PURBANCHAL UNIVERSITY

HIMALAYAN WHITEHOUSE INTERNATIONAL COLLEGE

PUTALISADAK, KATHMANDU

A

Mini

Fourth Semester Project Report

On

“ATM Simulation System”

Submitted By:

Tilak Tamang

Roshni Karki

Ravi Raj Ghising

Submitted To:

THE DEPARTMENT OF SCIENCE & TECHNOLOGY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR IN INFORMATION AND TECHNOLOGY

December, 2025

Kathmandu, Nepal

APPROVAL CERTIFICATE

This is to certify that the project entitled “ATM Simulation System” submitted **by Tilak Tamang, Roshni Karki, and Ravi Raj Ghising** has been carried out under our supervision in partial fulfilment of the requirements for the degree of Bachelor in Information Technology (BIT).

The project work has been reviewed and found to be satisfactory. It is approved for submission to Purbanchal University as part of the academic requirements for the Bachelor in Information Technology program.

……………………………………

Project Supervisor

Kulraj Khanal
Department of Science & Technology
Himalayan Whitehouse International College

Date: ……………………

……………………………………

Head of Department

Bimal Sharma
Department of Science & Technology
Himalayan Whitehouse International College

Date: ……………………

ACKNOWLEDGEMENT

We would like to express our profound gratitude to the Department of Science & Technology, Himalayan WhiteHouse International College, for providing us with the opportunity and necessary resources to undertake this mini project as a partial requirement for the Bachelor in Information Technology (BIT) program.

We extend our sincere appreciation to our project supervisor for their invaluable guidance, continuous support, and constructive feedback throughout the duration of this project. Their expertise, encouragement, and supervision were instrumental in the successful completion of this work.

We are also grateful to all the faculty members of the department for their academic guidance, cooperation, and support, which greatly contributed to enhancing our knowledge and understanding of software development and system design concepts.

Finally, we would like to acknowledge our friends and family members for their patience, encouragement, and moral support throughout the course of this project.

ABSTRACT

The ATM Simulation project is designed to replicate the core functionalities of a real Automated Teller Machine in a virtual environment. It enables users to perform essential banking operations such as account authentication, balance inquiry, cash withdrawal, deposit, and mini statements. The system is developed using Java and follows an object-oriented approach to ensure modularity, reusability, and maintainability. This simulation provides a safe, cost-effective platform to understand ATM processes, test banking workflows, and enhance user interaction without involving actual financial transactions. The project also demonstrates the application of software engineering principles, data handling, and secure authentication mechanisms.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ATM	Automated Teller Machine
DFD	Data Flow Diagram
DOB	Date Of Birth
ER	Entity-Relationship
GUI	Graphical User Interface
IDE	Integrated Development Environment
NFC	Near Field Communication
PAN	Permanent Account Number
PIN	Personal Identification Number
SDLC	Software Development Life Cycle
SQL	Structured Query Language
UAT	User Acceptance Testing
UML	Unified Modelling Language

CHAPTER 1

INTRODUCTION

1.1 Background

The **ATM Simulation System** is a virtual extension of the “Bank Account Management System,” designed to emulate the functionalities of a real-world Automated Teller Machine (ATM). As digital banking becomes increasingly essential in today’s fast-paced, technology-driven environment, there is a growing need for simulated environments that demonstrate and test how core banking services can be accessed securely and conveniently outside of traditional bank branches.

Java offers platform independence and robust object-oriented features, making it ideal for developing scalable and modular applications. SQL, on the other hand, ensures efficient storage, retrieval, and manipulation of account data.

By integrating Java and SQL, the ATM Simulation System becomes a lightweight, portable, and powerful tool useful for banking demonstrations, training, and backend logic testing all without needing physical ATM hardware**.**

1.2Problem Statement

Managing and operating traditional banking services manually is no longer feasible in a fast-paced, globally connected world. Customers expect 24/7 access to their funds and banking information through secure and user-friendly interfaces. While Internet Banking covers many online services, there remains a significant need for physical and simulated interfaces like ATMs that allow for quick, secure access to basic banking functionalities.

However, real ATM systems are complex, expensive, and difficult to test without disrupting live operations. This creates a challenge for developers, testers, and banking trainees who need a risk-free platform to validate transaction workflows and system behaviour. In the current era of digital banking, there is a growing need for automated, self-service banking interfaces that allow customers to securely perform banking tasks at any time. While physical ATMs fulfil this need, they come with high deployment and maintenance costs, and testing them in real-time can pose risks to actual data and infrastructure.

Additionally, traditional manual banking systems are inefficient, error-prone, and not scalable to meet modern demands.

1.3 Objective

The main motto of our project is to develop an ATM Simulation & to be familiar with the features of JAVA & database. The project will be mainly based on the following objectives:

- To develop a secure, console-based ATM Simulation System that enables users to perform basic banking transactions such as balance inquiry, deposit, withdrawal, and transaction history retrieval.
- To implement a PIN-based authentication system that ensures only authorized users can access their account and perform transactions.
- To integrate a relational database (MySQL/SQLite) for securely storing user and transaction data with real-time updates.

1.4Scope & limitations

Our system will be designed to minimize the manual work. It will aim to maximize the productivity and provide improved managed system. Some of the scope and limitations of our proposed system is listed below:

Scope

- This system lets users perform basic ATM tasks like logging in securely with a PIN, checking their balance, depositing and withdrawing money, and viewing their transaction history.
- It uses a simple, easy-to-use text-based interface that anyone familiar with basic computers can operate
- All account details and transactions are safely stored in a database, ensuring information is updated instantly.

Limitations

- Right now, the system supports only one user at a time, so it doesn’t handle multiple users accessing it simultaneously.
- It doesn’t connect to real ATM machines or hardware like card readers so, it’s purely a software simulation.

CHAPTER 2
LITERATURE REVIEW

2.1Background

Initially, customers were required to visit bank branches to perform basic banking transactions such as cash withdrawal, deposit, and balance inquiry. This made routine banking activities time-consuming and inconvenient. The introduction of Automated Teller Machines (ATMs) significantly improved this situation by providing faster and more accessible banking services. However, physical ATMs also introduced challenges, particularly for users unfamiliar with their operation, and they involved high installation, maintenance, and security costs.

Since real ATM systems require active bank accounts and involve actual financial transactions, they are not suitable for training, testing, or learning purposes. Any error during testing could result in financial loss or data inconsistency. To overcome these limitations, ATM simulation systems were introduced as a safer and educational alternative. These systems replicate the behaviour of real ATMs in a controlled environment, allowing users to understand ATM operations and transaction workflows without any financial risks.

2.2 Research based on similar project

Several ATM simulations projects have been developed using different technologies and implementation approaches.

The project developed by Yadnyesh Raut focuses on a basic command-line ATM simulation system implemented in Java. The system includes essential features such as PIN-based authentication, balance inquiry, cash deposit, withdrawal, and account detail management. It uses file handling instead of a database to store account information, making the system simple and suitable for beginners, though less scalable and secure for large-scale applications [1].

Another ATM Simulation System proposed by Yash Kayastha uses a Java Swing–based graphical user interface to replicate core banking operations. The system supports PIN authentication, balance inquiry, cash deposit and withdrawal, and transaction history viewing. While the graphical interface provides a more realistic user experience compared to text-based systems, it also relies on file handling rather than database integration, which limits its scalability and data management capabilities [2].

A more advanced ATM simulation system is presented by Ankith Kumar S K, Vinaykumar H R, and Dr. Usha J, which utilizes Java, MySQL, and Swing within the Eclipse IDE. This system includes features such as user registration with validation (name, Aadhar, PAN, and date of birth), secure authentication using card number and PIN, core banking transactions, random card number and PIN generation, and full database integration for real-time data updates. The project emphasizes data security, validation testing, and user convenience, making it suitable for professional training and realistic ATM system simulation [3].

2.3 Functional and Non-functional Requirement

2.3.1. Functional Requirement

Functional requirements define specific operations the ATM Simulation System must support:

- **User Authentication**: The system should allow customers to securely log in using their card number and PIN.
- **Balance Inquiry**: Users can check the available balance in their account.
- **Cash Withdrawal**: Users should be able to withdraw a valid amount within their account balance and daily withdrawal limit.
- **Mini Statement**: Users can view the last few transactions.

2.3.2. Non-Functional Requirements

Non-functional requirements specify the quality attributes or constraints that the system must adhere to, such as performance, reliability, usability and security. These requirements define how the system should behave or perform rather that what it should do.

- **Performance**: Performance refers to how well the system performs in terms of speed, responsiveness, and efficiency. The system should respond to transactions quickly and handle multiple users without lag.
- **Reliability**: The system should be available whenever needed, minimizing downtime. It should ensure accurate transaction processing and data consistency at all times.
- **Usability**: Usability focuses on the ease of use and user experience of the system. It includes factors like user interface design, navigation, accessibility, and user support features. Interfaces should be clear and user-friendly for both customers and admins, even for users with minimal technical knowledge.

- **Security**: Security encompasses measures to protect the system from unauthorized access, data breaches, and malicious attacks. PIN-based authentication, session management, and encrypted database storage must protect user data from unauthorized access and attacks.

2.4 Feasibility Analysis

A feasibility study is a systematic analysis to determine the practicality and potential success of a proposed project or venture.

2.4.1. Technical Feasibility

The system will be implemented using Java (Swing for GUI) and MySQL/Oracle/SQLite for database operations. These technologies are widely supported and suitable for simulating ATM operations. Our group has verified that the required tools, libraries, and environment are available and compatible.

2.4.2. Operational Feasibility

Operational feasibility is all about problems that may arise during operations. There are two aspects related with the issue. What is the probability that the solution developed may not be put to use or may not work? What is the inclination of the management and end users towards the solution? The system is designed to replicate real-world ATM functionalities in a simplified, controlled environment. Since this is a simulation system, operational complexity is reduced, and the probability of end-user resistance is minimal. With proper guidance and usage instructions, users can operate the system smoothly.

2.4.3. Time Feasibility

Time feasibility refers to the assessment of whether a proposed project can be completed within a reasonable timeframe. After the study of the tasks involved in completing the project including requirement gathering, frontend development, backend development, coding, testing and others, we have concluded that the project timeline appears feasible. By carefully analysing tasks, resource availability & potential risks or delays, we can assess the feasibility of meeting the projects deadlines and make adjustments to the schedule as needed to ensure timely completion

2.4.4. Cost Feasibility

Since the project uses open-source tools (e.g., Java, MySQL) and does not require external hardware, the cost is limited to development time. Therefore, the system is cost-feasible and fits within academic or internal project constraints.

2.4.5. Legal Feasibility

Although this is a simulation, basic data protection principles are followed. Only authorized users can log in, and all personal and transactional data is stored securely in the database. The system does not handle real financial transactions, so legal risks are minimal, but privacy measures are still enforced.

CHAPTER 3
SYSTEM DESIGN AND METHODOLOGY

3.1. SDLC Model

The Software Development Life Cycle (SDLC) model is a structured approach used by software development teams to design, develop, test, deploy, and maintain software systems. It encompasses a series of phases or stages, each with specific activities and deliverables. Common SDLC models include Waterfall, prototype & spiral model with its own set of principles, methodologies, and best practices tailored to different project requirements and organizational needs.

The phases in the Software Development Life Cycle (SDLC) typically include:

1. **Planning**: This phase involves defining the project scope, objectives, timelines, and resources required. It may also include feasibility studies and risk assessments to ensure the project's viability.
2. **Requirement Analysis**: During this phase, the development team gathers and analyses requirements from stakeholders, such as users, customers, and business owners. The goal is to understand the needs and expectations of the software system to be developed.
3. **Design**: In this phase, the system architecture and design specifications are created based on the requirements gathered. This includes defining the software components, data structures, interfaces, and algorithms to be used in the system.
4. **Implementation**: Also known as coding or development, this phase involves translating the design specifications into actual code. Developers write, compile, and test the code to ensure it meets the requirements and design standards.
5. **Testing**: Once the code is developed, it undergoes rigorous testing to identify and fix defects or bugs. Testing includes various techniques such as unit testing, integration testing, system testing, and UAT.
6. **Deployment**: After successful testing, the software is deployed or released to the production environment. This phase involves installing the software on users' machines or servers and configuring it for use.
7. **Maintenance**: The final phase involves maintaining and supporting the software post-deployment. This includes addressing any issues or bugs reported by users, making enhancements or updates as needed, and ensuring the software remains functional and efficient over time.

These phases may vary slightly depending on the specific SDLC model or methodology being used, but they generally encompass the key activities involved in software development projects.

3.1.1. Selected model

Our project is aimed at developing an ATM Simulation System using Java programming language and a database for backend operations. After analysing various SDLC models, we concluded that the Spiral Model is the most suitable for our project. The Spiral Model is an iterative and risk-driven approach to software development that combines elements of both design and prototyping in stages, allowing for refinement through repeated cycles. It emphasizes early identification and resolution of risks, which is essential for a system like ours that deals with security, transaction integrity, and user authentication.

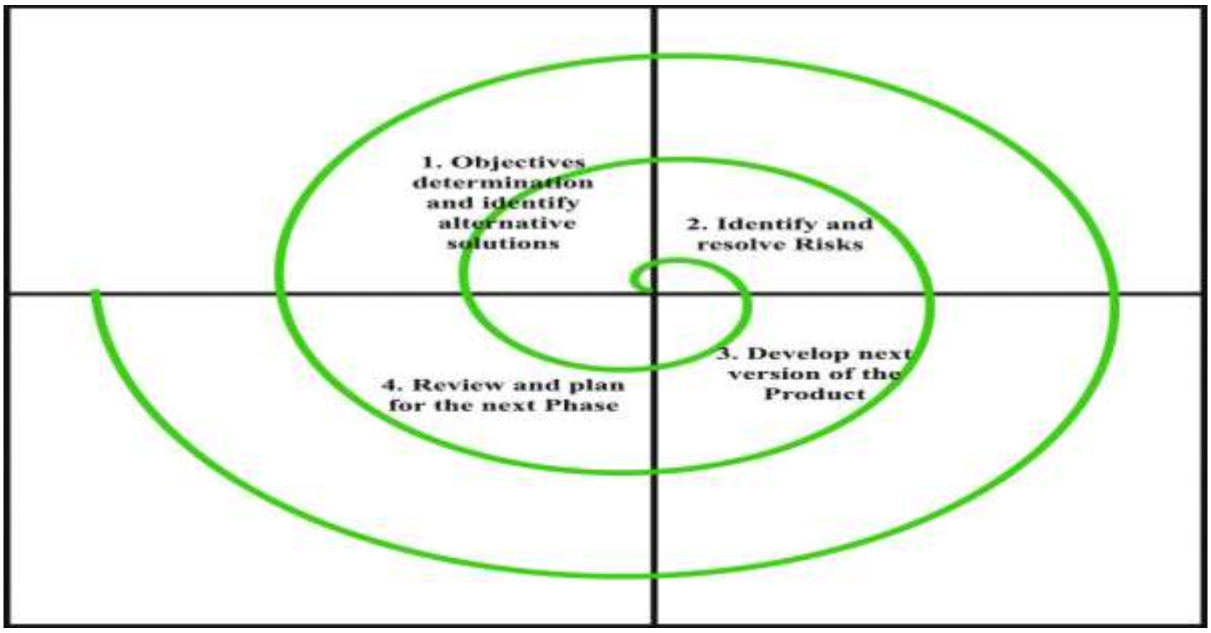
1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

Figure 1: The Spiral Model

The Spiral Model allows us to build the system incrementally by going through multiple iterations of planning, risk assessment, design, implementation, and evaluation. Each phase is revisited with more detail and functionality, which makes it easier to manage complexities and implement core ATM features such as balance inquiry, cash withdrawal, deposit, and transaction history with continuous feedback and improvement. Since our ATM simulator may involve frequent refinements and testing at every step, the iterative nature of the Spiral Model ensures flexibility and reliability throughout development.

Given the need for progressive development, secure functionality, and user-friendly design, the Spiral Model fits our project well. It allows us to address potential issues at each phase, build confidence through early prototypes, and deliver a well-tested and secure ATM simulation system.

3.2 Algorithm

Algorithm of our project ATM Simulation:

Step 1: Start.

Step 2: Enter ATM card number.

Step 3: Check if the card number exists in the database.

- If **No**, go to Step 4.
- If **yes**, go to Step 6.

Step 4: Sign up for a new account.

- Enter personal details.
- System generates a unique card number and PIN.

Step 5: Go to login screen.

Step 6: Enter PIN.

Step 7: Check if the PIN is correct.

- If **No**, display "Invalid PIN" and allow retry or exit.
- If **yes**, go to Step 8.

Step 8: Display ATM main menu:

1. Deposit
2. Withdraw
3. Fast Cash
4. Mini Statement
5. PIN Change

6. Balance Inquiry

7. Exit

Step 9: Read user choice and perform the selected action:

Case 1 – Deposit:

- Enter deposit amount.
- Update balance in database.
- Show confirmation.
- Return to Step 8.

Case 2 – Withdraw:

- Enter withdrawal amount.
- Check if balance is sufficient.
    - – If yes, deduct and dispense cash.
    - – If no, display "Insufficient Balance".
- Return to Step 8.

Case 3 – Fast Cash:

- Select predefined amount.
- Check balance and dispense if sufficient.
- Return to Step 8.

Case 4 – Mini Statement:

- Display last few transactions.
- Return to Step 8.

Case 5 – PIN Change:

- Enter new PIN.
- Update in database.

• Show confirmation.

• Return to Step 8.

Case 6 – Balance Inquiry:

• Display current balance.

• Return to Step 8.

Case 7 – Exit:

• Eject card.

• Go to Step 10.

Step 10: Stop.

3.3 Flowchart

A flowchart is a simple diagram that shows the steps or actions in a process or system. It uses different shapes like arrows, boxes, and diamonds to represent what happens and how things move from one step to another. Flowcharts help people understand how something works or what to do next, making complicated tasks easier to follow.

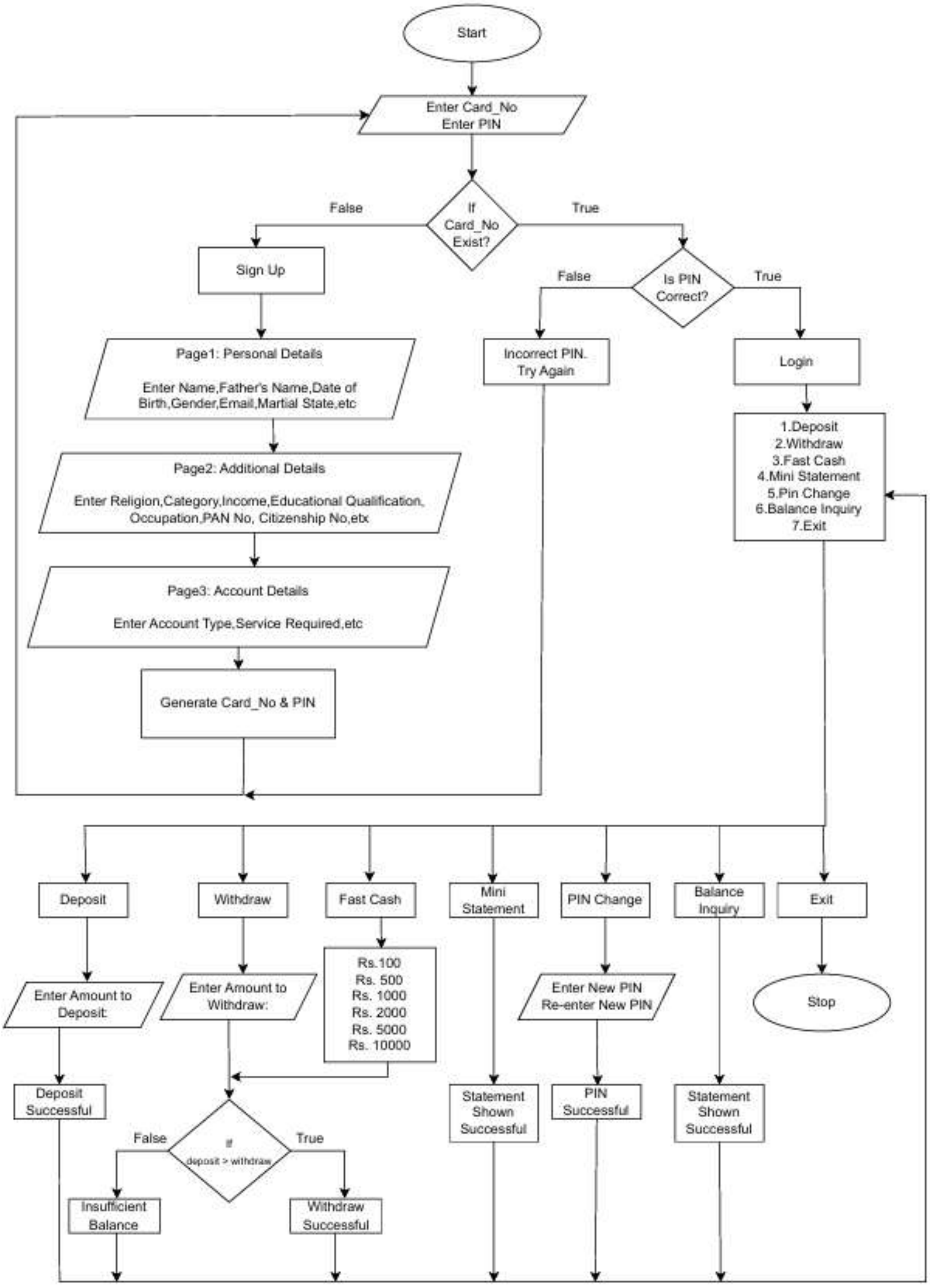

Figure 2: Flowchart

3.4. Context diagram

A context diagram is a high-level visual representation that illustrates the scope and boundaries of a system or process within its environment. It provides an overview of the interactions between the system being analysed and its external entities, such as users, other systems, or external stakeholders. Context diagrams help stakeholders understand the context in which the system operates and facilitate discussions about its requirements, interfaces, and dependencies.

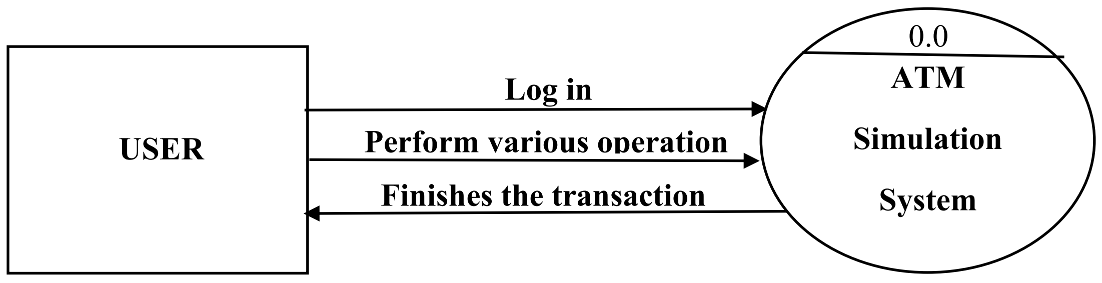

Figure 3: Context- Diagram

3.5. DFD

DFD stands for Data Flow Diagram. It's a graphical representation that illustrates how data flows through a system or process. DFDs consist of processes, data stores, data flows, and external entities. Processes represent activities or transformations that occur within the system, data stores depict where data is stored, data flows show the movement of data between processes and data stores, and external entities represent sources or destinations of data outside the system. The level 1 DFD of our proposed system is as shown below:

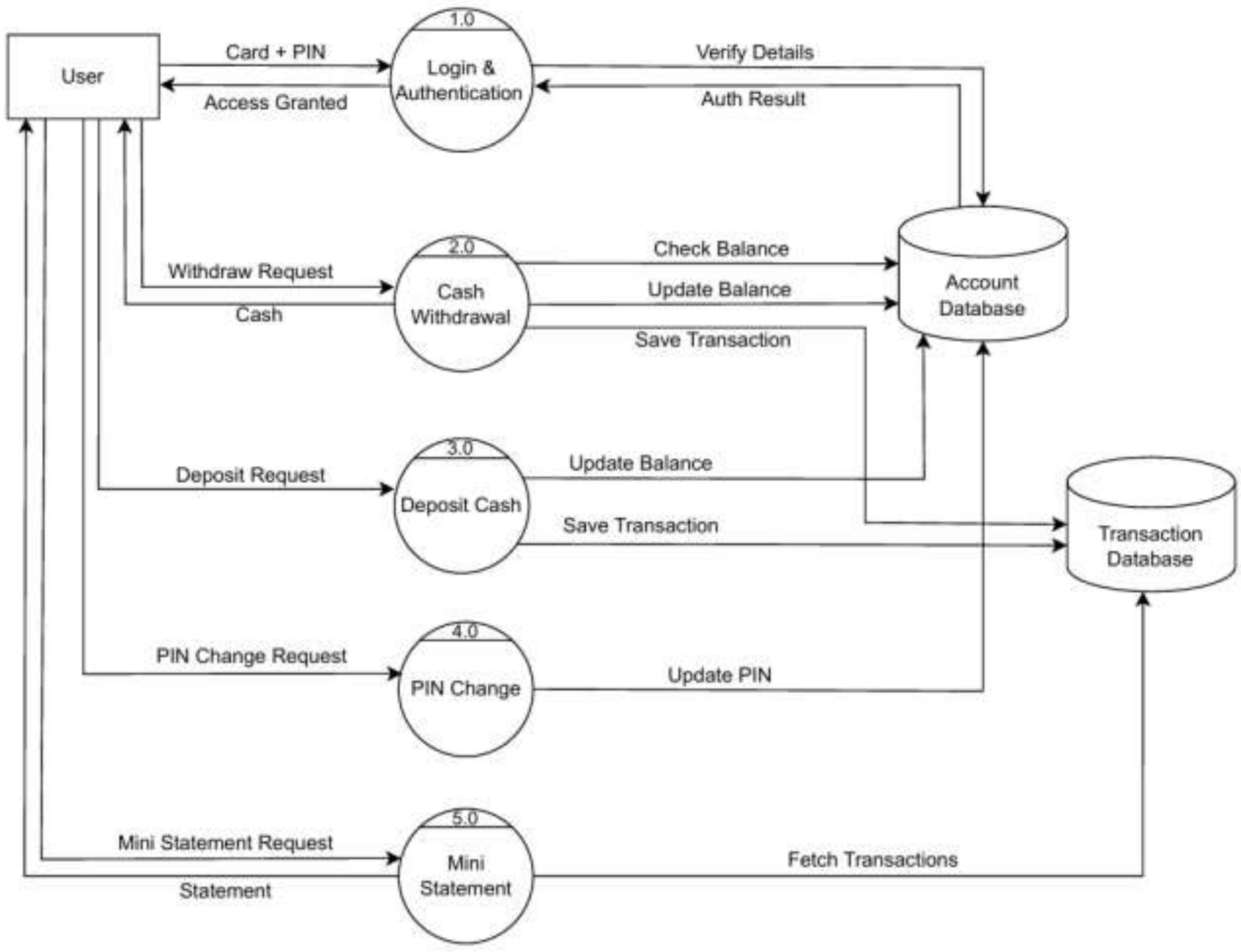

Figure 4: Level 1 DFD

3.6. ER-Diagram

An ER diagram is a visual representation that depicts the relationships among entities within a database. ER diagrams help in understanding the structure of a database and are commonly used during the database design phase to model the relationships between different entities and their attributes. The ER diagram of our purposed system which will be further modified according to our requirements.

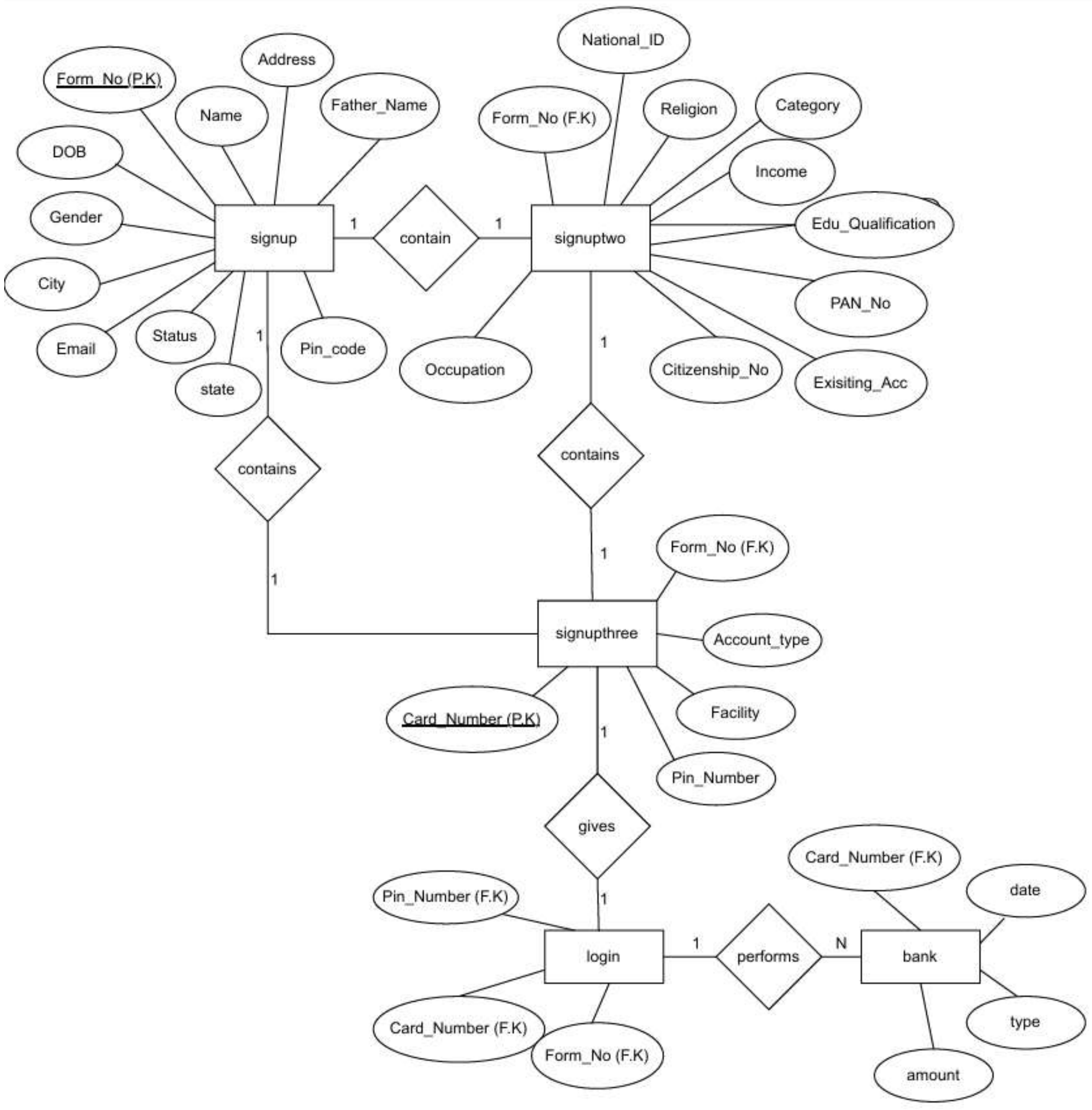

Figure 5: ER-Diagram

3.7. Sequence Diagram

A sequence diagram is a UML diagram that shows how objects interact with each other by passing messages in a specific order over time. It helps to understand the flow of a process step-by-step, such as user login or ATM transactions. The sequence diagram of our purposed system which will also be further modified according to our requirements.

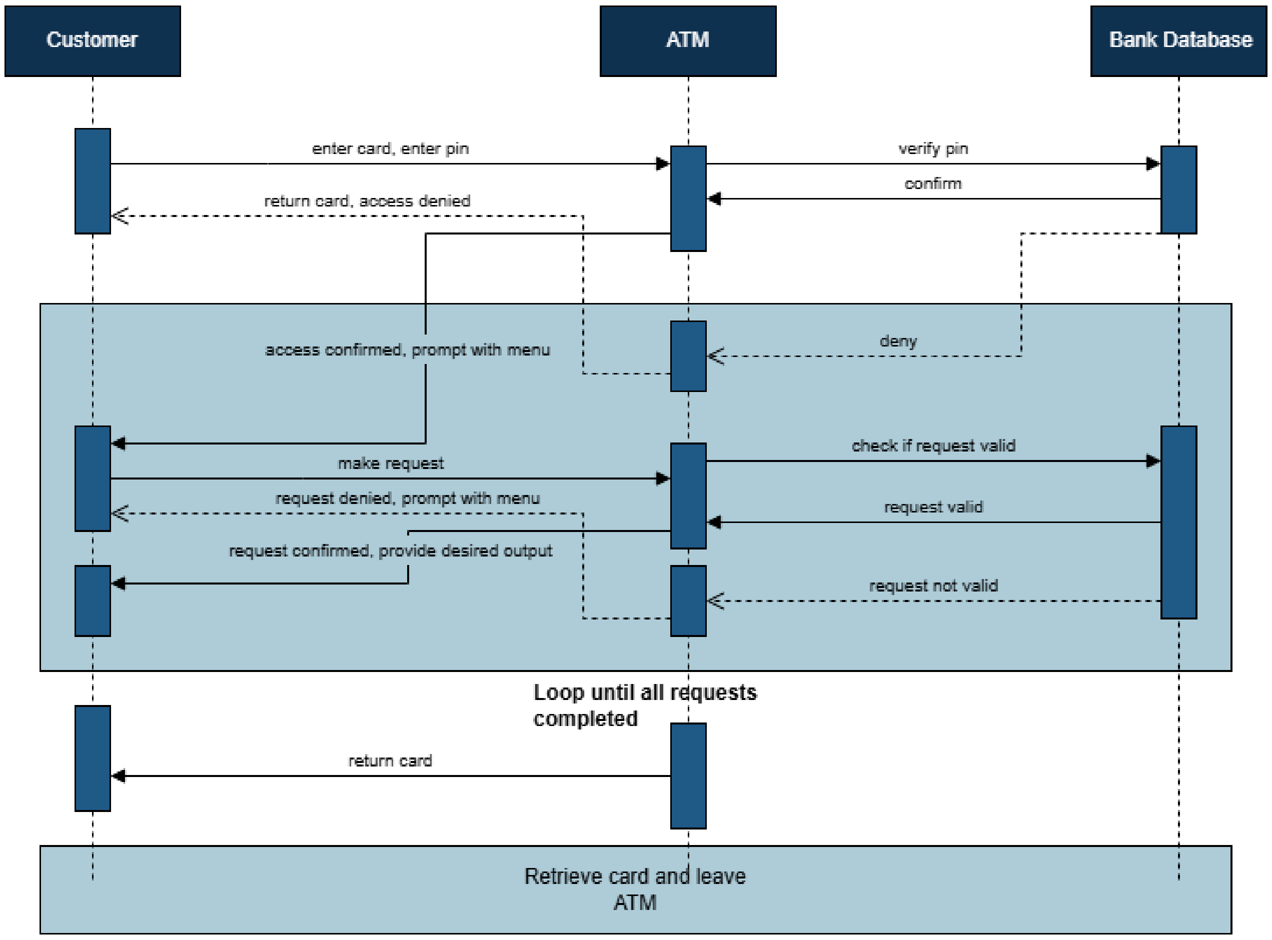
Customer
ATM
Bank Database
enter card, enter pin
verify pin
confirm
return card, access denied
access confirmed, prompt with menu
deny
check if request valid
make request
request denied, prompt with menu
request valid
request confirmed, provide desired output
request not valid
Loop until all requests completed
return card
Retrieve card and leave ATM

Figure 6: Sequence Diagram

3.8. Class Diagram

A class diagram is a type of UML (Unified Modelling Language) diagram that shows the structure of a system by displaying classes, their attributes, methods, and the relationships between them. The Class diagram of our purposed system which will be further modified according to our requirements.

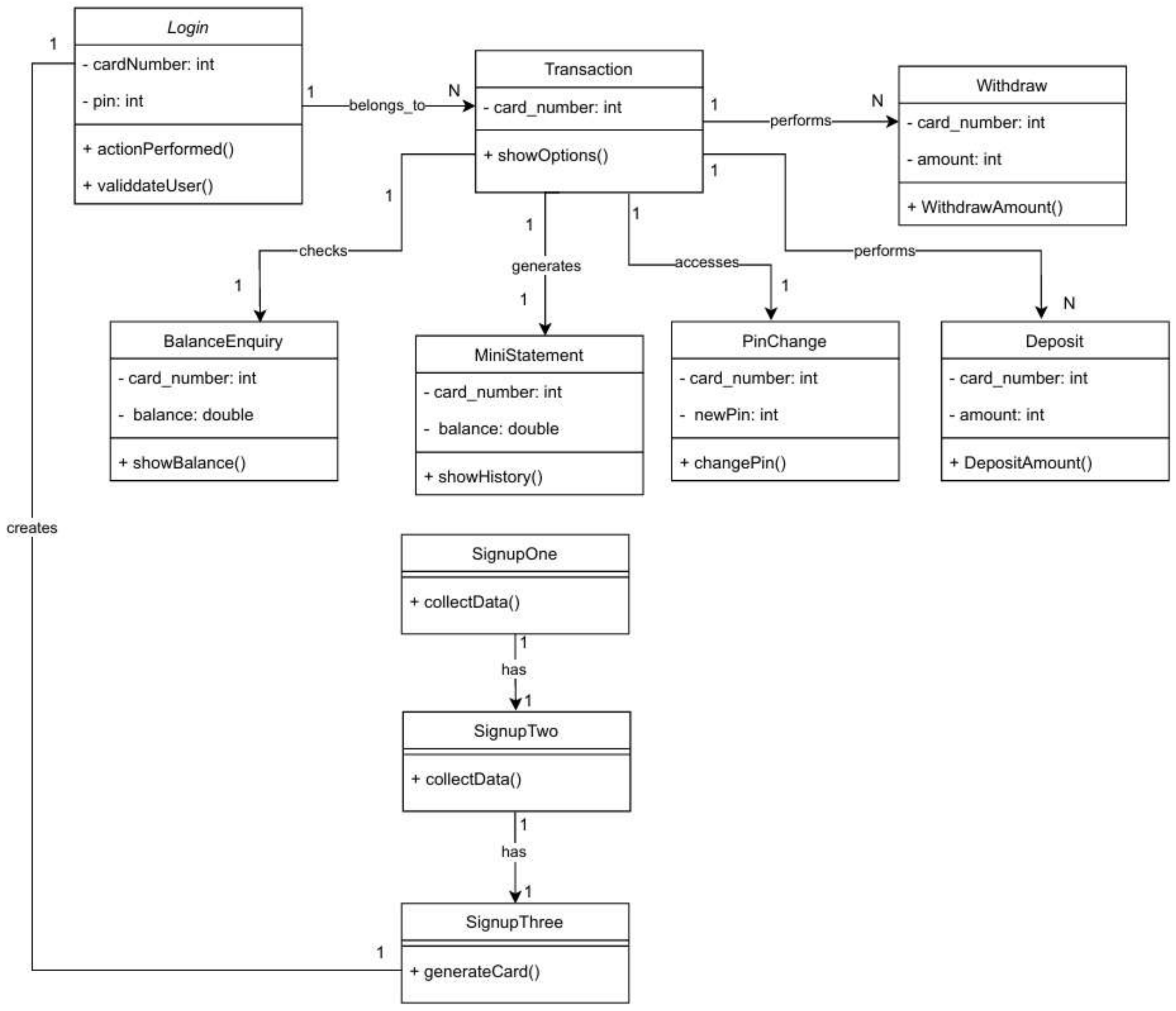

Figure 7: Class Diagram

3.9. Use Case Diagram

A use case diagram is a type of behavioural diagram in Unified Modelling Language (UML) which shows the functionality of a system from the perspective of its users and helps to understand how users interact with the system.

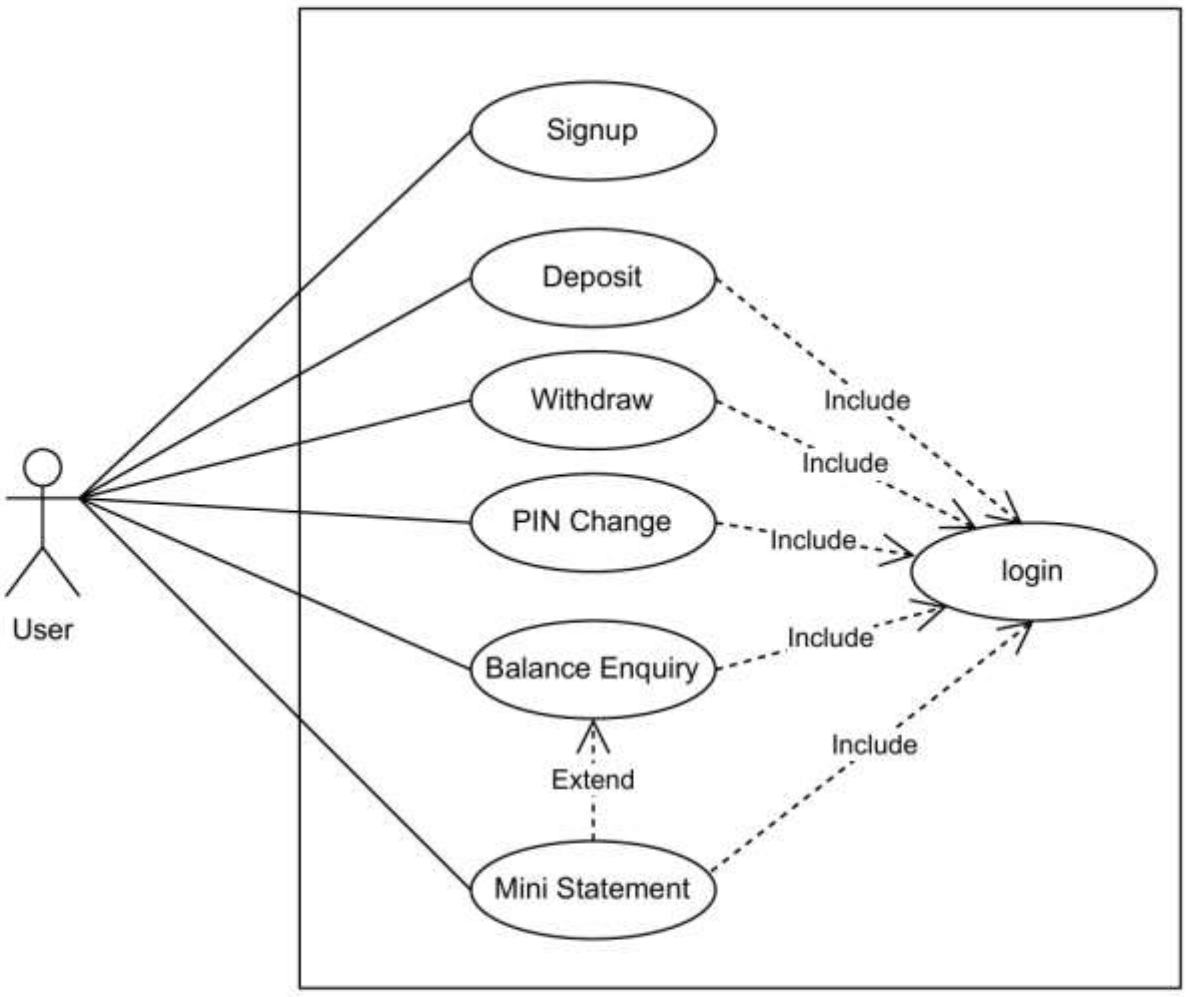

Figure 8: Use Case Diagram

CHAPTER 4

IMPLEMENTATION

4.1 Software and Hardware Requirement

4.1.1 Hardware Requirement

The hardware requirements define the minimum physical resources needed to run the ATM Simulation System efficiently. Since this is a software-based simulation, the hardware requirements are minimal.

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB
- Hard Disk: At least 20 GB free storage
- Input Devices: Keyboard, Mouse
- Display: Standard monitor with minimum 1024×768 resolution.

4.1.2 Software Requirement

The software requirements specify the development tools, platforms, and technologies used to design, develop, and test the ATM Simulation System.

- Operating System: Windows 10 or higher
- Programming Language: Java
- Database: MySQL
- Web Server: Apache (via XAMPP)
- IDE: SpringSource Tool Suite / Eclipse IDE
- Database Management Tool: XAMPP Control Panel
- Browser: Google Chrome or any modern web browser.

4.2 Testing

Testing is a crucial phase of the software development process that ensures the ATM Simulation System functions correctly according to the specified requirements. The primary goal of testing is to identify defects, verify system behaviour, and ensure reliability, security, and performance before deployment. Various testing methods are applied to validate individual components as well as the complete system.

4.2.1 Test Cases for Testing Method

Different testing techniques are used to ensure the correctness and stability of the ATM Simulation System.

- **Unit Testing**

Unit testing focuses on testing individual modules or components of the system independently. Each function, such as user authentication, balance inquiry, cash withdrawal, deposit, and PIN change, is tested separately to verify that it produces the expected output for given inputs. This helps in detecting errors at an early stage of development.

- **System Testing**

System testing evaluates the complete and integrated ATM Simulation System as a whole. It verifies whether all modules work together correctly and ensures that the system meets the functional requirements. Scenarios such as valid and invalid login, sufficient and insufficient balance during withdrawal, and successful transaction updates are tested during this phase.

- **Integration Testing**

Integration testing is performed after unit testing to verify the interaction between different modules of the system. This ensures that data flows correctly between components such as the user interface, business logic, and database.

- **User Acceptance Testing (UAT)**

User Acceptance Testing is carried out to ensure that the system meets user expectations and is easy to use. End users test the system in a controlled environment to validate its functionality, usability, and overall performance.

CHAPTER 5

ANALYSIS AND EVALUATION

5.1 Analysis of output obtained

The analysis of the output obtained from the ATM Simulation System is carried out based on the predefined objectives of the project. The system was evaluated by executing various ATM operations and observing whether the outputs matched the expected results.

The first objective was to develop a secure authentication mechanism using a card number and PIN. During testing, the system successfully validated user credentials and restricted access when incorrect PINs were entered, ensuring secure login and preventing unauthorized access.

Another objective was to implement core ATM functionalities such as balance inquiry, cash deposit, cash withdrawal, mini statement generation, and PIN change. The system correctly displayed the current account balance, updated the balance after deposit and withdrawal transactions, and generated accurate mini statements showing recent transactions. Withdrawal operations were restricted when the account balance was insufficient, demonstrating proper validation and error handling.

Database integration was also a key objective of the project. The output analysis showed that all transaction data and account details were stored and retrieved accurately from the database. Real-time updates ensured data consistency after every transaction.

The system also aimed to provide a simple and user-friendly interface. The output results confirmed that users could easily navigate through the available options and perform transactions without confusion.

Overall, the observed outputs of the ATM Simulation System met the predefined objectives effectively. The system demonstrated correctness, reliability, security, and efficiency in handling ATM operations, validating the successful implementation of the proposed project.

5.2 Gantt Chart

We have outlined the timeline for the implementation of the ATM below using a Gantt chart. This chart illustrates the major tasks, their dependencies and the estimated duration for each task.

Table 1: Gantt Chart

GANTT CHART								
S.N	Starting date:7 June					Ending date:7 September		
	TASK	WEEKS						
1		1/2nd	3/4th	5/6th	7/8th	9/10th	11/12th	13/14th
2	Requirement Analysis	▨	▨					
3	Risk Assessment		▨	▨				
4	Design		▨	▨	▨			
5	Prototype Development			▨	▨			
6	Frontend development			▨	▨	▨	▨	▨
7	Backend developemnt				▨	▨	▨	▨
8	Database implementation					▨	▨	▨
9	Integration & testing						▨	▨
	Documentation	▨	▨	▨	▨	▨	▨	▨

CHAPTER 6

FINAL OUTCOME

At the completion of this project, an Automated Teller Machine (ATM) Simulation System has been successfully developed using Java and database technologies. The system effectively replicates the core functionalities of a real ATM in a controlled and risk-free environment.

The final system provides the following outcomes:

- **Secure Login Interface:**
  The system includes a secure authentication mechanism where users log in using their card number and Personal Identification Number (PIN).
- **User Authentication:**
  The entered credentials are verified against stored records in the database to ensure that only authorized users can access the system.
- **Transaction Dashboard:**
  Upon successful authentication, users are presented with a menu-driven interface that allows them to perform various banking operations.
- **ATM Transaction Functionalities:**
  The system supports essential ATM operations including:
  - Deposit of cash
  - Withdrawal of cash
  - Fast cash withdrawal
  - Balance enquiry
  - Mini-statement viewing
  - PIN change facility
- **Database Integration:**
  All user details and transaction records are stored and updated in real time within the database, ensuring data accuracy and consistency.
- **Transaction Security and Integrity:**
  The system ensures that all transactions are processed securely, with proper validation to prevent unauthorized access and invalid operations.

Overall, the developed ATM Simulation System fulfils the predefined project objectives by providing a reliable, secure, and user-friendly platform for understanding and demonstrating ATM operations. The system serves as an effective educational and practical tool for learning core banking transaction processes and software development concepts

CHAPTER 7
CONCLUSION & FUTURE RECOMMENDATION

7.1. Problem faced and their implementation with limitation of the project

User Authentication Errors:

- Problem: Handling incorrect PIN entries and unauthorized access.
- Implementation: PIN validation was implemented with restricted access for invalid attempts.
- Limitation: Advanced security features like biometric authentication are not included.

Database Connectivity Issues:

- Problem: Ensuring consistent connection between the application and the database.
- Implementation: Proper database configuration and error handling mechanisms were used.
- Limitation: The system depends on a local database and does not support cloud-based databases.

Transaction Consistency:

- Problem: Maintaining accurate balance updates during deposit and withdrawal operations.
- Implementation: Transactions were validated before updating the database.
- Limitation: The system does not handle concurrent transactions from multiple users.

Limited Scalability:

- Problem: Supporting multiple users simultaneously.
- Implementation: The system was designed for single-user operation to ensure simplicity.
- Limitation: Multi-user and multi-threaded transaction handling is not supported.

User Interface Constraints:

- Problem: Providing an intuitive interface with limited resources.
- Implementation: A simple and clear interface was designed for ease of use.
- Limitation: Advanced graphical and interactive UI features are not implemented.

Hardware Simulation:

- Problem: Simulating real ATM hardware components.
- Implementation: ATM operations were simulated purely through software logic.
- Limitation: Physical components such as card readers and cash dispensers are not integrated

7.2. Conclusion

After conducting an in-depth analysis of the project’s outcomes and requirements, the implementation of the ATM Simulation System demonstrates significant improvements in transaction efficiency, data accuracy, security, user experience, compliance, and operational cost-effectiveness. Efficiency gains are evident through streamlined processes such as deposits, withdrawals, balance inquiries, and PIN changes, reducing manual errors and processing time. Enhanced data accuracy is achieved with reliable transaction records and minimized inconsistencies. The incorporation of secure authentication and user verification features strengthens security, ensuring that only authorized users can access sensitive account information and perform transactions safely.

7.3. Future enhancement

Looking ahead, here are some forward-thinking recommendations for improving the ATM Simulation System:

1. **Biometric Authentication:** Integrate biometric methods such as fingerprint or facial recognition to enhance user authentication and security during ATM access.
2. **Mobile Banking Integration:** Develop features to allow users to perform ATM-related transactions through a mobile app for greater convenience.
3. **Multi-language Support:** Implement support for multiple languages to cater to a wider range of users and improve accessibility.
4. **Advanced Fraud Detection:** Introduce intelligent monitoring to detect and prevent suspicious activities and unauthorized transactions.
5. **Contactless Transactions:** Add support for contactless card payments or NFC-based transactions to improve speed and user convenience.

REFERENCES

[1] Y. Raut, “ATM Simulation System using Java,” *Code with Curious*, Aug. 2023. [Online]. Available: https://codewithcurious.com/projects/atm-simulation-system-using-java/ [Accessed: 5th August]

[2] Y. Kayastha, “ATM Simulation,” *GitHub*, Sep. 2023. [Online]. Available: https://github.com/kyt47000/ATM-simulation [Accessed: 5th August]

[3] A. K. S. Kumar, V. H. R., and U. J., “ATM Simulation Using Java,” *International Advanced Research Journal in Science, Engineering and Technology*, vol. 9, no. 2, pp. 1–6, Feb. 2022. [Online]. Available: https://iarjset.com/wp-content/uploads/2022/02/IARJSET.2022.9202.pdf [Accessed: 5th August].