Karnatak Law Society’s

# GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008 Karnataka, India.



A Course Project Report on

## Banking System using RMI API

Submitted for the requirements of 5th semester B.E. in ISE

for **“Advanced Java”**

**(Professional Elective - 21IS544)**

## Submitted by

**NAME USN**

**1) Ayush Anand 2GI21IS013**

**2) Nandeesh Hiremath 2GI21IS035**

**3) Raghupraveer Nippani 2GI21IS065**

## Under the guidance of

**Dr. Padma Dandannavar**

**Associate Professor, Dept. of ISE**

**Academic Year 2023-2024 (Odd semester)**

Karnatak Law Society’s

# GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008 Karnataka, India.

**Department of Information Science and Engineering**



# **Certificate**

This is to certify that the Course Project work titled **“Banking System using RMI API”**

carried out by **Ayush Anand, Nandeesh Hiremath, Raghupraveer Nippani** bearing **USNs: 2GI21IS013, 2GI21IS035, 2GI21IS065** for **Advanced Java (21IS544)** **Integrated Course** is submitted in partial fulfilment of the requirements for 5th semester B.E. in **INFORMATION SCIENCE AND ENGINEERING,** Visvesvaraya Technological University, Belagavi. It is certified that all corrections/ suggestions indicated have been incorporated in the report. The course project report has been approved as it satisfies the academic requirements prescribed for the said degree.

|  |  |
| --- | --- |
| Date: 27-01-2024 |  |
| Place: Belagavi | Dr. Padma Dandannavar |

Asso., Prof., Dept. of ISE

KLS Gogte Institute of Technology, Belagavi

Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

**Academic Year 2023-24 (Odd Semester)**

**Semester: V**

**Course: Advanced Java (21IS544)**

**Rubrics for evaluation of Course Project Phase 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl.**  **No** | **Project**  **Component** | **Max. Marks** | **Marks Earned** | | |
| **2GI21IS013** | **2GI21IS035** | **2GI21IS065** |
| Ayush Anand | Nandeesh Hiremath | Raghupraveer Nippani |
| 1 | Understanding of the problem | 05 |  |  |  |
| 2 | Problem  Statement and  Objectives of the  Project | 05 |  |  |  |
| 3 | Proposed  Methodology | 05 |  |  |  |
| 4 | Presentation and Viva-voce | 10 |  |  |  |
|  | **Total** | **25** |  |  |  |

Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

**Academic Year 2023-24 (Odd Semester)**

**Semester: V**

**Course: Course: Advanced Java (21IS544)**

**Rubrics for evaluation of Course Project Phase 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.**  **No** | **Project**  **Component** | **Max. Marks** | **Marks Earned** | | |
| **2GI21IS013** | **2GI21IS035** | **2GI21IS065** |
| Ayush Anand | Nandeesh Hiremath | Raghupraveer Nippani |
| 1 | Design | 05 |  |  |  |
| 2 | Implementation | 05 |  |  |  |
| 3 | Results | 03 |  |  |  |
| 4 | Teamwork | 02 |  |  |  |
| 5 | Project  Demonstration and Viva-voce | 10 |  |  |  |
|  | **Total** | **25** |  |  |  |

**ACKNOWLEDGEMENT**

We express our heartfelt gratitude to the individuals and resources that played a pivotal role in the successful completion of our project, "Design and Implementation of a Distributed Banking System Utilizing Remote Method Invocation (RMI) Technology." This endeavor was made possible through the collaborative efforts and support of various contributors. Our sincere thanks go to our project guide, Dr. Padma Dandannavar, Associate Professor, (Dept. of ISE) for their unwavering guidance, expertise, and encouragement throughout the development process. Their insightful feedback and mentorship were instrumental in shaping the architecture and functionality of our distributed banking system.

We extend our appreciation to our fellow students and colleagues who actively participated in discussions, providing valuable insights that enriched the project's design and implementation. The exchange of ideas and collaborative spirit fostered a dynamic environment, enhancing the overall quality of our work. The foundation of our understanding of distributed systems and RMI was built upon the works of several authors and contributors. We acknowledge their valuable literature and resources that served as a bedrock for our exploration of advanced concepts in banking system development.

We express our gratitude to our friends and family for their unwavering support and understanding during the project's duration. Their encouragement provided the motivation needed to navigate complexities and overcome obstacles. This project has been a collective effort, and each contributor, whether directly or indirectly, has played a vital role in its successful execution. As we present this report, we recognize and appreciate the collaborative spirit that fueled the development of our distributed banking system.

**ABSTRACT**

This project presents the design and implementation of a distributed banking system utilizing Remote Method Invocation (RMI) technology. The objective is to create a scalable and secure architecture where clients can perform banking operations such as deposit, withdrawal, and balance inquiry through interactions with a central server. The application adheres to a client-server model, allowing users to access banking services over a network.

The project employs Java RMI for remote communication, enabling seamless invocation of methods on the server from client applications. The client-server architecture ensures separation of concerns, with the server managing account-related operations and the client interacting with the user interface. Key features include account creation, deposit, withdrawal, and balance inquiry.

The report outlines the system's architecture, detailing the design choices made to achieve distributed functionality. It covers the implementation of the server-side logic for account management and the client-side application for user interaction. Emphasis is placed on the use of synchronization for concurrent access to shared resources and error handling for robustness.

Throughout the project, considerations for security, reliability, and scalability are addressed. The application's extensibility is demonstrated through the ability to handle multiple client connections concurrently. Challenges faced and solutions devised during the development process are discussed.

In conclusion, this distributed banking system provides a foundation for secure and scalable financial transactions over a network, showcasing the effectiveness of Java RMI in facilitating remote communication between client and server components. The report encapsulates the project's goals, methodologies, and outcomes, offering insights into the design and implementation of the distributed banking application.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Content** | **Page No.** |
| 1 | Title of the project | 8 |
| 2 | Statement of the Problem | 8 |
| 3 | Objectives and scope of the project | 8 |
| 4 | Tools/Framework used | 9 |
| 5 | Methodology / Design | 10 |
| 6 | Algorithm | 11-12 |
| 7 | Screenshots | 13 -14 |
| 8 | Applications and Future Enhancement of the Project | 15 |
| 9 | Conclusion | 16 |
| 10 | References | 16 |

**Title of Project: Banking System using RMI**

**Problem Statement:**

Distributed Banking System: Develop a basic banking system where clients can perform banking operations like deposit, withdrawal, and balance inquiry through RMI on a central serve.

**Objectives and Scope:**

The primary objectives of Banking System are as follows:

1. Client-Server Architecture Implementation:

- Develop a robust client-server architecture to facilitate communication between multiple clients and a centralized server.

2.RMI Integration:

- Incorporate Java RMI technology to enable seamless remote method invocation, allowing clients to perform banking operations on the server.

3. Account Management Functionality:

- Implement essential banking functionalities on the server, including account creation, deposit, withdrawal, and balance inquiries.

The scope of the distributed banking system project extends to various areas where secure, scalable, and remote banking services are required. This includes, but is not limited to:

1. Retail Banking: Providing banking services to individual customers, including account management, deposits, withdrawals, and balance inquiries.

2. Corporate Banking: Serving the banking needs of businesses, including managing company accounts, processing transactions, and accessing financial data remotely.

3. Online Banking Platforms: Integration with online banking platforms to enable customers to access their accounts and perform transactions over the internet securely.

**Tools and Frameworks:**

**Tools:**

1. Java Development Kit (JDK): Provides the necessary tools and libraries for Java development.

2. Integrated Development Environment (IDE):

- Examples include IntelliJ IDEA, Eclipse, or NetBeans for coding, debugging, and project management.

3. Build Automation Tool:

- Maven or Gradle for managing project dependencies, compiling code, and generating executable artifacts.

**Frameworks:**

1. Java Remote Method Invocation (RMI):

- Enables remote communication between client and server components in Java applications.

2. JUnit: Testing framework for unit testing to ensure the correctness and reliability of individual components.

3. Java Cryptography Extension (JCE):

- Provides cryptographic functionality, such as encryption and digital signatures, for implementing security features.

**Methodology:**

**1. System Design:**

* Define System Architecture:
  + Establish a client-server architecture utilizing Java RMI.
* Design Interfaces:
  + Create the BankingInterface defining remote methods for banking operations.
* Identify Classes:
  + Determine classes like BankAccount, BankingClient, and BankingServer.

**2. Implementation:**

* Create BankingInterface:
  + Implement the remote interface with methods like addAccount, deposit, withdraw, and getBalance.
* Develop BankAccount Class:
  + Code the BankAccount class with methods for deposit, withdrawal, and balance inquiry.
* Build BankingServer:
  + Implement the server logic for account management and remote methods.
* Code BankingClient:
  + Develop the client application with a user interface for interaction.

**3. Concurrency Control:**

* Implement Synchronization:
  + Ensure thread safety with synchronization mechanisms to handle concurrent access.

**4. Testing:**

* Unit Testing:
  + Employ JUnit or TestNG for unit tests to verify the correctness of individual components.

**Algorithm:**

**Server-Side Algorithm (BankingServer):**

Initialization:

1. Start the RMI registry on a specific port.

2. Create an instance of the `BankingServerImpl` class.

3. Bind the `BankingServerImpl` instance to a specific RMI URL in the registry.

Account Management:

1. Define methods for account management operations (addAccount, deposit, withdraw, getBalance).

2. Implement these methods in the `BankingServerImpl` class.

3. Ensure thread safety by using synchronization mechanisms.

4. Maintain a collection of bank accounts (e.g., HashMap) to store account information.

Handling Client Requests:

1. Listen for client requests via RMI.

2. Upon receiving a request, invoke the corresponding method on the `BankingServerImpl` instance.

3. Process the request, perform the required banking operation, and return the result to the client.

**Client-Side Algorithm (BankingClient):**

Initialization:

1. Look up the `BankingServer` object in the RMI registry using its URL.

2. Obtain a reference to the remote banking interface (`BankingInterface`).

User Interaction:

1. Display a menu to the user with options for banking operations (e.g., create account, deposit, withdraw, check balance).

2. Prompt the user to select an option and provide necessary inputs (e.g., account number, amount).

3. Invoke the corresponding method on the remote banking interface based on the user's selection.

Handling Server Responses:

1. Receive the response from the server after invoking a remote method.

2. Display the result of the operation to the user (e.g., success message, error message, account balance).

**Algorithm for BankingInterface and BankAccount:**

BankingInterface:

1. Define a remote interface (`BankingInterface`) with methods for banking operations (addAccount, deposit, withdraw, getBalance).

2. Ensure that each method throws `RemoteException` to handle remote invocation exceptions.

BankAccount:

1. Define a `BankAccount` class representing a bank account with attributes (e.g., account number, balance).

2. Implement methods for deposit, withdrawal, and balance inquiry.

3. Use synchronization to ensure thread safety when modifying the account balance.

**Screen Shots, Input text, output text :**

Input text :

Enter account number:

728

Enter initial balance to deposit:

200

Account with initial deposit created.

Menu:

1. Deposit

2. Withdraw

3. Balance Inquiry

4. Exit

Enter your choice:

1

Enter deposit amount:

200

Deposit completed.

Menu:

1. Deposit

2. Withdraw

3. Balance Inquiry

4. Exit

Enter your choice:

2

Enter withdrawal amount:

200

Withdrawal completed.

Menu:

1. Deposit

2. Withdraw

3. Balance Inquiry

4. Exit

Enter your choice:

3

Current balance for account 728: $200.0

Menu:

1. Deposit

2. Withdraw

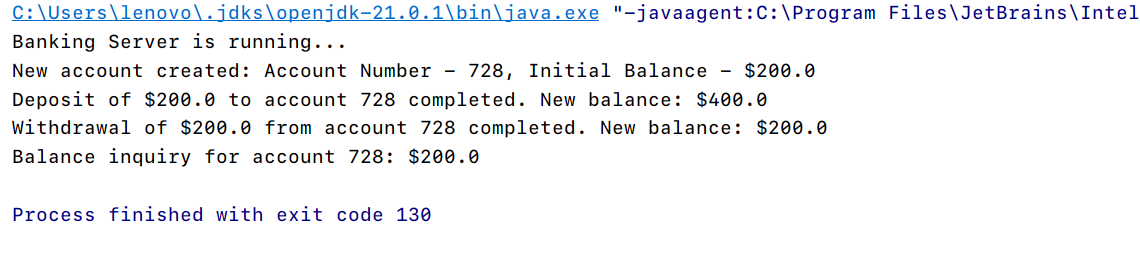
3. Balance Inquiry

4. Exit

Enter your choice:

4

Exiting...

**Output text:**

**Applications:**

Retail Banking Platforms: Deployment within banks or financial institutions to offer banking services to individual customers, including account management and transactions.

Corporate Banking Solutions: Integration into corporate banking systems for managing company accounts, processing payments, and accessing financial data.

Online Banking Portals: Incorporation into online banking platforms to enable customers to access their accounts, make transactions, and manage finances over the internet.

Mobile Banking Apps: Integration with mobile banking applications, allowing customers to perform banking activities using their smartphones or tablets

**Future Enhancements:**

Transaction History: Addition of functionality to view transaction history, including details of deposits, withdrawals, and transfers.

Account Management Tools: Development of tools for managing multiple accounts, setting up automatic transfers, and configuring account preferences.

Budgeting and Financial Planning: Integration of budgeting and financial planning features to help users track expenses, set savings goals, and manage finances effectively.

Security Enhancements: Implementation of advanced security measures such as multi-factor authentication, biometric authentication, and fraud detection algorithms to enhance account security.

**Conclusion:**

In conclusion, the development of the Distributed Banking System has successfully demonstrated the implementation of a robust and scalable financial application using Remote Method Invocation (RMI) technology. The project addressed the specific objectives of creating a secure, user-friendly, and extensible banking system, emphasizing the principles of distributed computing. By incorporating core functionalities such as deposit, withdrawal, and balance inquiry, the system not only enhances accessibility and convenience for clients but also serves as a valuable educational resource for understanding distributed systems and RMI. While the current implementation showcases basic functionality, future iterations can focus on further enhancing security measures and exploring integration possibilities with external financial systems, solidifying the project's foundational role in the dynamic landscape of distributed banking solutions.

**Reference:**

* <https://www.javatpoint.com/RMI>
* <https://www.geeksforgeeks.org/remote-method-invocation-in-java/>
* Oracle. (n.d.). Java Remote Method Invocation (RMI). Retrieved from <https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/>
* "Java: The Complete Reference" by Herbert Schildt