Low Level Design (LLD)

ATM Interface in Java (Console Based Application)

Written By:	Sohel Datta
Last Revised Date:	30-12-2024

Document Version Control:

Version	Date	Author	Description
1.0	30-12-2024	Sohel Datta	Initial draft of the document.
1.5	31-12-2024	Sohel Datta	Architecture & Architecture Description appended and updated
2.0	31-12-2024	Sohel Datta	Final Version of LLD

Contents

1. Introduction

1.1. Why this Low-Level Design Document?

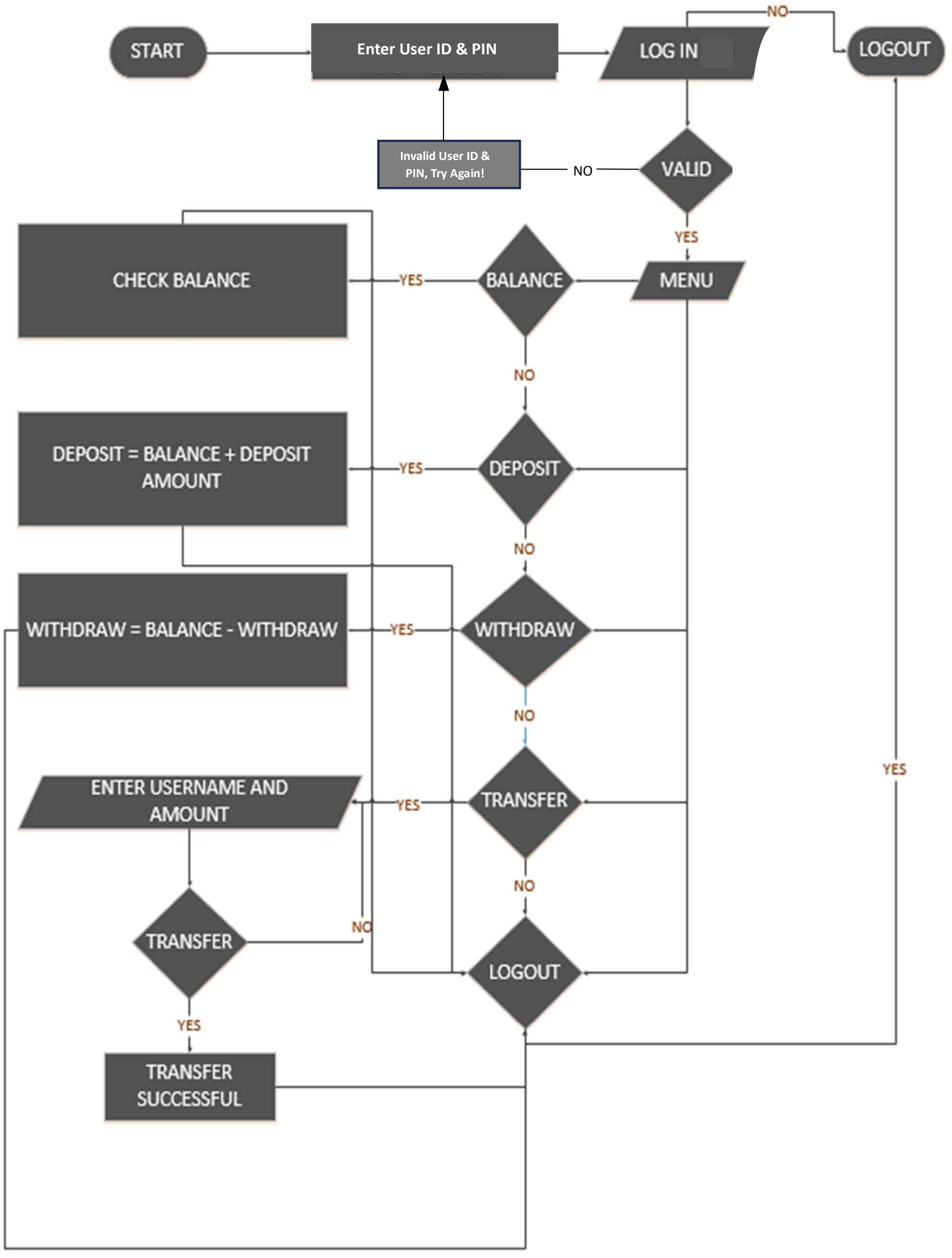
A Low-Level Design (LLD) document provides detailed information about the architecture and components of a software system, focusing on the implementation details of each module or component within the system. It serves as a blueprint for developers to understand how to build and integrate various parts of the application.

1.2. Scope

The scope of this document is to outline the design of a console-based ATM interface application in Java, detailing its architecture, data flow, and interactions with users and databases.

2. Architecture

2.1. Architecture Diagram



3. Architecture Description

3.1. Data Description

- **User Information:** Contains user account details such as account number, PIN, balance, etc.
- **Transaction Records:** Stores details of each transaction performed (withdrawals, deposits).

3.2. Data Transformation

Data transformation occurs when user inputs are validated and processed to ensure they conform to expected formats before being used in transactions.

3.3. Data Insertion into Database

When a transaction occurs (e.g., deposit or withdrawal), relevant data is inserted into the database to maintain an accurate record of user activities.

3.4. Export Data from Database

Data can be exported for reporting purposes or for backup using SQL queries that retrieve transaction records.

3.5. Data Pre-processing

Before processing transactions, the application checks if user inputs are valid (e.g., correct PIN, sufficient balance).

3.6. Data Clustering

Not applicable in this context as clustering is typically used in data mining; however, categorization of transactions can be implemented for reporting.

3.7. Model Building

The application logic is built around a model that defines how transactions are processed and how user interactions occur.

3.8. Data from User

User data is collected through console inputs where users enter their account numbers, PINs, and transaction amounts.

3.9. Data Validation

The application validates user inputs to ensure they meet criteria (e.g., correct PIN length, numeric values for amounts).

3.10. User Data Inserting into Database

User account information is inserted during account creation and updated during transactions.

3.11. Data Clustering

As mentioned earlier, this does not apply directly but could refer to grouping similar transaction types for analysis.

3.12. Model Call for Specific Cluster

In a more complex system, this would involve calling specific methods based on transaction types (e.g., withdrawal vs deposit).

3.13. Deployment

The application will be deployed as a standalone console application on client machines or servers where users can access it via command line.

4. Unit Test Cases

Test Case Description	Pre-Requisite	Expected Result
Validate correct User ID and PIN input	User has an existing account	System accepts valid credentials
Validate incorrect User ID or PIN input	User has an existing account	System rejects invalid credentials
Check balance after withdrawal	User has sufficient balance	Balance is updated correctly
Check balance after deposit	User deposits money	Balance reflects new total
Attempt withdrawal exceeding balance	User attempts to withdraw more than available balance	System rejects transaction with error message
Insert new user account	None	New user account is created in database

5. Conclusion

The development of the ATM Interface as a console-based application has successfully achieved its primary objectives, providing a user-friendly and efficient platform for performing essential banking transactions. Throughout the project, we focused on key functionalities, including user authentication, balance inquiries, deposits, withdrawals, fund transfers, and transaction logging.