High-Level Documentation

1. Introduction

The ATM Interface project is designed to simulate a basic Automated Teller Machine (ATM) system using Core Java and JDBC. The project aims to provide a user-friendly interface for performing banking operations such as account balance inquiry, cash withdrawal, fund transfer, and account statement generation.

2. Technologies Used

- Core Java: The project is implemented using Core Java, which provides the foundation for developing the ATM interface.
- JDBC (Java Database Connectivity): JDBC is used to establish a connection with the underlying database to store and retrieve customer account information.

3. Architecture Overview

The project follows a client-server architecture, where the ATM interface acts as the client, interacting with the database server to perform various banking operations.

4. Database Design

The project uses a relational database to store customer account details. The database design includes the following tables:

- Customer: Stores information about each customer, including their account number, name, PIN, and balance.
- Transactions: Records transaction details such as transaction ID, account number, type (withdrawal, deposit, transfer), amount, and timestamp.

5. User Interface

The ATM interface provides a menu-driven interface for users to interact with the system. The user interface includes options such as:

- Account Balance Inquiry: Allows users to check their account balance.
- Cash Withdrawal: Enables users to withdraw cash from their accounts.
- Fund Transfer: Allows users to transfer funds between accounts.
- Account Statement: Generates a statement with recent transactions.

6. Authentication and Security

The ATM interface implements user authentication to ensure that only authorized individuals can access the system. The authentication process involves validating the user’s account number and PIN against the database records.

7. Error Handling

The project includes appropriate error-handling mechanisms to handle exceptions such as invalid user input, insufficient funds, network failures, and database connectivity issues. Error messages are displayed to the user, providing helpful information about the encountered issue.

8. Logging and Audit Trail

To maintain a log of system activities, the project incorporates logging mechanisms. Relevant information, such as transaction details and error messages, are logged into a file for auditing purposes and troubleshooting.

9. Concurrency Control

To handle multiple user requests simultaneously, the project incorporates concurrency control mechanisms. This ensures that the
the system maintains data consistency and integrity during concurrent operations.

10. Testing
The ATM interface project includes unit testing to verify the correctness and robustness of the implemented functionalities.
Test cases are designed to cover various scenarios, including both valid and invalid inputs.

11. Deployment
The project can be deployed on a server or a local machine. The necessary setup includes configuring the database connection
properties, ensuring the availability of required libraries, and running the application.

12. Conclusion
The ATM Interface project using Core Java and JDBC provides a user-friendly and secure platform for performing banking operations.
It showcases the implementation of core programming concepts, database connectivity, error handling, security measures, and
concurrency control.