Banking Application in Java (Console Based Application)

Document Version Control

Date Issued	Version	Description	Author
9/5/23	1	Initial HLD — V1.0	Gagan, gaurav

Contents

Document Version Control. 2
Abstract. 4
1 Introduction 5
1.1 Why this High-Level Design Document? 5
1.2 Scope. 5
2 General Description. 6
2.1 Product Perspective 6
2.2 Problem statement 6
2.3 FURTHER IMPROVEMENTS 6
2.4 Technical Requirements. 6
2.5 Tools used. 7
3 Design Details 7
3.1 Process Flow. 7
3.2 Error Handling 8
3.3 Performance. 8
3.4 Reusability. 8
4 Conclusion. 8
5 references 9

Abstract

This console-based Java banking application is designed to provide basic banking functionalities such as account creation, deposit and withdrawal transactions, balance inquiry, and transaction history to users. The application has been built with object-oriented principles, using various Java programming concepts such as inheritance, polymorphism, and encapsulation. Users can create accounts with various types such as savings and checking, each with its specific interest rates and transaction fees. The application also includes basic security features such as user authentication and authorization to ensure data privacy and security. The application uses a simple and intuitive command-line interface, allowing users to interact with the application seamlessly. The application stores user data and transaction history in a local database, ensuring that data is persistent and readily available for future use.In summary, this Java-based banking application provides a straightforward, secure, and efficient way for users to manage their bank accounts.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	*Description*
IDE	Integrated Development Environment

2 General Description

2.1 Product Perspective

This console-based Java banking application is designed to provide basic banking functionalities such as account creation, deposit and withdrawal transactions, balance inquiry, and transaction history to users. The application has been built with object-oriented principles, using various Java programming concepts such as inheritance, polymorphism, and encapsulation.

2.2 Problem statement

Take input from the user using Scanner class, basics about string, how to print in java, variables, if/else statements, methods, loops, etc. In simple banking application will be coding the simple bank operations like check balance, deposit, withdraw, and exit..

2.3 FURTHER IMPROVEMENTS

The banking application it can be added with the GUI integration, multiuser support, transaction notification, payment gateway, Accessibility Features, Analytics and Reporting. Overall, these improvements can help enhance the functionality and usability of the console-based Java banking application, making it more appealing and engaging to users.

2.4 Technical Requirements

Technical requirements for a console-based Java banking application:

2.4.1 Programming Language: The application must be developed using Java, as it is a popular and widely used programming language that is well-suited for building enterprise-level applications.

2.4.2 Object-Oriented Design: The application must be designed using object-oriented principles such as inheritance, encapsulation, and polymorphism. This design approach can help improve the application's maintainability and extensibility.

2.4.3 Security Features: The application must include security features such as user authentication and authorization to ensure data privacy and prevent unauthorized access to user accounts.

2.4.4 Command-Line Interface: The application must have a simple and intuitive command-line interface that allows users to interact with the application seamlessly.

2.4.5 Exception Handling: The application must handle exceptions and errors

appropriately to ensure that the application runs smoothly and to prevent data loss or corruption.

2.4.6 Code Quality: The application code must follow best practices such as proper naming conventions, comments, and modularization to improve code readability, maintainability, and extensibility.

2.4.7 Testing: The application must be thoroughly tested to ensure that it meets the functional and non-functional requirements and is free of bugs and errors.

2.4.8 Scalability: The application must be designed to handle a large number of users and transactions, and it must be scalable to accommodate future growth and changes.

2.5 Tools used

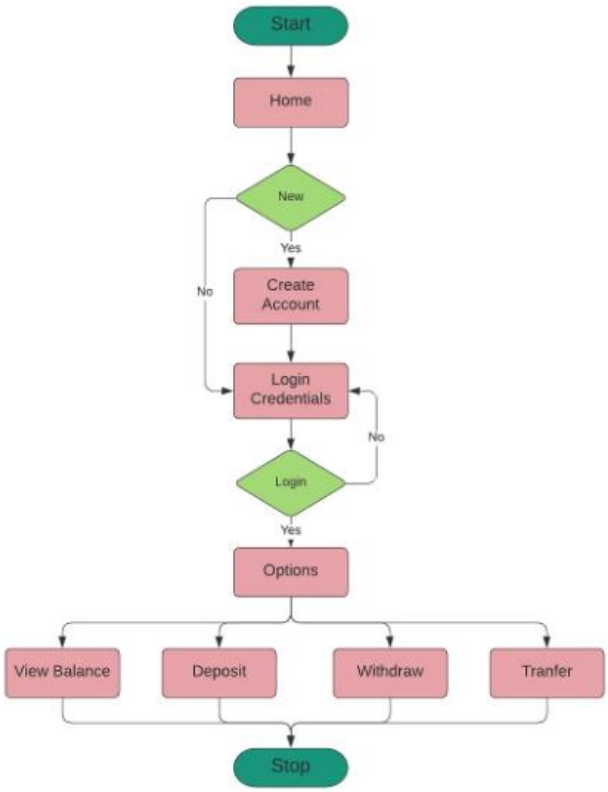
2.5.1 Eclipse is used as IDE.

2.5.2 Java programming language is used for coding the application.

2.5.3 GitHub is used as version control system.

3 Design Details

3.1 Process Flow



3.2 Error Handling

Error handling in a console-based Java banking application involves anticipating and handling potential errors that may occur during user interactions with the application. Some common errors include invalid user input, insufficient funds, account not found, database connection errors, and system errors. To handle these errors, you can use exception handling techniques such as try-catch blocks, throw statements, and custom exception classes. Proper error handling can improve the usability and reliability of the application for users.

3.3 Performance

The performance considerations for a console-based Java banking application:

- Speed: The application should complete operations quickly using efficient algorithms and optimized code.
- Memory usage: The application should use memory efficiently to prevent slowdowns or crashes on low-end machines.
- Scalability: The application should be designed to handle increasing loads without slowing down or crashing.
- Responsiveness: The application should feel fast and snappy to users, even if certain operations take longer to complete in the background.
- Error handling: The application should anticipate potential errors and handle them gracefully to prevent slowdowns or crashes.

By addressing these performance considerations, developers can create a fast, efficient, and reliable console-based Java banking application that meets the needs of users.

3.4 Reusability

The code written and the components used should have the ability to be reused with no problem.

4 Conclusion

A console-based Java banking application provides a simple, efficient, and secure way for users to manage their bank accounts. The application includes basic functionalities such as account creation, deposit and withdrawal transactions, balance inquiry, and transaction history. The application has been built using various object-oriented principles, including inheritance, polymorphism, and encapsulation, to ensure that it is maintainable and extensible. The application also includes security features such as user authentication and authorization to ensure data privacy and security. While a console-based interface may not be as visually appealing as a graphical user interface, it provides a straightforward and intuitive way for users to interact with the application. Moreover, the application can be extended to include more advanced functionalities

such as multi-user support, payment gateway integration, and investment options. Overall, a console-based Java banking application provides a robust and reliable solution for users to manage their bank accounts. It can be easily customized and scaled to meet the needs of different users and financial institutions.

5 References

1. https://www.java.com/en/.
2. https://www.paymentsjournal.com/the-essential-features-of-a-banking-app/