Low Level Design (LLD)

For

Console Based Banking Application

Revision Number: 1.5
Last date of revision: 09/07/2023

iNeuron .ai

- Sukumar Tusam

Low Level Design (LLD)

Document Version Control

Date Issued	Version	Description	Author
20th June 2020	1.1	First Draft	Sukumar Tusam
21th June 2023	1.2	Added Workflow chart	Sukumar Tusam
22th June 2023	1.3	Added Exception Scenarios Overall, Constraints	Sukumar Tusam
22st June 2023	1.4	Added KPIs	Sukumar Tusam
24th June 2023	1.5	Added user I/O flowchart	Sukumar Tusam

Contents

Abstract:

The Simple Banking Application is a basic banking system designed to provide users with essential banking functionalities such as checking the balance, depositing funds, withdrawing funds, and exiting the program. This application serves as an introductory example of a Java-based banking system and incorporates fundamental programming concepts such as input/output handling, variables, conditional statements, methods, and loops.

Low Level Design (LLD)

1 Introduction

1.1 Why this Low Level Design Document

The Low level design Document for the Simple Banking Application is created to provide a clear understanding of the design choices and decisions made during the development of the application. It serves as a reference for developers, stakeholders, and future maintainers, explaining the rationale behind the chosen architecture and facilitating the comprehension and evolution of the system

1.2 Scope

The scope of the Simple Banking Application is to provide users with basic banking functionalities, including checking the balance, depositing funds, withdrawing funds, and exiting the program. This application aims to simulate a simplified banking system and serves as a learning exercise for understanding core programming concepts.

1.3 Constraints

Constraints highlight the limited scope and simplicity of the Simple Banking Application like limited user input, Basic security, limited error handling and simple CLI.

1.4. Risks

Document specific risks that have been identified or that should be considered.

1.5. Out Of Scope:

The Simple Banking Application has certain limitations, and the following functionalities are considered out of scope for this project like;

Advanced Security feature, Persistent Data Storage, Multi-User Support, Extensive Error handling etc.

Low Level Design (LLD)

2 Technical Specification

.

2.1 Data Set

User Input	**Finalized**	**Source**
Deposit	yes	https://www.javatpoint.com/banking-application-in-java
Withdraw Amount	Yes	
Check Amount	Yes	
Exit	Yes	

2.1.1 User Dataset Overview

As the console menu will appear, user will choose different options as their input. Here user choose deposit as option and deposit his/her amount.

```
Welcome to the Simple Banking Application
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 2
Enter the amount to deposit: $40000
Amount deposited successfully.
```

```
Welcome to the Simple Banking Application
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 1
Your current balance is: $40000.0
```

```
Welcome to the Simple Banking Application
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 3
Enter the amount to withdraw: $1200
Amount withdrawn successfully.
```

```
Welcome to the Simple Banking Application
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter your choice: 4
Thank you for using the Simple Banking Application. Goodbye!
```

2.1.2 Hardware Component description

1. Computer:
   - A desktop, laptop, or server-class computer capable of running Java applications.
   - The computer should have a compatible operating system installed, such as Windows, mac OS, or Linux.
2. Processor (CPU):
   - A modern processor with adequate processing power to run the Java Virtual Machine (JVM) and execute the application's operations smoothly.
   - The processor should meet or exceed the minimum system requirements for running Java applications.
3. Memory (RAM):
   - Sufficient memory to accommodate the Java runtime environment and the application's data and code.
   - The recommended minimum RAM capacity for running Java applications is 4 GB, but this can vary depending on the size and complexity of the application and the operating system being used.
4. Storage:
   - Adequate storage capacity to hold the application's executable file and any associated resources.
   - The application itself is relatively small, so a few hundred megabytes of available storage should be sufficient.
5. Input and Output Devices:
   - A keyboard and mouse or other input devices for interacting with the command-line interface of the application.
   - A display device (monitor) to view the output and prompts displayed by the application.
6. Networking (Optional):
   - If the application needs to communicate with external systems or access remote resources, an internet connection or network interface may be required.

Low Level Design (LLD)

2.2 Logical architecture overview

- Describe the top level software components and their interactions/relationships.
- Use UML package diagrams and/or layer diagrams and/or interface diagrams.
- Describe also the operating systems on which the software runs.

2.2.1 Software Component description

The Simple Banking Application is a Java-based banking system that requires certain software components to run properly. The following software requirements should be considered:

1. Java Development Kit (JDK):
    - The JDK provides the necessary tools, libraries, and runtime environment to develop and execute Java applications.
    - The application requires a compatible version of the JDK to compile and run the Java code.
2. Integrated Development Environment (IDE):
    - An IDE, such as Eclipse, IntelliJ IDEA, or NetBeans, is recommended for development and debugging of the application.
    - The IDE should support Java development and provide features for code editing, compilation, and debugging.
3. Java Libraries and Packages:
    - The application may utilize standard Java libraries and packages for various functionalities.
    - These libraries may include java.util for input/output operations, java. Text for formatting and parsing, and java. Math for mathematical calculations.
4. Command-Line Interface (CLI):
    - The application utilizes a command-line interface for user interaction.
    - The CLI should be accessible and support basic input/output operations.
5. Operating System:
    - The Simple Banking Application is designed to be compatible with major operating systems such as Windows, mac OS, and Linux.
    - The application's code and execution should be compatible with the chosen operating system.
6. Java Virtual Machine (JVM):
    - The JVM is required to execute Java bytecode and run the Simple Banking Application.
    - The application should be compatible with the version of JVM installed on the system.
7. Testing Framework (Optional):
    - If unit tests or automated testing is desired, a testing framework like JUnit can be used to ensure the correctness of the application's functionalities.
8. Documentation and Version Control:
    - The project should include proper documentation to describe the application's functionality, usage, and code structure.
    - Version control systems like Git can be used to manage source code versions, facilitate collaboration, and track changes.

2.3 Deployment

Low Level Design (LLD)

3 Technology Stack

Front End	No UI
Backend	Java
Database	N/A
Deployment	GitHub

Low Level Design (LLD)

4 Proposed Solution

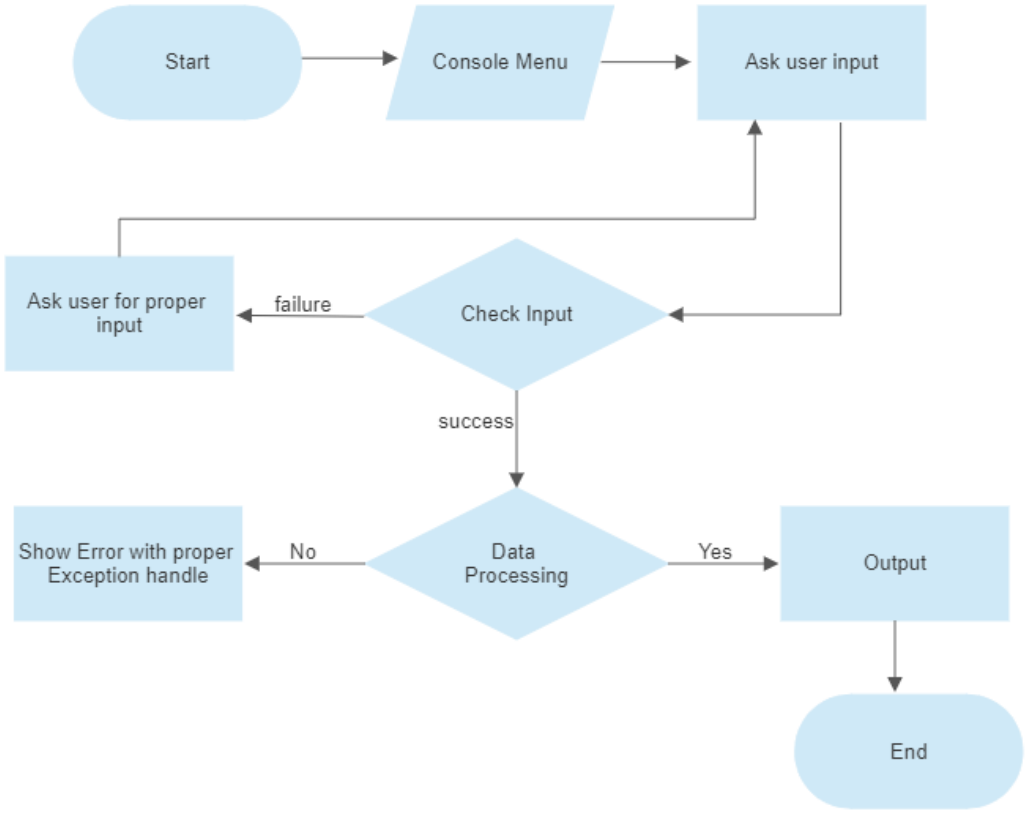
The proposed solution for the Simple Banking Application is to develop a Java-based banking system that provides basic banking operations such as checking the balance, depositing funds, withdrawing funds, and exiting the program. The application will have a command-line interface (CLI) for user interaction and will utilize in-memory data storage for user account details

Here is an outline of the proposed solution:

1. User Interface:
    - The application will provide a command-line interface (CLI) for users to interact with.
    - Users will be prompted with options to check balance, deposit funds, withdraw funds, or exit the program.
    - The CLI will display relevant information, such as account details and transaction results.
2. Basic Banking Operations:
    - The application will implement core banking functionalities, including:
        - Checking the balance: Users can view their current account balance.
        - Depositing funds: Users can deposit funds into their account by specifying the amount.
        - Withdrawing funds: Users can withdraw funds from their account if they have sufficient balance.
3. User Input Handling:
    - The application will use the Scanner class from the Java.util package to capture user input.
    - Input validation will be performed to ensure correct formatting and handle invalid inputs.
    - Appropriate error messages will be displayed for invalid operations or insufficient funds.
4. Error Handling:
    - Basic error handling mechanisms will be implemented to handle errors and exceptions during runtime.
    - The application will display informative error messages for users when necessary.
5. Security Measures:
    - While the focus of the application is on basic functionality, basic security measures should be implemented.
    - For example, user input may be masked when entering sensitive information such as passwords or PINs.

Low Level Design (LLD)

5. User / Application and Validation workflow



6. Exceptional Scenarios

Step	Exception	Mitigation	Module
20th June 2023	1.1	First Draft	Tapan Kumar Prusti
21th June 2023	1.2	Added Workflow chart	Tapan Kumar Prusti

7. Test Cases

Low Level Design (LLD)

Test case	Steps to perform test case	Module	Pass/Fail
1	Check Balance	Check Balance	Pass
2	Deposit 100/-	Deposit	Pass
3	Check Balance	Check Balance	Pass
4	Withdraw 50/-	Withdraw	Pass
5	Withdraw 200/-	Withdraw	Pass
6	Check Balance	Check Balance	Pass
7	Deposit 0/-	Deposit	Fail
8	Check Balance	Check Balance	Pass
9	Exit	Main	Pass
10			

8. Key performance Indicator

- User Adoption Rate: Measure the percentage of users who actively engage with and use the console banking application.
- Transaction Volume: Track the total number of transactions processed by the application over a specific period.
- Error Rate: Monitor the percentage of transactions or operations that result in errors or failures.
- Response Time: Measure the average time taken by the application to respond to user requests.
- Security: Evaluate the application's security measures to ensure the safety of user data and transactions.
- Application Stability: Monitor the application's stability and uptime, measuring the frequency and duration of any downtime.

Low Level Design (LLD)

- Code Quality: Assess the quality of the code base, including readability, maintainability, and adherence to best practices.
- Cost Efficiency: Evaluate the project's cost-effectiveness by comparing development and maintenance costs to the benefits delivered.