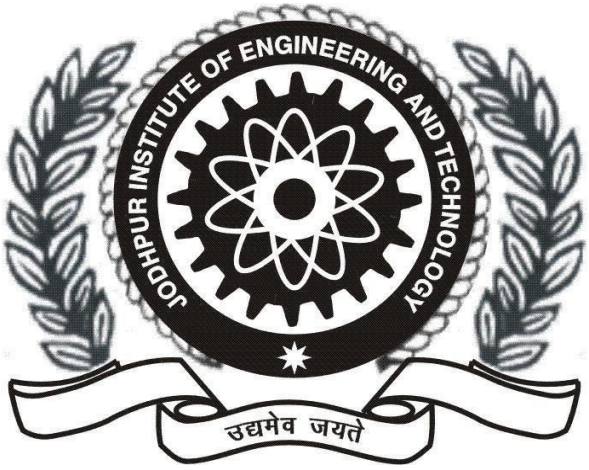
Jodhpur Institute of Engineering and Technology, Jodhpur
Department of Computer Science and Engineering
Session 2022-26

SOFTWARE ENGINEERING LAB PROJEC REPORT ON

ATM Management System

In partial fulfillment of…

B.Tech. III year **(Computer Science & Engineering.)**



Work Carried Out At "JIET Group of institutions"

Submitted To:

Rajendra Purohit
Assistant Professor

Submitted by:

Abhishek malav
T22EJICS010

Acknowledgment

I would like to acknowledge the contributions of the following people without whose help and guidance this project would not have been completed.

I respectfully thank Urvashi Harsh , for providing me an opportunity to do this project work and giving me all support and guidance, which made me complete the project up to very extent.

I am also thankful to Mamta Garg, HoD of Computer Science and Engineering Department, Jodhpur Institute of Engineering and Technology, for his/her constant encouragement, valuable suggestions and moral support and blessings.

Although it is not possible to name individually, I shall ever remain indebted to the faculty members of Jodhpur Institute of Engineering and Technology, for their persistent support and cooperation extended during his work.

This acknowledgement will remain incomplete if I fail to express our deep sense of obligation to my parents and God for their consistent blessings and encouragement.

Table of Contents

1. Introduction

Project Name: **ATM Management System**

The Automated Teller Machine (ATM) Management System is a critical component of modern banking infrastructure, enabling customers to perform a variety of financial transactions with convenience and efficiency. As the banking landscape continues to evolve, the role of ATMs has expanded beyond simple cash withdrawal and deposit functionalities. Today, ATMs serve as multifaceted banking kiosks, allowing users to check account balances, transfer funds, pay bills, and even purchase mobile top-ups.

This report aims to provide a comprehensive overview of an ATM Management System, detailing its architecture, functionalities, and the technological advancements that have influenced its development. The system is designed to enhance user experience by ensuring quick access to financial services while maintaining robust security measures to protect sensitive data.

The proliferation of ATMs globally has transformed customer interactions with banks, reducing the reliance on physical branches and providing round-the-clock access to financial resources. This shift has necessitated the implementation of sophisticated management systems to oversee the operation of ATMs, including transaction processing, monitoring cash levels, managing maintenance schedules, and ensuring compliance with regulatory standards. Moreover, the rise of digital banking and mobile applications has driven the integration of ATMs with various technologies such as cloud computing, biometric authentication, and real-time data analytics. These innovations not only improve operational efficiency but also enhance security protocols, thus fostering greater trust among users.

In this report, we will explore the fundamental components of an ATM Management System, including hardware and software architecture, user interface design, and the backend processes that ensure seamless transactions. Additionally, we will discuss the challenges faced by ATM operators, such as fraud prevention, machine maintenance, and user support, as well as future trends that may shape the evolution of ATM technology.

By examining these aspects, we aim to provide insights into the importance of a well-designed ATM Management System and its impact on the overall banking experience, paving the way for further advancements in the realm of financial technology.

2. Technologies and Tools Learned

❖ Frontend Development:

- **HTML & CSS:** For structuring and styling the web pages.
- **JavaScript:** For Some functionality and logic.
- **Database**: MySQL (relational database).

❖ Development Tools and Environments:

- **VS Code:** A powerful code editor with numerous extensions and integrations.
- **Git & GitHub:** For version control and collaborative development.

❖ Design and Prototyping:

- **Smart Draw:** For creating system architecture diagrams and flowcharts.

❖ Libraries and Frameworks:

- **React:** It is JavaScript library for use Sigle page component.
- **CSS:** CSS is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

By learning and applying these technologies, I have developed a comprehensive skill set that has been instrumental in the successful completion of the ATM Management System project. This knowledge base will also be invaluable for future enhancements and additional projects.

3. Technical Details of Project

Technical Details of the Training Management System :

The **ATM Management System** is designed with modern web technologies and follows best practices for software development. Below are the detailed technical aspects of the project, covering the system architecture, technologies used, database design, and various key components that make up the entire system.

---

1. System Architecture:

The ATM Management System follows a **three-tier architecture** comprising the Presentation Layer, Business Logic Layer, and Data Layer. This architectural design separates the responsibilities of the application, ensuring that the system is maintainable, scalable, and secure.

- **Presentation Layer (Front-End)**: The user-facing part of the system that handles the interaction between the user (ATM, user) and the system.
    - **Technologies**: HTML5, CSS3, JavaScript, Bootstrap
    - **Frameworks/Libraries**: jQuery, Ajax, or other JavaScript libraries
    - **Responsiveness**: Bootstrap ensures that the system is fully responsive, adapting to various devices such as desktops, tablets, and smartphones.
    - **Security**: Front-end security is managed by input validation and using HTTPS to encrypt data between client and server.
- **Business Logic Layer (Back-End)**: The middle layer that processes user requests, performs data validation, and executes the necessary logic before communicating with the database.
    - **Technology**: PHP (Hypertext Preprocessor)
    - **Framework (optional)**: Laravel or CodeIgniter (if a PHP framework is chosen to provide MVC architecture)
    - **Role**: Handles the core logic of the system including user authentication, course management, reporting, and email notifications.
    - **APIs**: RESTful APIs may be developed for communication between the front-end and back-end or for integrating third-party services (e.g., email, SMS, payment gateways).
    - **Security**: Implementing server-side validation, securing APIs using tokens (JWT or OAuth2), and sanitizing inputs to prevent SQL Injection and Cross-Site Scripting (XSS).
- **Data Layer (Database)**: The storage layer responsible for persisting and retrieving data.
    - **Database**: MySQL (Relational Database Management System)
    - **Tables**: Designed with normalization principles to avoid data redundancy and maintain data integrity.
    - **Security**: SQL prepared statements are used to prevent SQL injection. Data is encrypted and

only authorized users have access to sensitive data.
    - **Backup and Recovery**: Regular database backups are implemented, and data recovery protocols are in place in case of system failure.

---

2. Technologies Used

- **Front-End**:
    - **HTML5**: For structuring web pages and content.
    - **CSS3**: For styling the web pages and providing an attractive, user-friendly interface.
    - **JavaScript**: For adding interactivity to the web pages (e.g., form validation, dynamic content loading).
    - **Bootstrap**: A front-end framework for responsive design and to make the application mobile-friendly.
    - **AJAX**: For asynchronous requests to the server without reloading the page, improving user experience.
- **Back-End**:
    - **PHP**: A widely used open-source scripting language, particularly suited for web development. It handles business logic, interacts with the database, and serves content dynamically.
    - **Laravel (optional)**: A PHP framework that follows the MVC (Model-View-Controller) architecture, making the code modular and easier to maintain.
    - **Apache HTTP Server**: The server software used to host the system locally (via XAMPP) or in production.
    - **Email APIs**: Integrated with email services (e.g., PHPMailer or SendGrid) to automate notifications and alerts.
- **Database**:
    - **MySQL**: A relational database management system that stores all data related to user, transactions, user deatails, and Bank details.
    - **SQL Queries**: Used to interact with the database, retrieving, updating, or deleting data.
    - **Normalization**: The database is normalized to ensure data integrity and minimize redundancy.
- **Version Control**:
    - **Git**: For source code version control. Git ensures code changes are tracked and helps in collaboration among team members.
    - **GitHub**: Used to host the code repository and manage project collaboration.
- **Development Tools**:
    - **Visual Studio Code**: The integrated development environment (IDE) used for writing and managing code.
    - **XAMPP**: A local server package that provides an environment for running Apache, MySQL,

and PHP locally.

---

3. Database Design

The database schema for the ATM Management System is designed using a relational model, allowing for efficient data management and clear relationships between various entities. The system comprises multiple tables that collectively manage the information associated with ATMs, transactions, users, and maintenance. Below is an outline of the primary tables involved in the database design:

1. User Table

- **user_id**: Primary key, unique identifier for each user.
- **username**: Stores the user's chosen username.
- **password**: Stores the user's password (hashed for security).
- **email**: Stores the user's email address.
- **phone**: Stores the user's phone number.
- **role**: Indicates the role of the user (e.g., Customer, Bank Staff, Administrator).

2. ATM Table

- **atm_id**: Primary key, unique identifier for each ATM.
- **location**: Stores the physical address or location of the ATM.
- **status**: Indicates whether the ATM is operational, under maintenance, or out of service.
- **cash_balance**: Stores the current cash balance available in the ATM.

3. Transaction Table

- **transaction_id**: Primary key, unique identifier for each transaction.
- **user_id**: Foreign key referencing the User table.
- **atm_id**: Foreign key referencing the ATM table.
- **transaction_type**: Specifies the type of transaction (e.g., Withdrawal, Deposit, Transfer).
- **amount**: The amount involved in the transaction.
- **transaction_date**: Timestamp of when the transaction occurred.

4. Maintenance Table

- **maintenance_id**: Primary key, unique identifier for each maintenance record.
- **atm_id**: Foreign key referencing the ATM table.
- **maintenance_date**: Date when maintenance was performed.
- **description**: Details about the maintenance work carried out.
- **technician_id**: Foreign key referencing the User table (if the technician is also a user).

5. Audit Log Table

- **audit_id**: Primary key, unique identifier for each log entry.
- **user_id**: Foreign key referencing the User table.
- **action**: Describes the action taken (e.g., Login, Logout, Transaction).
- **timestamp**: Date and time when the action occurred.

Relationships and Integrity

These tables are interconnected through foreign keys, ensuring referential integrity within the database. For example:

- The **user_id** in the Transaction table references the corresponding **user_id** in the User table, linking transactions to the specific user who performed them.
- The **atm_id** in both the Transaction and Maintenance tables references the ATM table, ensuring that all transactions and maintenance records are associated with the correct ATM.

---

4. Key Components and Features

1. User Authentication

- **Secure Credential Storage**: User passwords are stored using hashing algorithms (e.g., bcrypt) to ensure security.
- **Role-Based Access Control**: Different user roles (e.g., Customers, Bank Staff, Administrators) are implemented to control access to various functionalities within the system.

2. ATM Management

- **ATM Configuration**: Administrators can add, edit, or remove ATMs from the system, managing details such as location, status, and cash balance.
- **Real-Time Monitoring**: The system provides real-time status updates on ATM operational health, including cash levels and maintenance needs.

3. Transaction Management

- **Transaction Processing**: Users can perform various transactions, including withdrawals, deposits, and transfers, with real-time updates to their account balances.
- **Transaction History**: Users can view their transaction history, providing transparency and aiding in financial tracking.

4. Automated Notifications

- **Event-Driven Emails**: Users and administrators receive email notifications for important events, such as transaction confirmations, maintenance alerts, and ATM status changes.
- **Integrated Email APIs**: Notifications are triggered automatically based on user actions and system events, ensuring timely communication.

5. Maintenance Management

- **Maintenance Logging**: The system tracks maintenance activities performed on ATMs, including dates, descriptions, and technician details.
- **Alerts for Scheduled Maintenance**: Administrators receive reminders for upcoming maintenance tasks to ensure ATMs remain operational.

6. Security Features

- **Fraud Detection**: The system implements measures to detect suspicious activities and potential fraud attempts, enhancing overall security.
- **Session Management**: User sessions are securely managed with automatic timeouts to protect against unauthorized access.

7. Reporting and Analytics

- **Comprehensive Reports**: Administrators can generate detailed reports on ATM performance, transaction volumes, and user activity, aiding in decision-making.
- **Data Visualization**: Visual reports (e.g., charts and graphs) are created using JavaScript libraries to present data in an easily interpretable format.

8. Responsive User Interface

- **User-Friendly Design**: The interface is designed with a mobile-first approach using frameworks like Bootstrap, ensuring compatibility across various devices and screen sizes.
- **Intuitive Navigation**: The UI facilitates easy navigation, allowing users to access functionalities without confusion.

9. Audit Logging

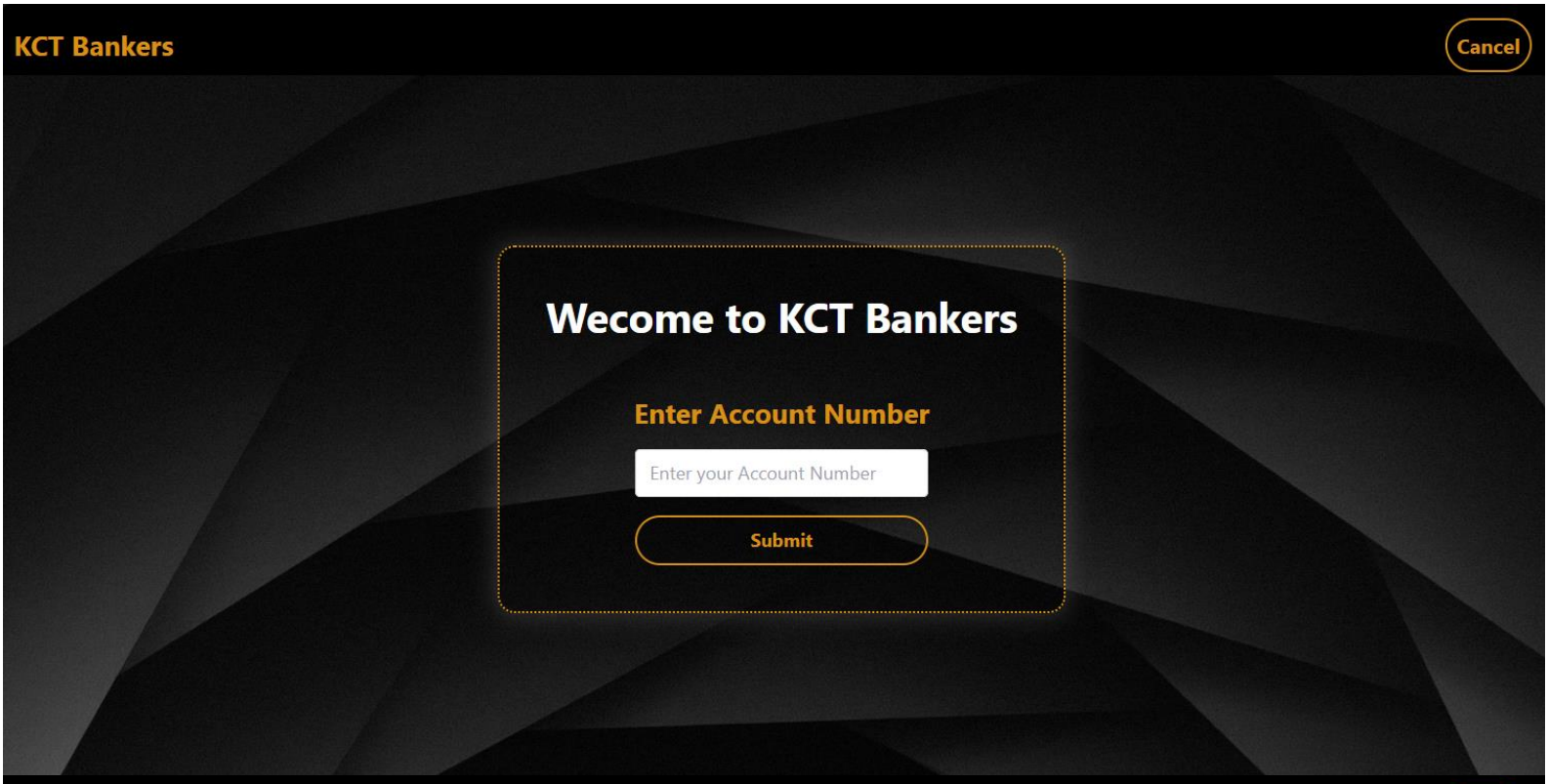
- **Activity Tracking**: All user actions are logged, providing an audit trail for accountability and security.
- **Access to Audit Logs**: Administrators can view and analyze logs to identify trends or potential security issues.

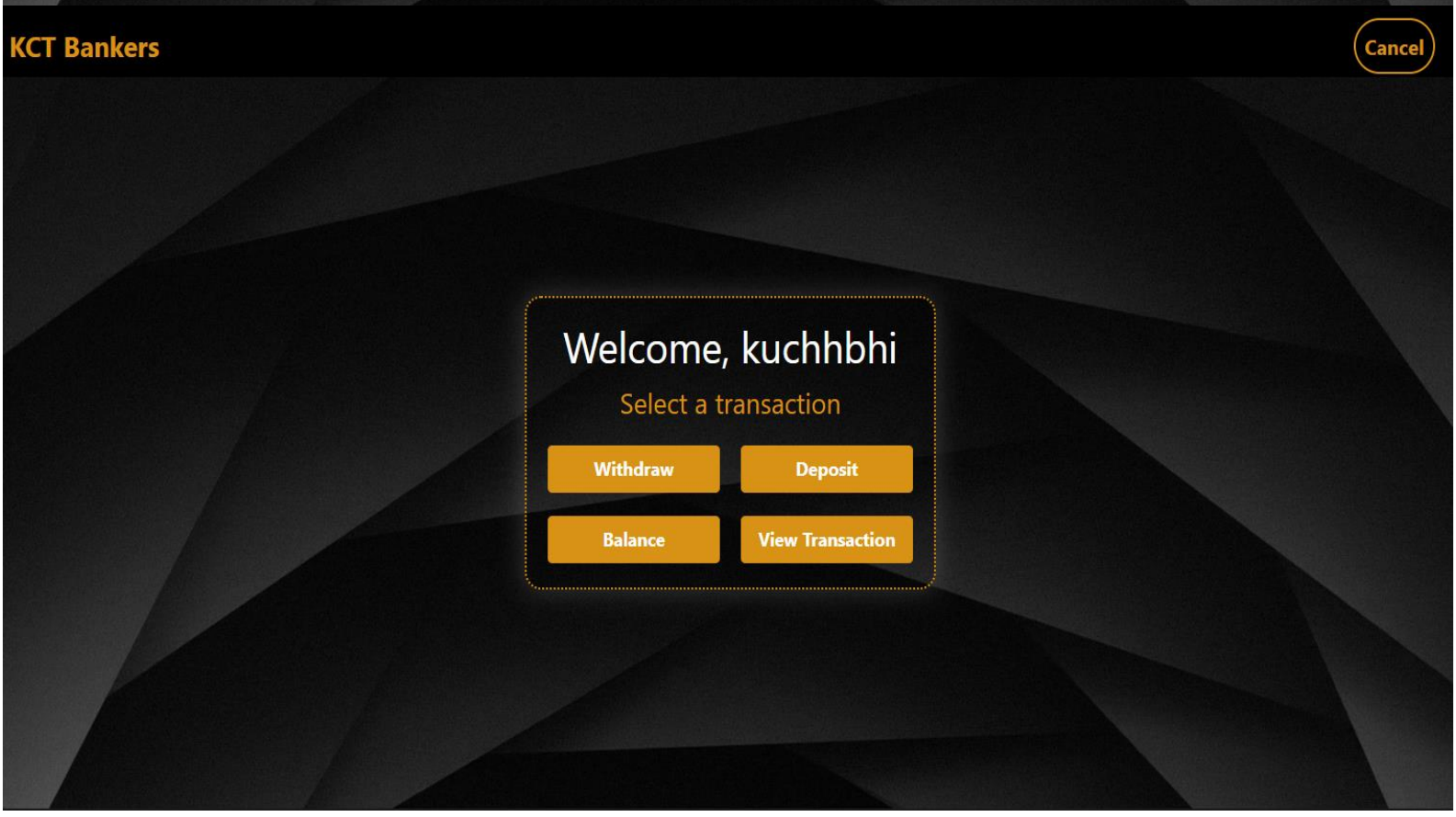
5. Security Considerations

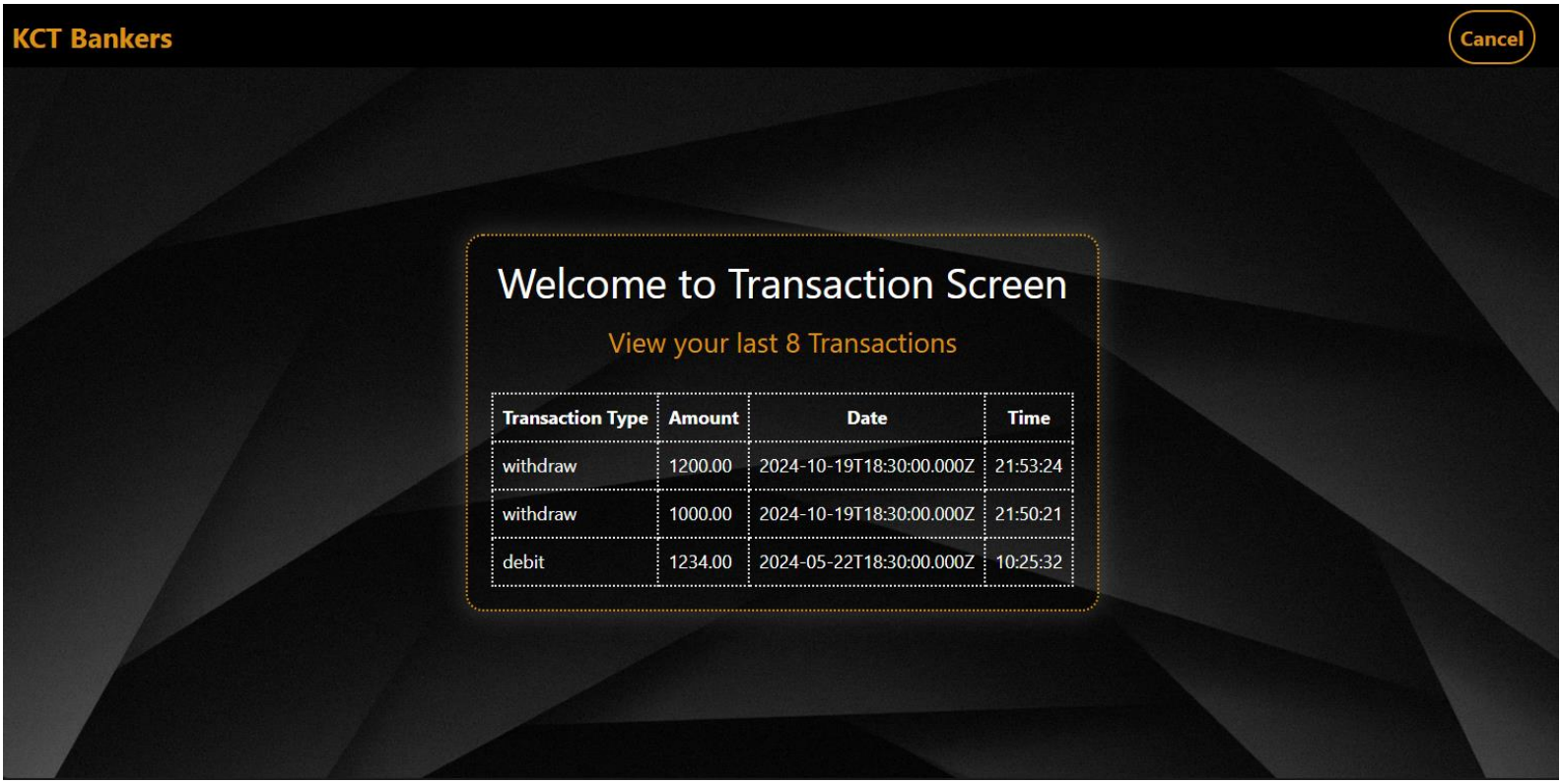
Security is a critical aspect of the ATM Management System. The system implements several measures to protect data and ensure the integrity of the application.

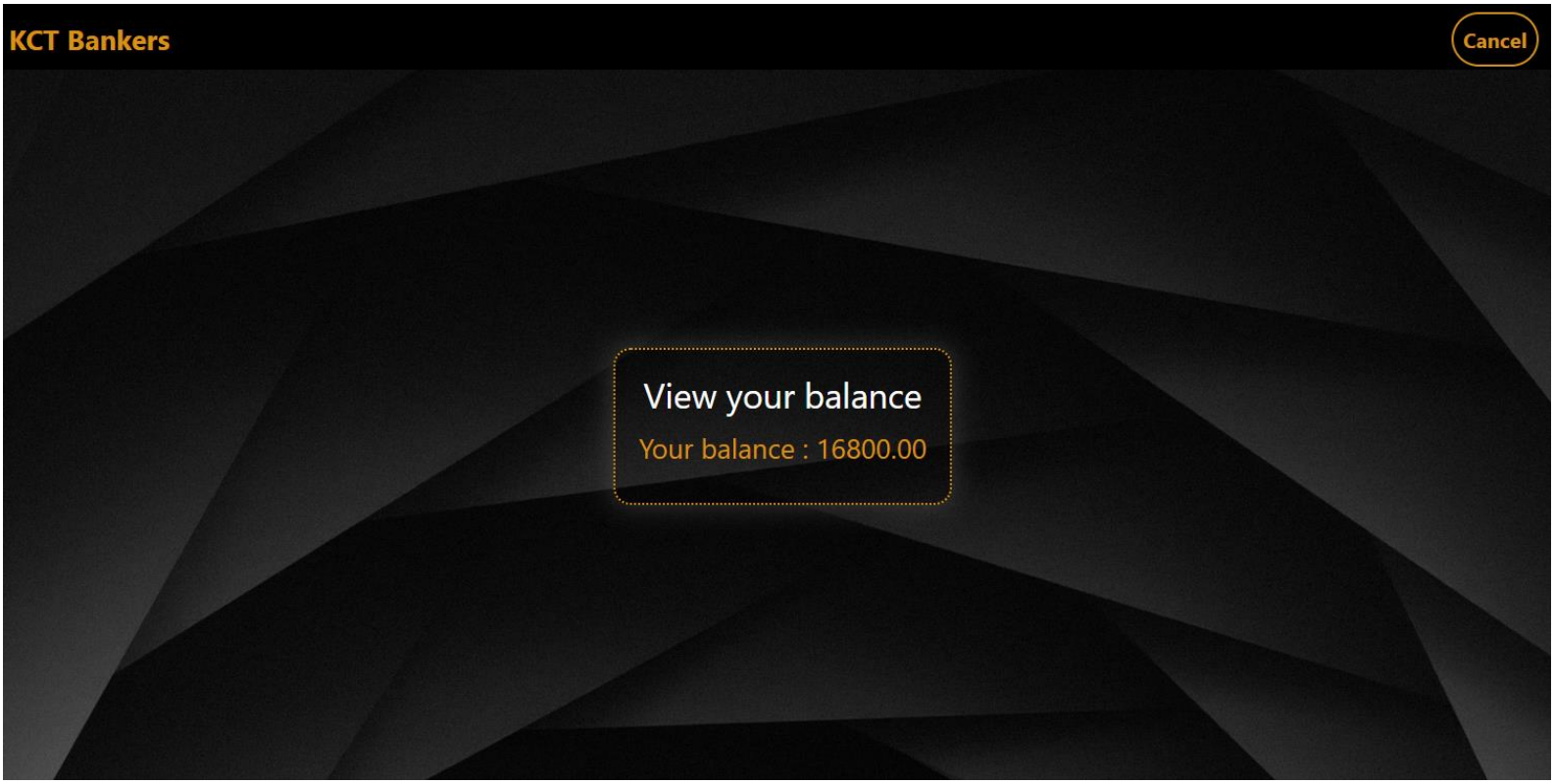
- **SQL Injection Prevention**: All database queries use prepared statements and parameterized queries to prevent SQL injection attacks.
- **Cross-Site Scripting (XSS) Protection**: Input sanitization and escaping outputs to prevent malicious scripts from being executed in the user's browser.
- **Password Hashing**: User passwords are hashed using bcrypt, making it nearly impossible for attackers to retrieve original passwords even in case of a data breach.
- **HTTPS/SSL**: All communication between the client and server is encrypted using SSL certificates to protect sensitive data.

PROJECT OUTPUT






KCT Bankers
Cancel
Welcome to Transaction Screen
View your last 8 Transactions
Transaction Type
Amount
Date
Time
withdraw
1200.00
2024-10-19T18:30:00.000Z
21:53:24
withdraw
1000.00
2024-10-19T18:30:00.000Z
21:50:21
debit
1234.00
2024-05-22T18:30:00.000Z
10:25:32


KCT Bankers
Cancel
View your balance
Your balance : 16800.00

Introduction/Overview of Project:

Objectives

- **User Authentication: Ensure secure access for users through PIN verification.**
- **Transaction Management: Enable withdrawals, deposits, balance inquiries, and fund transfers.**
- **Account Management: Handle customer account details and transaction history.**
- **Error Handling: Manage system errors and provide user-friendly messages.**
- **Reporting: Generate reports for transactions, user activity, and system performance.**

Components

1. **User Interface (UI):**
    - **Simple and intuitive screens for users to interact with.**
    - **Options for selecting transactions, entering amounts, and viewing balances.**
2. **Backend System:**
    - **Database Management: Store user data, account details, transaction records, and logs securely.**
    - **Server Logic: Handle requests from the ATM interface, process transactions, and communicate with the database.**
3. **Security Features:**
    - **Encryption for sensitive data (e.g., PINs and account numbers).**
    - **Session management to prevent unauthorized access.**
4. **Hardware Integration:**
    - **Connect with ATM hardware components like card readers, cash dispensers, and receipt printers.**
    - **Ensure compatibility with various ATM models.**
5. **Network Connectivity:**
    - **Facilitate communication between ATMs and the central banking system.**
    - **Implement protocols for secure data transfer.**

Technologies Used

- **Programming Languages: Java, C#, Python, etc., for backend development.**
- **Database Systems: MySQL, PostgreSQL, or Oracle for data storage.**
- **Web Technologies: HTML, CSS, JavaScript for any web-based UI.**
- **Security Protocols: SSL/TLS for secure communication.**

Challenges

- **Ensuring high availability and reliability.**
- **Maintaining security against fraud and cyber threats.**
- **Managing multiple concurrent transactions efficiently.**

Deliverables

- **A fully functional ATM management system.**
- **User and administrator documentation.**
- **Testing and deployment plans.**

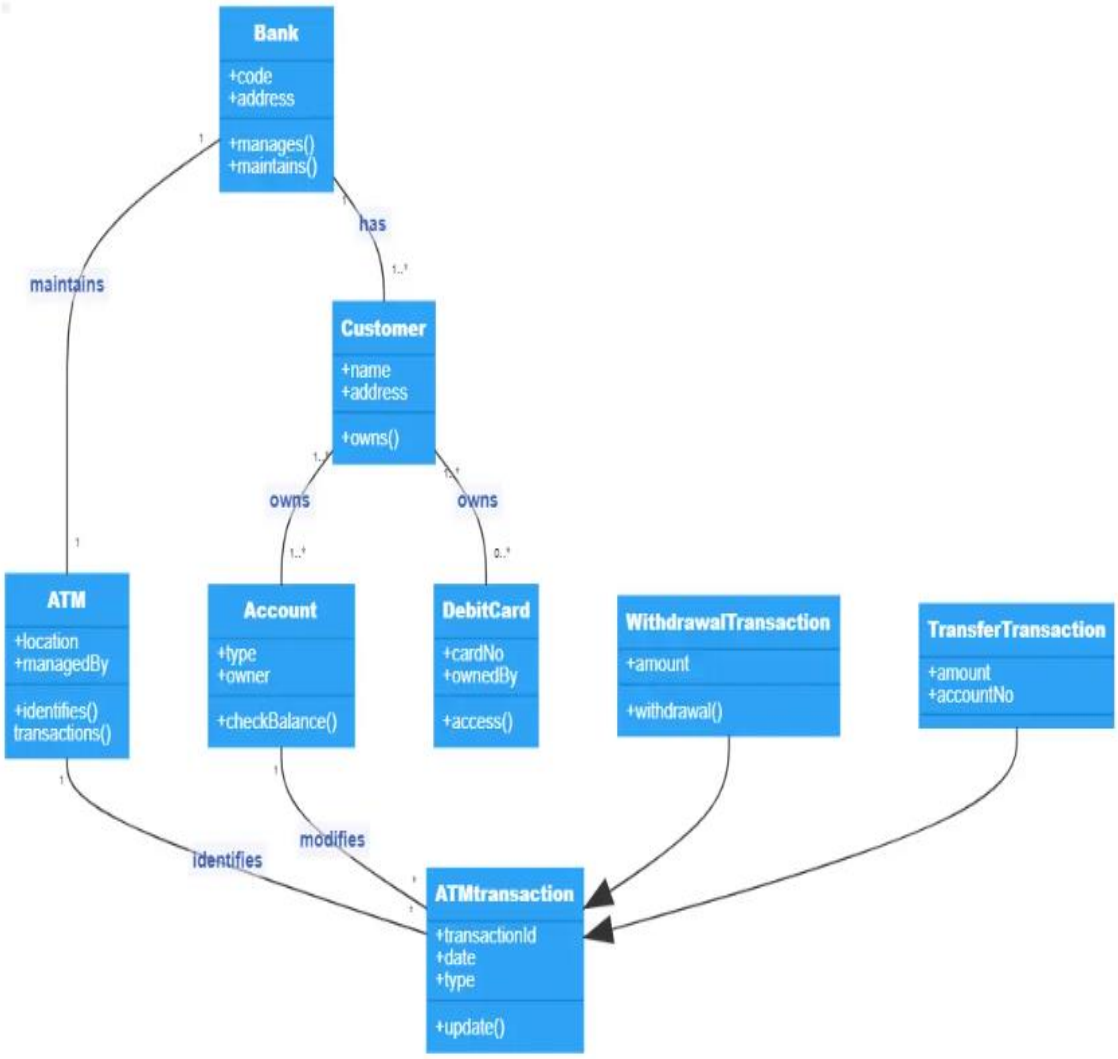
Future Enhancements

- **Mobile integration for account management.**
- **AI-driven analytics for transaction monitoring and fraud detection.**
- **Integration with additional services like bill payments and fund transfers.**

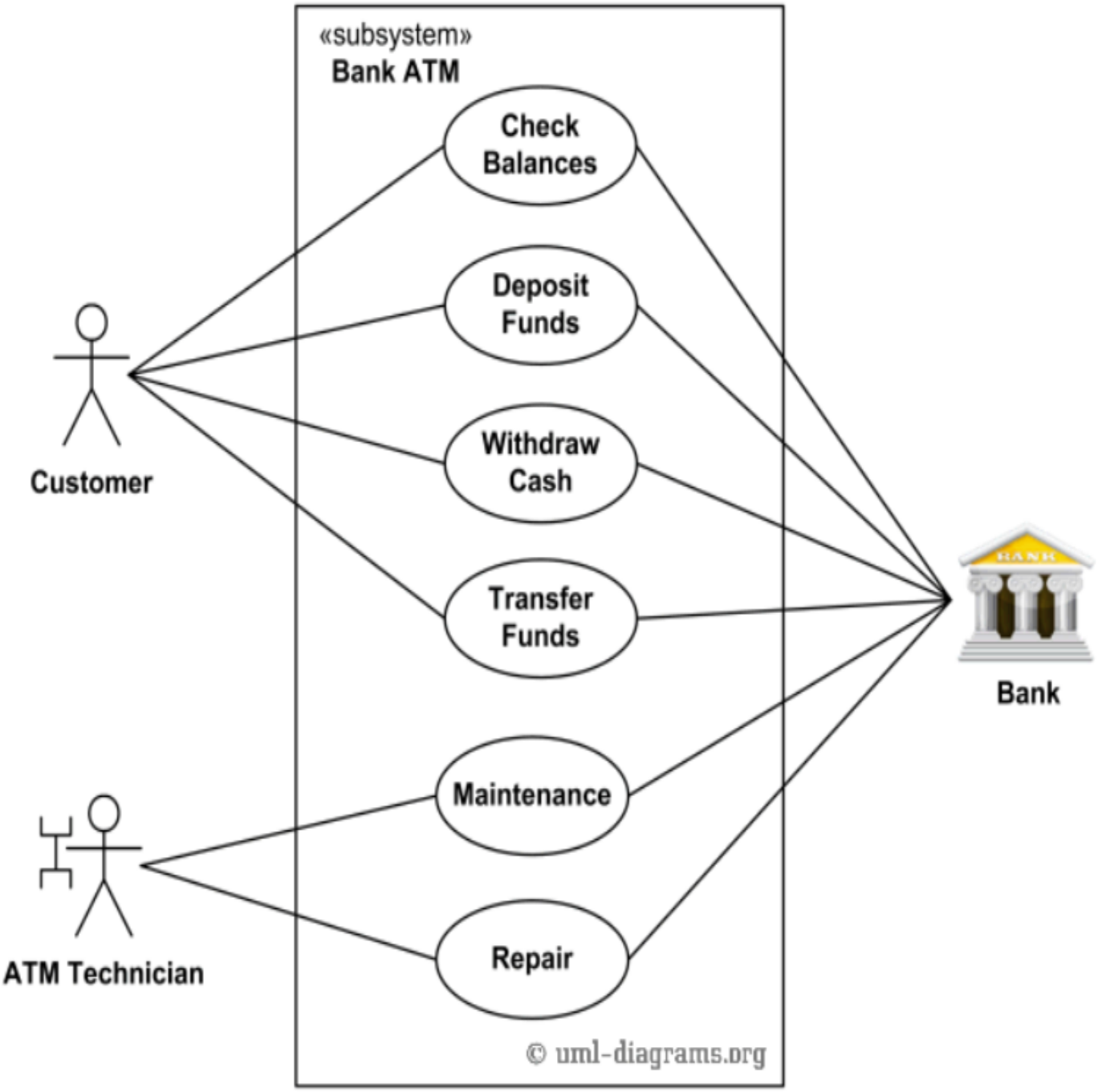
This overview provides a foundational understanding of an ATM management system project, focusing on its objectives, components, and technologies involved.

Functional Flowchart Diagram

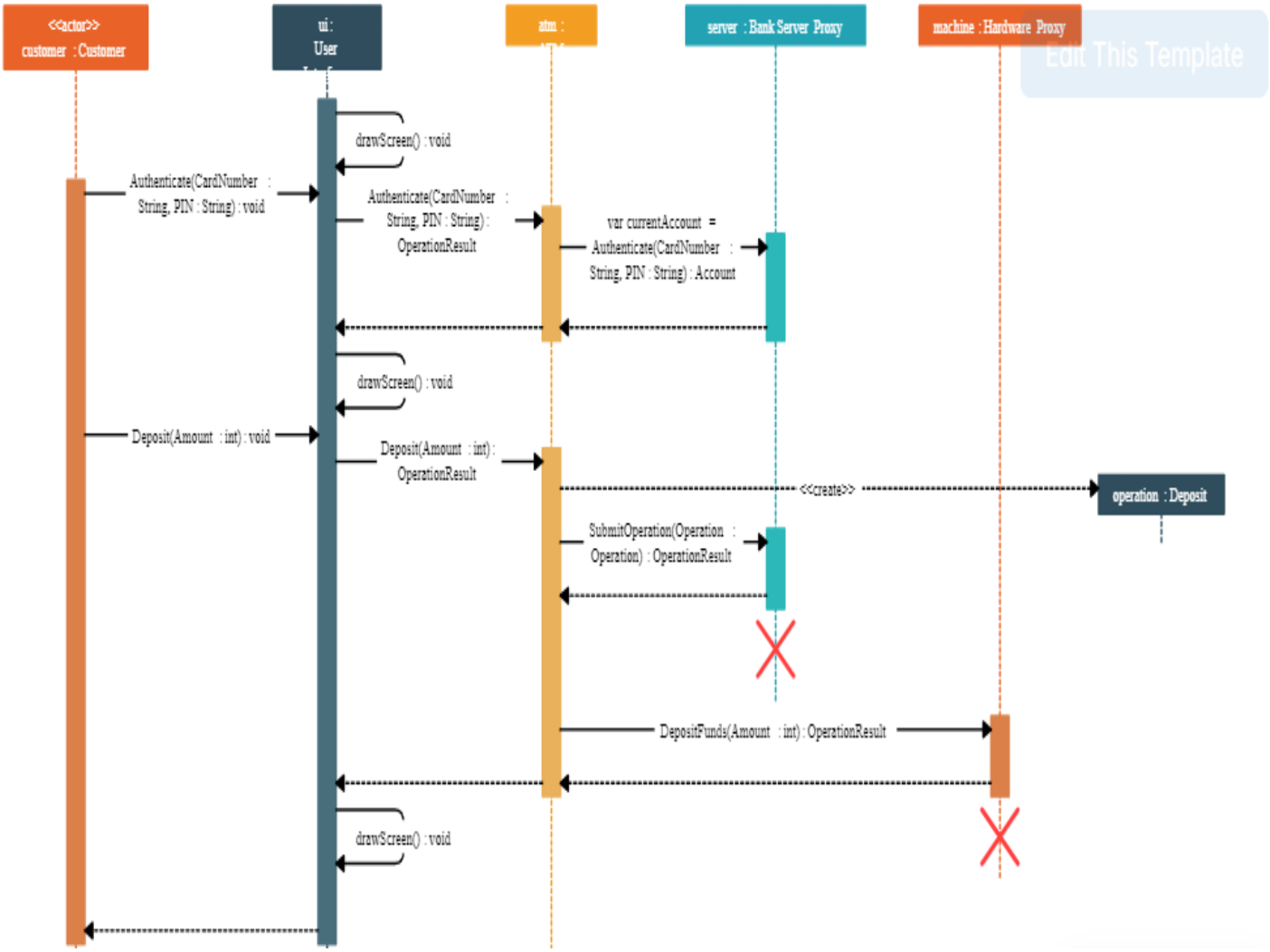
CLASS DIAGRAM



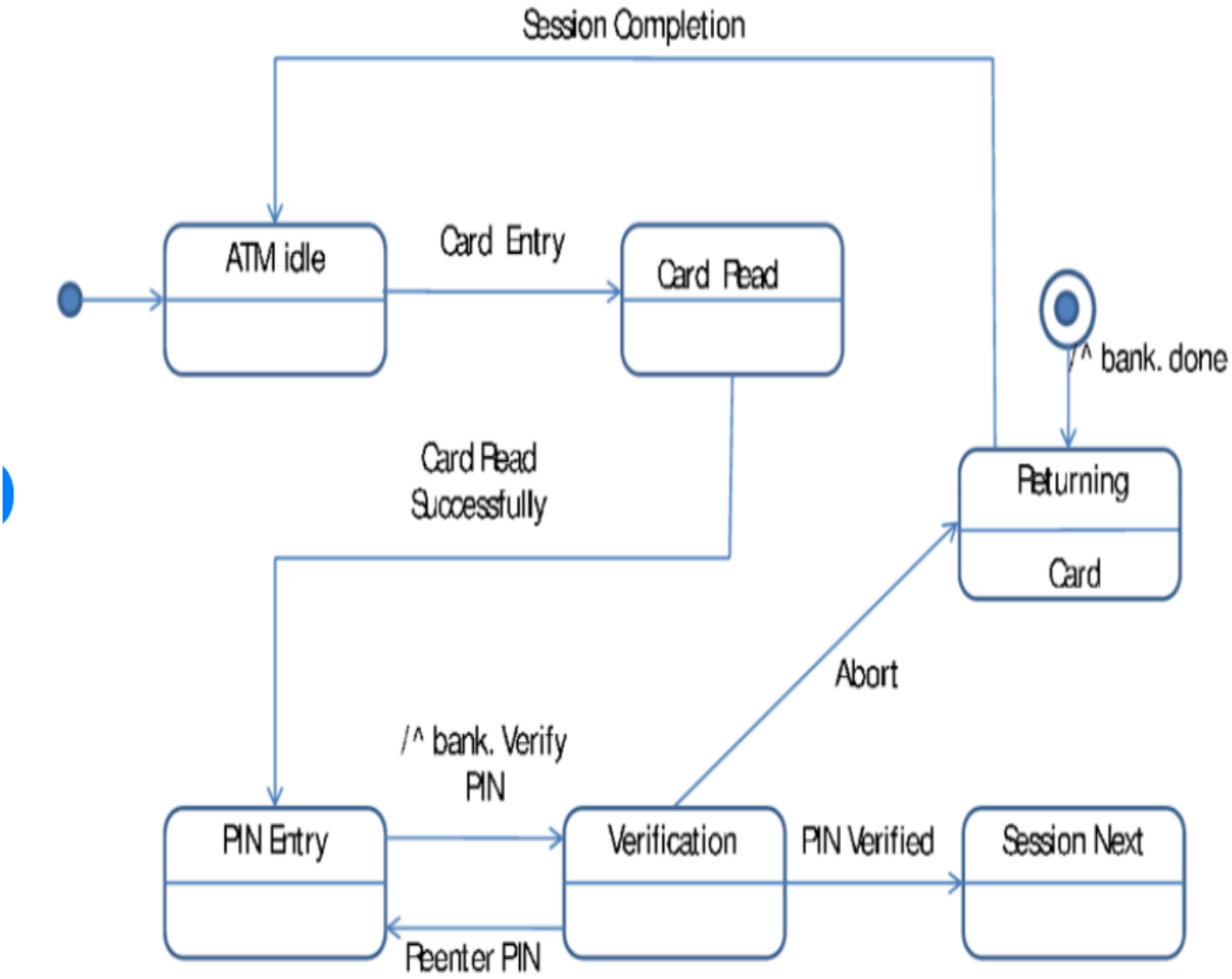
UseCase Diagram.



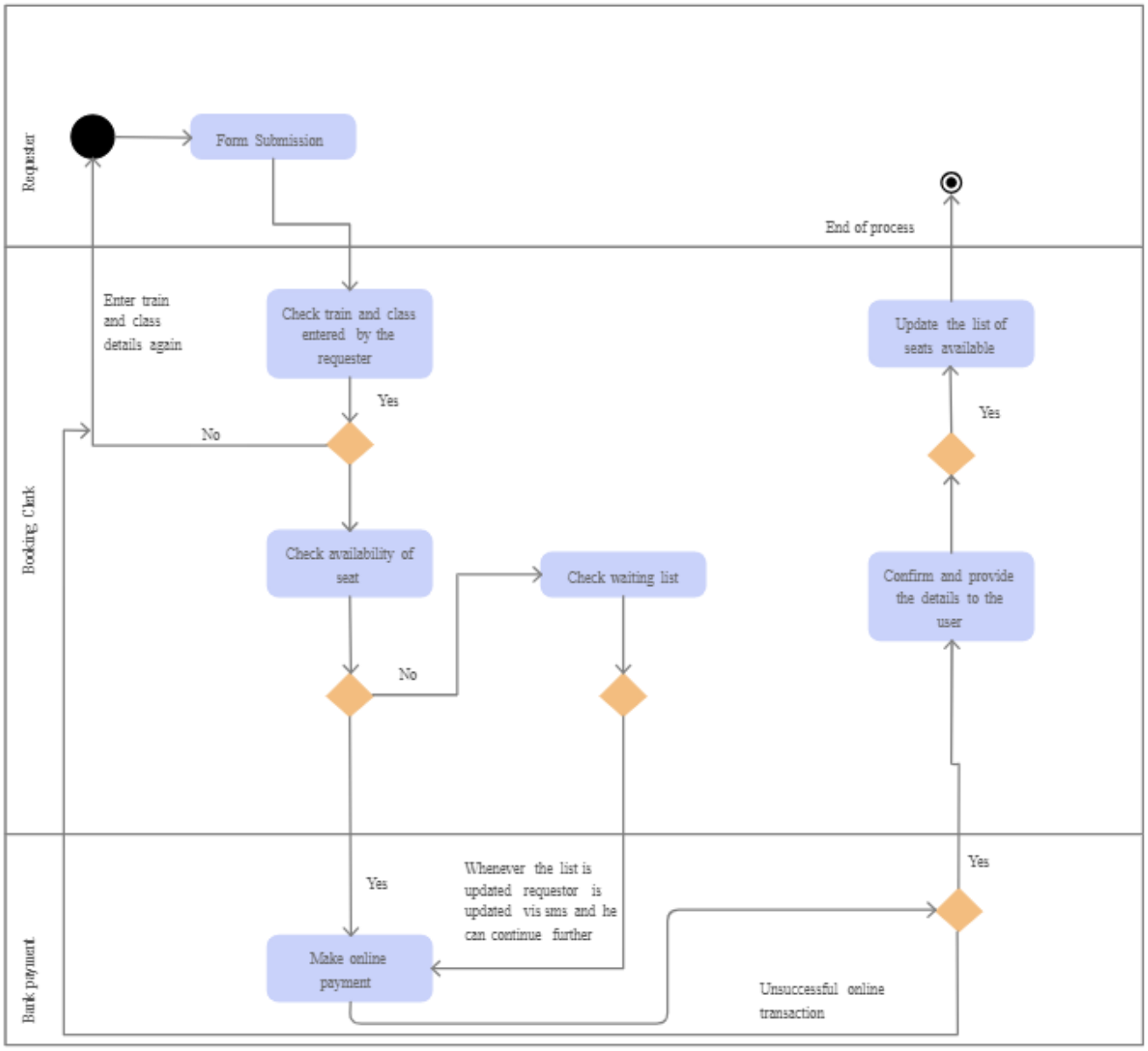
Sequence Diagram.



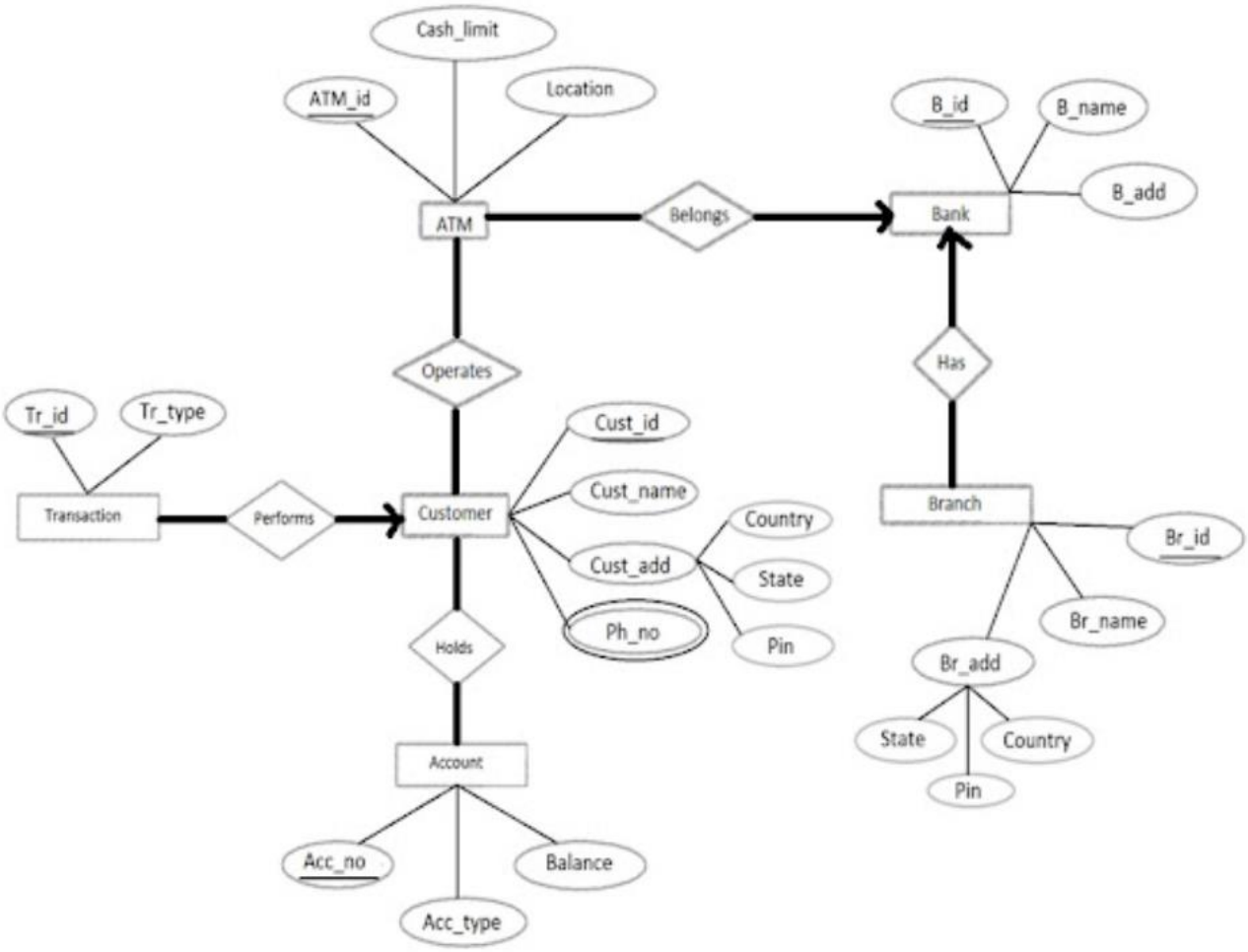
State Chart Diagram.



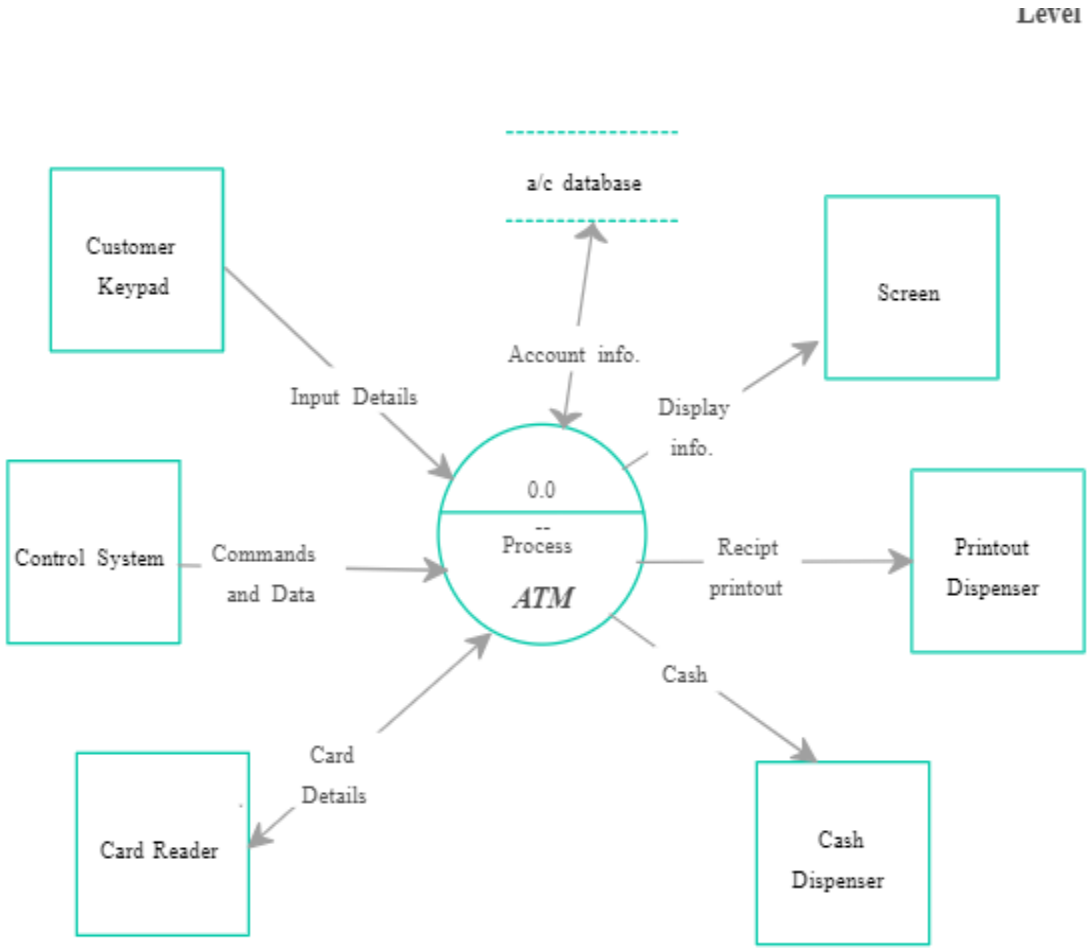
Activity Diagram.



ER Diagram.


Cash_limit
ATM_id
Location
ATM
Belongs
B_id
B_name
B_add
Bank
Operates
Has
Tr_id
Tr_type
Transaction
Performs
Customer
Cust_id
Cust_name
Cust_add
Country
State
Pin
Ph_no
Holds
Branch
Br_id
Br_name
Br_add
State
Country
Pin
Account
Acc_no
Balance
Acc_type

DF Diagram.



Software Requirements Specification (SRS)

A **Software Requirements Specification (SRS)** is a comprehensive, structured document that outlines the functional and non-functional requirements of a software system. It serves as a formal agreement between stakeholders—such as clients, developers, and project managers—detailing what the software system should do and the constraints under which it must operate.

Key Components of an SRS:

1. **Introduction**:
    - **Purpose**: Explains the purpose of the SRS and the intended audience.
    - **Scope**: Describes the software product being specified, its benefits, objectives, and goals.
    - **Definitions and Acronyms**: Provides explanations of terms and acronyms used in the document.

- **References**: Lists any external documents or resources referenced in the SRS.

2. **Overall Description**:
    - **Product Perspective**: Context of the product within the environment and its interfaces with other systems.
    - **Product Features**: High-level description of the product's capabilities.
    - **User Characteristics**: Information about the intended users and their requirements.
    - **Constraints**: Constraints that could affect the design or implementation (e.g., regulatory policies, hardware limitations).
3. **Functional Requirements**:
    - Detailed descriptions of the software functions, including inputs, processing, outputs, and error handling.
4. **Non-Functional Requirements**:
    - **Performance Requirements**: Speed, response time, and throughput expectations.
    - **Security Requirements**: Measures to protect data and ensure privacy.
    - **Usability Requirements**: User interface and user experience considerations.
    - **Reliability Requirements**: Availability, fault tolerance, and failure recovery.
5. **System Requirements**:
    - Hardware, software, network, and platform specifications.

Data Flow Diagram (DFD)

A **Data Flow Diagram (DFD)** is a graphical representation of the flow of data through a system, illustrating how data is processed, stored, and transferred between different components or entities. It is used to visualize the overall structure of a system or process, focusing on the movement of data rather than the physical or hardware aspects.

Key Components of a DFD:

1. **Processes**:
    - Represented by circles or rounded rectangles.
    - Depict the transformations, computations, or actions performed on data within the system.
2. **Data Stores**:
    - Represented by open-ended rectangles or two parallel lines.
    - Show where data is stored within the system, such as databases or files.

3. **Data Flows**:
    - Represented by arrows.
    - Indicate the movement of data between processes, data stores, and external entities.
    - Each data flow is labeled with the type of data being transferred.
4. **External Entities**:
    - Represented by squares or rectangles.
    - Denote external agents that interact with the system, such as users, organizations, or other systems.

Purpose of a DFD:

- **Understanding Requirements**: Helps stakeholders understand the system's processes and data flow.
- **System Analysis and Design**: Aids in identifying and defining system requirements and designing the system architecture.
- **Communication**: Facilitates clear communication among developers, analysts, and stakeholders.
- **Identifying Weaknesses**: Helps in spotting inefficiencies, redundancies, and potential bottlenecks in the system.

1. Conclusion

The ATM Management System (AMS) developed in this project has been designed to streamline and optimize the operations of Automated Teller Machines for financial institutions. This comprehensive web-based platform enables administrators to efficiently manage transactions, user access, and machine functionality while providing robust features for transaction tracking, reporting, and automated routine tasks.

By adhering to a three-tier architecture, the system effectively separates concerns into the presentation, business logic, and data layers, ensuring that each component operates efficiently and independently. This modularity enhances scalability and maintainability, allowing for future upgrades and features to be incorporated with minimal disruption to existing functionalities.

The system emphasizes user-centric design, ensuring that both customers and administrators can interact with the platform seamlessly. Features such as secure user authentication, transaction management, and account oversight enhance the overall user experience while reducing administrative overhead. Integrated security measures, including encryption and secure access controls, safeguard user data and ensure compliance with global data protection standards.

Moreover, the commitment to automation—such as automated transaction notifications, real-time monitoring, and reporting—demonstrates the system's potential to reduce manual workload, making ATM management more efficient and less error-prone. The reporting module empowers administrators with comprehensive insights into transaction patterns and machine performance, facilitating data-driven decision-making.

Overall, the AMS provides a scalable, flexible, and secure solution for managing ATM operations, with the added ability to be customized to fit various organizational needs. The architecture and feature set ensure that the system is future-proof and adaptable to evolving requirements.

4.Future Work

While the current version of the ATM Management System covers many essential features, there are several areas for improvement and expansion in future releases. The following future work and enhancements could further increase the system's value:

1. Mobile Application Development
    - Objective: Develop a dedicated mobile application for Atm and user to enhance accessibility and user experience.
    - Plan: The mobile app would provide features such as offline course viewing, mobile notifications, and quick progress tracking.
2. AI-Powered Learning Analytics
    - Objective: Integrate AI and machine learning to provide deeper insights into trainee performance and learning patterns.
    - Plan: Use predictive analytics to identify users who may be falling behind or excelling in their transaction. Additionally, AI could offer personalized course recommendations and insights into optimizing atm and user experience.
3. Gamification
    - Objective: Increase engagement and motivation for users by incorporating gamification elements.
    - Plan: Introduce features such as transactions,cash dispenser,check amount etc. These gamified elements can enhance user engagement, increase course completion rates, and provide a sense of accomplishment for users.
4. Enhanced Reporting and Data Visualization
    - Objective: Upgrade the current reporting module to offer more sophisticated data visualizations and dynamic report generation.
    - Plan: Incorporate real-time dashboards with advanced charts, graphs, and KPIs. Provide interactive reporting tools that allow admins to customize and drill down into data, helping them make more informed decisions based on trends and patterns in the training process.
5. Integrating Video Conferencing Tools
    - Objective: Enable seamless integration with video conferencing tools like Zoom, Microsoft Teams, or Google Meet to facilitate virtual training sessions.
    - Plan: Integrating these tools would allow the TMS to be a one-stop solution for both in-person and remote training, ensuring that virtual sessions are scheduled, tracked, and recorded within

the system.

5. References

1. **Agile Development Methodologies**

   Beck, K., Beedle, M., van Bennekum, A., et al. (2001). *Manifesto for Agile Software Development*. Retrieved from https://agilemanifesto.org

   This reference discusses the agile principles followed during the development cycle of the ATM Management System, emphasizing iterative development, continuous feedback, and adaptability.

2. **Three-Tier Architecture in Web Applications**

   Fowler, M. (2003). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

   This book outlines the architecture pattern used in the project, specifically the separation of concerns between the presentation, business logic, and data layers.

3. **User-Centric Design and Human-Computer Interaction**

   Norman, D. (2013). *The Design of Everyday Things* (Revised and Expanded Edition). Basic Books.

   This reference provides insights into designing systems with a focus on user experience, a principle applied throughout the ATM Management System development.

4. **Database Management and MySQL**

   DuBois, P. (2009). *MySQL Cookbook: Solutions for Database Developers and Administrators* (3rd Edition). O'Reilly Media.

5. **Data Privacy and Security Regulations**

   European Union. (2016). *General Data Protection Regulation (GDPR)*. Official Journal of the European Union, L 119/1.

   This regulation was referenced to ensure that the ATM Management System adhered to data protection laws, safeguarding personal information and user privacy.