existing accounts using the account number as a key.

**5. Graphical User Interface (GUI):**

- Developed using Java's Swing library to create a user-friendly interface for the application.

- Includes buttons for creating savings and checking accounts, performing transactions, and exiting the application.

**6. Main Application:**

- The `**Advanced Bank System GUI** class serves as the main application class.

- Contains the `main` method that initializes the application and sets up the GUI components.

- Utilizes action listeners for the buttons to handle user interactions.

The project's structure reflects a clear separation of concerns, enabling the efficient management of various account types and transactions. This modular approach promotes code reusability and scalability, making it easier to extend the functionality of the banking system in the future. The use of object-oriented principles facilitates a clean and organized codebase, allowing for easier maintenance and enhancements.

1. **USER INTERFACE**

The User Interface (UI) of this project plays a pivotal role in ensuring a seamless and user-friendly experience for account holders. Developed using Java's Swing library, the GUI presents an intuitive and visually appealing platform for managing both savings and checking accounts. Here are the key elements of the project's graphical user interface:

**1. Frame Initialization:**

- The main frame, titled "Advanced Bank System," is created using the `JFrame` class.

- The frame's dimensions are set to 400 pixels in width and 300 pixels in height.

- A close operation is defined to exit the application when the user closes the frame.

**2. GUI Components:**

- Four buttons are included to facilitate user interactions:

- Create Savings Account Button: Allows users to create a new savings account.

- Create Checking Account Button: Enables the creation of a new checking account.

- Perform Transaction Button: Opens a menu for users to perform transactions.

- Exit Button: Closes the application when clicked.

**3. Layout Management:**

- The frame's layout is managed using a vertical `BoxLayout`, ensuring a neat and organized presentation of components.

- Each button is added to the frame, one below the other, to maintain a user-friendly interface.

**4. Action Listeners:**

- Action listeners are implemented for each button to respond to user actions.

- When a button is clicked, the associated action listener is triggered, allowing users to create accounts, perform transactions, or exit the application.

**5. User Input and Feedback:**

- User input is collected through various input dialog boxes to gather account details, initial balances, interest rates, and overdraft limits.

- Feedback to users, such as success messages and error notifications, is provided using `JOptionPane` dialog boxes, ensuring clear and concise communication.

The GUI's design and functionality are aimed at enhancing the user experience and simplifying account management. Users can easily create accounts, perform transactions, and obtain real-time account balance information through the intuitive graphical interface. The use of Java's Swing library provides a robust foundation for creating interactive and visually appealing applications, making it an essential component of this project.

1. **IMPLEMENTATION**

The project's implementation is a fusion of object-oriented programming and the integration of Java's Swing library to create a functional and user-centric Basic Bank System. Below are some key implementation details:

**1. BankAccount Class:**

- The `BankAccount` class serves as the foundation for all accounts in the system.

- It encapsulates attributes for the account number, account holder's name, and balance.

- The class includes methods for depositing and withdrawing funds, ensuring that account balances are updated accurately.

- Additionally, the `toString` method is implemented to provide a well-structured representation of account details.

**2. SavingsAccount and CheckingAccount Classes:**

- Both `SavingsAccount` and `CheckingAccount` classes extend the `BankAccount` class, inheriting its attributes and methods.

- The `SavingsAccount` class introduces an interest rate attribute and a method, `applyInterest`, to calculate and apply interest to the account balance.

- The `CheckingAccount` class includes an overdraft limit attribute and overrides the `withdraw` method to handle overdrafts, allowing account balances to go below zero by a specified limit.

**3. Bank Class:**

- The `Bank` class is responsible for managing and storing a collection of bank accounts.

- It uses a `HashMap` data structure, where account numbers serve as unique keys.

- The class provides methods to add new accounts and retrieve existing accounts based on their account numbers.

**4. Comprehensive Testing:**

- The project is rigorously tested to ensure the accuracy and reliability of account management and transaction operations.

- Test cases are designed to cover various scenarios, including deposits, withdrawals, interest calculations, overdraft handling, and account retrieval.

- Testing is conducted to verify the correctness of the implementation and to identify and rectify any issues or errors.

**5. Main Application:**

- The `AdvancedBankSystemGUI` class functions as the main application class.

- It contains the `main` method, which initializes the application and sets up the graphical user interface components.

- The class utilizes action listeners for the buttons, enabling the handling of user interactions.

The implementation of the project is guided by object-oriented principles, emphasizing code modularity, reusability, and maintainability. Each class fulfills a specific role within the system, resulting in a well-structured codebase. The project's comprehensive testing ensures that account management and transaction operations are robust and reliable, enhancing the user experience and functionality of the Basic Bank System.

1. **RESULTS AND DISCUSSION**

The implementation and testing of the Basic Bank System with savings and checking accounts, along with its intuitive graphical user interface (GUI), have yielded several notable results and discussions:

**1. Successful Implementation:** The project has been successfully implemented, demonstrating the effective creation and management of both savings and checking accounts. Users can create accounts, perform transactions, and obtain real-time balance information.

**2. Object-Oriented Approach:** The project showcases the power of an object-oriented approach by utilizing classes and inheritance. This design ensures code modularity, reusability, and maintainability.

**3. User-Friendly Interface:** The GUI development with Java's Swing library enhances the user experience. The graphical interface simplifies account operations, making the system accessible to a wide range of users.

**4. Account Varieties:** The introduction of two account types—savings and checking—offers users versatility in managing their financial affairs. The project's code structure allows for easy extension to include more account types.

**5. Error Handling:** Robust error handling mechanisms have been put in place, guaranteeing that users receive clear and informative messages when issues arise. For example, the system provides warnings for insufficient funds during withdrawals.

**6. Comprehensive Testing:** Extensive testing has been conducted to ensure the reliability and accuracy of account management and transaction operations. Test cases cover a variety of scenarios, verifying the project's correctness and robustness.

**Discussion:**

The project's successful implementation underscores the potential of Java programming in crafting practical and user-centric financial solutions. The object-oriented approach is instrumental in maintaining a clean and organized codebase, making it easy to manage and extend the system's functionality. The graphical user interface enhances the project's accessibility, providing users with an intuitive platform for managing their accounts.

In terms of future developments, there are opportunities to further enrich the system. Advanced banking features, enhanced security measures, and improved user interfaces could be considered to meet the evolving needs of users.

In conclusion, the Basic Bank System is an achievement that combines functionality, educational value, and user-friendliness. It not only provides users with a reliable platform for managing their accounts but also serves as an educational resource, inspiring further exploration and innovation in the field of digital banking solutions. The project's results and discussions emphasize the significance of a well-structured design and the potential of Java programming in creating practical applications.

1. **CONCLUSION**

The development of the Basic Bank System with savings and checking accounts, coupled with a user-friendly graphical user interface (GUI) in Java's Swing library, marks the culmination of an innovative and educational project. This project has demonstrated the potential of Java programming in creating efficient and intuitive solutions for the financial domain. In conclusion, several key takeaways and achievements emerge:

**1. Modular Design:** The project's modular design and implementation follow best practices in object-oriented programming. The use of classes and inheritance ensures code reusability and scalability.

**2. User-Centric Approach:** The integration of a graphical user interface (GUI) enhances the user experience, making account creation, transactions, and balance monitoring accessible and straightforward.

**3. Account Variety:** The project introduces two distinct account types—savings and checking—enabling users to explore different financial functionalities.

**4. Error Handling:** The system incorporates robust error handling mechanisms, providing clear and informative notifications to users when issues arise, such as insufficient funds for withdrawals.

**5. Comprehensive Testing:** Rigorous testing has been conducted to validate the correctness and reliability of account management and transaction operations, ensuring the system's integrity.

**6. Educational Value:** Beyond its functionality, the project serves as a valuable educational tool, showcasing the power of Java's Swing library for GUI development and the principles of object-oriented programming.

**7. Future Enhancements:** While the project provides a solid foundation, there are opportunities for future enhancements, such as the addition of more advanced banking features, enhanced security measures, and improved user interfaces.

In essence, this project underscores the versatility of Java programming in developing practical and user-centric applications. It has illustrated the potential of object-oriented design and highlighted the significance of a well-structured codebase. As an educational endeavor, it invites further exploration and innovation in the realm of digital banking solutions. The Basic Bank System encapsulates a rich blend of functionality, user-friendliness, and educational value, positioning it as a noteworthy accomplishment in the world of software development.