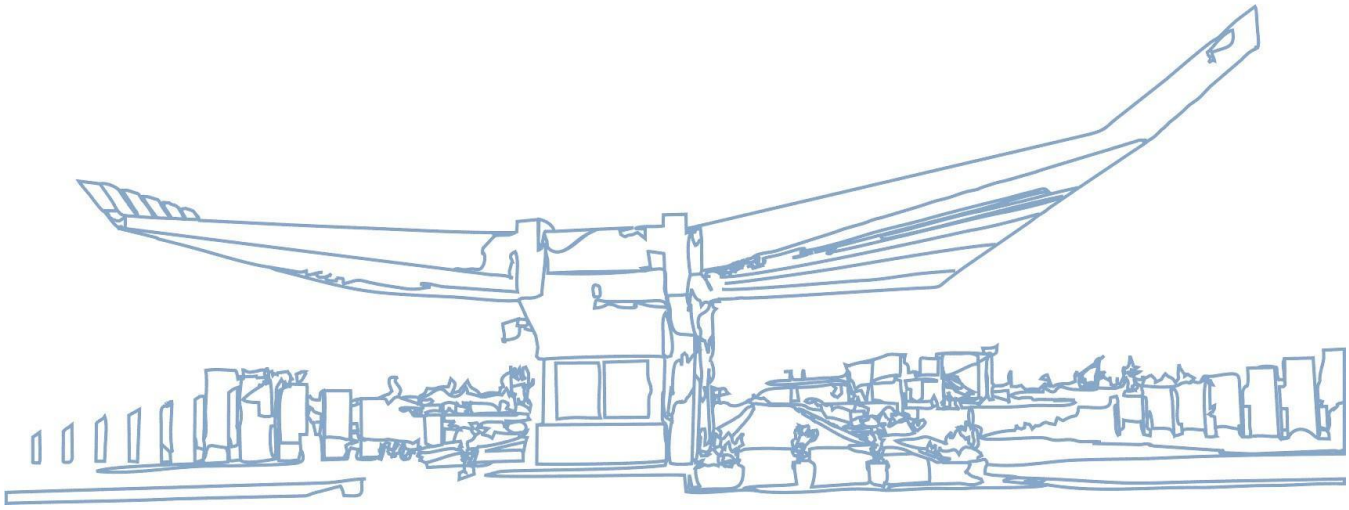
INTRODUCTION TO SOFTWARE ENGINEERING

[ATM Machine]



Team Members:

Maida Daulle

Aleksander Pacani

Tea Meraj

Irikli Karcini

[ATM Machine] Requirements Specification

Table of Contents

1. EXECUTIVE SUMMARY **1**

1.1 Project Overview 1
1.2 Purpose and Scope of this Specification

2. PRODUCT/SERVICE DESCRIPTION **2**

2.1 Product Context 4
2.2 User Characteristics 5
2.3 Assumptions 7
2.4 Constraints 8
2.5 Dependencies

3. REQUIREMENTS **9**

3.1 Functional Requirements 10
3.2 Non-Functional Requirements 13
3.2.1 User Interface Requirements 14
3.2.2 Usability 15
3.2.3 Performance 16
3.2.4 Manageability/Maintainability 18
3.2.5 Security 21
3.2.6 Standards Compliance 24
3.2.7 Other Non-Functional Requirements 25
3.3 Domain Requirements 27

4. **DESIGN THINKING METHODOLOGIES** **28**

4.1 Negotiation 28
4.2 Empathy 28
4.3 Noticing 28
4.4 GUI 28

5. **SOFTWARE DESIGN** **28**

5.1 Use Case 28
5.2 State Diagram 31
5.3 Class Diagram 32

APPENDIX **13**

Appendix A. Definitions, Acronyms, and Abbreviations 13
Appendix B. References 13
Appendix C. Organizing the Requirements 13

1. Executive Summary

1.1 Project Overview

This project represents the code for a simple ATM (Automated Teller Machine) system implemented in C. The project allows users to perform various banking operations, such as cash withdrawals, balance inquiries, and deposits. It maintains a linked list of accounts, where each account has an ID, a PIN, a balance, and a transaction history.

1.2 Purpose and Scope of this Specification

The purpose and scope of this code:

Purpose:

The purpose of this code is to simulate the functionality of an ATM (Automated Teller Machine). It enables users to carry out several banking tasks like cash deposits, balance checks, and cash withdrawals. Additionally, it provides transaction history functionality to display previous transactions.

Intended Audience:

The intended audience for this code is developers or individuals interested in understanding how an ATM system can be implemented using C programming language. It can serve as a learning resource or as a basis for building a more comprehensive ATM system.

Scope:

The scope of this code includes the following functionalities:

1. ***Account management:*** Loading account information from a file, including account ID, PIN, and balance.
2. ***Authentication:*** Prompting users for their ID and PIN to validate their access to the account.
3. ***Cash Withdrawal:*** Allowing users to withdraw cash from their account within specified withdrawal limits and available balance.

4. ***Balance Inquiry:*** Displaying the current balance of the user's account.
5. ***Deposit System:*** Allowing users to deposit cash or checks into their account, considering the deposit limit.
6. ***Transaction History:*** Providing options to display transaction history, including all transactions or a specific number of first or last transactions.

Out of Scope:

The following functionalities are outside the scope of this code:

1. ***User interface design:*** The code does not include any graphical user interface (GUI) elements. It assumes a command-line interface for user interaction.
2. ***Network communication:*** The code does not handle communication with a banking server or handle transactions with external systems.
3. ***Advanced security features:*** While the code implements basic account authentication with PIN, it does not cover advanced security measures like encryption, secure protocols, or biometric authentication.
4. ***Additional banking features:*** The code focuses on basic ATM operations and does not include advanced banking functionalities like fund transfers, bill payments, or account opening/closing.

2. Product/Service Description

The provided code represents a basic implementation of an ATM (Automated Teller Machine) system in C programming language. It simulates the functionality of an ATM machine that allows users to perform banking operations such as cash withdrawals, balance inquiries, and cash deposits.

The code includes features such as account management, where account information (ID, PIN, balance)

can be loaded from a file. It provides authentication by prompting users for their ID and PIN to access their account.

Users can initiate cash withdrawals, specifying the desired amount. The code checks if the withdrawal amount is within the available balance and withdrawal limit, providing appropriate feedback to the user.

Balance inquiry functionality is available to display the current balance of the user's account.

The code also includes a deposit system that allows users to deposit cash or checks into their account. It checks if the deposit amount is within the specified deposit limit and updates the account balance accordingly.

Additionally, the code provides transaction history functionality. Users can choose to display all transactions or a specific number of first or last transactions, showcasing details such as transaction amount, type (withdrawal or deposit), and payment method (cash or check).

2.1 Product Context

The product context for the provided code is an ATM (Automated Teller Machine) system. ATMs are self-service banking machines that allow customers to perform various financial transactions without the need for human teller assistance. They provide convenient and accessible banking services to customers, offering features such as cash withdrawals, balance inquiries, fund transfers, and more.

The code simulates the behavior and functionality of an ATM machine, providing a software-based implementation of the system.

In a real-world setting, the code would typically be integrated with hardware components such as a card reader, keypad, display, and cash dispenser to create a functional ATM machine. The software would interact with these hardware components to process user inputs, authenticate users, communicate with the bank's backend systems, and perform the requested financial transactions.

The product context also involves the security aspect of an ATM system. In a production environment, stringent security measures are implemented to protect user data, prevent unauthorized access, and ensure the integrity of transactions. This may include encryption, secure communication protocols, user authentication mechanisms, and transaction monitoring.

Overall, the provided code represents a software implementation of an ATM system and can serve as a foundation for building a more comprehensive and secure ATM solution in conjunction with appropriate hardware components and backend systems integration.

2.2 User Characteristics

1. ***Student:***
    - ***Experience:*** Students may have varying levels of experience using ATMs. Some may be frequent users, while others may have limited exposure.
    - ***Technical expertise:*** Students are generally familiar with technology and can adapt quickly to new systems and interfaces.
    - ***General characteristics:*** Students may have limited financial resources and may use the ATM primarily for basic transactions such as cash withdrawals, balance inquiries, and occasional deposits. They may value convenience, speed, and ease of use in the ATM system.
2. ***Faculty:***
    - ***Experience:*** Faculty members may have moderate to extensive experience using ATMs, as they are likely to have had personal bank accounts for a considerable period.
    - ***Technical expertise:*** Faculty members may have varying levels of technical proficiency. Some may be tech-savvy, while others may have limited familiarity with advanced banking systems.
    - ***General characteristics:*** Faculty members may use the ATM for a range of transactions, including cash withdrawals, balance inquiries, deposits, and possibly fund transfers. They may require more advanced features such as transaction history and may prioritize security and reliability in the ATM system.

3. ***Staff:***
    - ***Experience:*** Staff members may have a range of experience using ATMs, similar to the general population.
    - ***Technical expertise:*** Staff members may have varying levels of technical proficiency. Some may be comfortable with technology, while others may have limited familiarity.
    - ***General characteristics:*** Staff members may use the ATM for various transactions, including cash withdrawals, balance inquiries, and deposits. They may prioritize efficiency and reliability in the ATM system, as they may have limited time for banking transactions during their workday.
4. ***Other:***
    - ***Experience:*** This category includes users who don't fall into the student, faculty, or staff categories, such as retirees, self-employed individuals, or individuals from different professions.
    - ***Technical expertise:*** Technical expertise can vary widely within this category, ranging from those who are proficient with technology to those who may have limited experience.
    - ***General characteristics:*** Users in this category may have diverse banking needs and transaction patterns. They may require a range of services from the ATM, including cash withdrawals, balance inquiries, deposits, and other transaction types. The priorities and preferences of this group may vary greatly depending on their individual circumstances.

User research and analysis should be conducted to gain a deeper understanding of the target user base and their specific needs and expectations.

2.3 Assumptions

Assumptions:

1. ***Availability of ATMs:*** It is assumed that there will be an adequate number of functioning ATMs available for users to access the banking services provided by the system.
2. ***Basic user expertise:*** It is assumed that users will have basic familiarity with ATMs and understand common operations such as inserting a card, entering a PIN, and navigating through the ATM interface.

3. ***Network connectivity:*** It is assumed that the ATMs will have a stable and reliable network connection to communicate with the banking system and process transactions.
4. ***Security measures:*** It is assumed that appropriate security measures, such as encryption and authentication, are in place to protect user data and transactions during communication between the ATM and the banking system.
5. ***Availability of funds:*** It is assumed that the ATMs will have sufficient funds available to fulfill cash withdrawal requests made by users.
6. ***Compliance with banking regulations:*** It is assumed that the system will adhere to relevant banking regulations and compliance standards, ensuring that the transactions and services provided are in line with legal requirements.
7. ***Availability of support:*** It is assumed that there will be a support system in place to address any issues or problems encountered by users while using the ATM system.
8. ***Standard hardware and software:*** It is assumed that the ATMs will be equipped with standard hardware components (e.g., card reader, keypad, screen) and software systems that are compatible with the specified operating system.
9. ***User privacy:*** It is assumed that user privacy will be respected, and their personal and financial information will be securely stored and handled in accordance with applicable privacy laws and regulations.
10. ***Adequate system performance:*** It is assumed that the ATM system will provide acceptable response times and perform reliably to meet the demands of users during normal operating conditions.
11. ***Currency and denominations:*** It is assumed that the ATMs will support the local currency and provide appropriate denominations for cash withdrawal based on the availability of funds.

2.4 Constraints

Constraints:

1. ***Parallel operation with an old system:*** The new system needs to be able to operate in parallel with an existing or legacy system for a certain period of time during the transition phase. This may require compatibility with the old system's data formats or APIs to ensure smooth data exchange and integration.
2. ***Audit functions:*** The system should incorporate audit functions, such as an audit trail and log files, to track and record user activities, system events, and transaction details. These audit functions are necessary for security, compliance, and troubleshooting purposes.

Access, management, and security: The system should have appropriate access controls and user management features to ensure that only authorized individuals can perform specific actions and access sensitive information. It should also comply with security standards and protocols to protect user data, prevent unauthorized access, and mitigate security risks.

Criticality of the application: The importance and criticality of the application should be considered during the design process. The system should be designed to handle high availability, reliability, and fault tolerance to minimize downtime and ensure uninterrupted service to users.

System resource constraints: The design should take into account any resource limitations or constraints, such as disk space, memory, processing power, or network bandwidth. Optimization techniques may need to be applied to ensure efficient resource utilization and performance.

Design standards and frameworks: The system may need to adhere to specific design standards or frameworks mandated by the organization or industry. This could include using a particular programming language, following coding conventions, or leveraging specific design patterns or architectures.

Compliance requirements: The system should comply with relevant legal, regulatory, and industry standards, such as data protection regulations, financial regulations, and accessibility guidelines. Design choices should align with these requirements to ensure compliance.

Hardware and software compatibility: The system design should consider compatibility with specific hardware components, operating systems, databases, or third-party software that are part of the overall technology stack. This ensures that the system can be deployed and integrated effectively within the existing infrastructure.

Scalability and extensibility: The system should be designed to accommodate future growth, scalability, and extensibility requirements. It should be able to handle increasing user load, additional features, and potential system expansions without significant rework or performance degradation.

Time and budget constraints: The design should be mindful of time and budget limitations to ensure feasibility and cost-effectiveness. This may involve prioritizing essential features, optimizing development processes, or making trade-offs between functionality, quality, and resources.

2.5 Dependencies

Dependencies:

1. ***External data sources:*** The code may have dependencies on external data sources or APIs, such as data feeds, databases, or web services. The availability and reliability of these data sources may impact the requirements and functionality of the system.
2. ***Third-party libraries or frameworks:*** The code may rely on specific third-party libraries or frameworks to provide certain functionality or to integrate with other systems. The availability and compatibility of these dependencies may influence the design and implementation of the system.
3. ***Hardware or infrastructure dependencies:*** The code may have dependencies on specific hardware components or infrastructure, such as servers, network equipment, or cloud services. The availability, configuration, and performance of these dependencies may affect the requirements and deployment of the system.
4. ***Sequential dependencies:*** Certain modules or components of the code may have sequential dependencies, where one module needs to be completed or available before another module can be built or integrated. These dependencies should be identified and managed to ensure a smooth development and deployment process.

5. ***Data synchronization or integration:*** If the system needs to synchronize or integrate data with other systems or databases, the availability and compatibility of those systems may impact the design and requirements of the system.
6. ***Platform or operating system dependencies:*** The code may have dependencies on specific platforms or operating systems, such as Windows, Linux, or mobile operating systems. The compatibility and support for these platforms may influence the design and functionality of the system.

7. ***Service-level agreements (SLAs):*** If the system relies on external services or APIs that are governed by SLAs, the requirements of the system may need to align with those SLAs to ensure compliance and desired performance.
8. ***Development dependencies:*** The code may have dependencies on other development teams or projects, where certain functionalities or interfaces need to be provided by other teams or modules. These dependencies should be coordinated and managed to ensure timely development and integration.

3. Requirements

Design and implement an ATM machine simulation that allows users to withdraw, deposit, and transfer funds.

- ***User Interface:*** The ATM Machine should have a user-friendly interface that is easy to use and navigate. The UI should include options for cash withdrawals, balance inquiries, and other relevant banking services.
- ***Account Management:*** The ATM Machine should be able to authenticate and authorize account holders. It should verify account details such as account number and PIN, and display account information such as account balance and transaction history.
- ***Cash Withdrawal:*** The ATM Machine should allow account holders to withdraw cash from their account. The system should check for available balance, and dispense cash as per the request of the user.

- ***Balance Inquiry:*** The ATM Machine should allow account holders to check their account balance.
- ***Deposit System:*** The ATM Machine should allow account holders to deposit cash or checks into their account. It should verify the validity of the deposit and update the account balance accordingly.

- ***Transaction History:*** The ATM Machine should keep a record of all transactions made by account holders.
- ***Security:*** The ATM Machine should have robust security measures to prevent unauthorized access and fraudulent transactions.
- ***Error Handling:*** The ATM Machine should handle errors such as invalid input and memory errors.
- ***Testing:*** The ATM Machine should be thoroughly tested to ensure that it works correctly and accurately.
- ***Documentation:*** The project should be well-documented, including user manuals, source code documentation, and project reports.

3.1 Functional Requirements

Functional Requirements for the ATM Machine Simulation:

User Interface:

a. The ATM Machine should provide a user-friendly interface that is easy to navigate and understand.

b. The interface should display options for cash withdrawals, balance inquiries, and other relevant banking services.

c. The interface should provide clear instructions and feedback to guide the user through each transaction.

Account Management:

a. The ATM Machine should authenticate and authorize account holders by verifying their account number and PIN.

b. It should display account information such as the account balance and transaction history after successful authentication.

c. The system should maintain the confidentiality and integrity of account information, ensuring that only authorized users can access it.

Cash Withdrawal:

a. The ATM Machine should allow account holders to withdraw cash from their accounts.

b. It should check the available account balance before processing the withdrawal request.

c. The machine should dispense the requested amount of cash if the account has sufficient funds.

d. The system should update the account balance after a successful withdrawal transaction.

Balance Inquiry:

a. The ATM Machine should provide an option for account holders to check their account balance.

b. It should display the current account balance on the screen after the user selects this option.

Deposit System:

a. The ATM Machine should enable account holders to deposit cash or checks into their accounts.

b. It should validate the deposit to ensure that the cash or check is genuine and properly endorsed.

c. The system should update the account balance after a successful deposit transaction.

Transaction History:

a. The ATM Machine should maintain a record of all transactions made by account holders.

b. It should store transaction details such as date, time, type of transaction, and amount.

c. The system should allow users to access their transaction history through the user interface.

Security:

a. The ATM Machine should implement robust security measures to prevent unauthorized access.

b. It should require users to authenticate themselves with a valid account number and PIN.

c. The system should encrypt sensitive data, such as PIN numbers, to protect against unauthorized interception.

d. It should have mechanisms in place to detect and prevent fraudulent activities, such as card skimming or tampering.

Error Handling:

a. The ATM Machine should handle errors effectively, providing appropriate error messages to users.

b. It should handle scenarios such as invalid user input, insufficient funds, or system errors.

c. The system should gracefully recover from errors and maintain the integrity of account data.

Testing:

a. The ATM Machine should undergo thorough testing to ensure that it functions correctly and accurately.

b. It should be tested for different scenarios, including edge cases and stress testing.

c. The testing process should include both functional and non-functional testing to verify the system's reliability and performance.

Documentation:

a. The project should be well-documented, including user manuals that provide instructions on how to use the ATM Machine.

b. The source code should be adequately documented to facilitate future maintenance and enhancements.

c. A project report should be created to document the design, implementation, and testing of the ATM Machine simulation.

3.2 Non-Functional Requirements

Product Requirements

Usability: The ATM Machine should have an intuitive and user-friendly interface.The system should provide clear instructions and feedback to guide users through transactions.

Reliability: The ATM Machine should be available for use 24/7, with minimal downtime for maintenance. The system should handle high volumes of transactions without significant performance degradation.

Performance: The ATM Machine should process transactions quickly and efficiently. Response times for user interactions should be within acceptable limits.

Security: The system should have robust security measures to prevent unauthorized access and fraudulent activities. Sensitive data, such as PIN numbers and account information, should be encrypted and protected.

Maintainability: The codebase should be well-structured, modular, and easily maintainable. Updates and enhancements should be straightforward to implement without disrupting the system's functionality.

Organizational Requirements:

Compliance: The ATM Machine should comply with relevant banking and financial regulations. It should adhere to security standards and best practices in the industry.

Documentation: The project should be thoroughly documented, including user manuals, technical documentation, and system architecture. Documentation should be comprehensive and up-to-date to facilitate system understanding and future maintenance.

Training and Support: Adequate training should be provided to users and support staff on operating the ATM Machine. A support mechanism should be in place to assist users with any issues or questions that may arise.

External Requirements

Hardware and Software Compatibility: The ATM Machine should be compatible with a wide range of hardware components, such as card readers and cash dispensers. The system should run on standard operating systems and be compatible with commonly used software libraries and frameworks.

Network Connectivity: The ATM Machine should be able to establish and maintain a secure connection with the banking network for transaction processing. The system should handle intermittent or unstable network connections without data loss or compromise.

Accessibility: The ATM Machine should comply with accessibility guidelines, ensuring it can be used by individuals with disabilities. The interface should provide appropriate support for users with visual impairments or other accessibility needs.

Scalability: The system should be designed to handle future growth and accommodate an increasing number of users and transactions. It should be scalable in terms of hardware resources and processing capabilities.

3.2.1 User Interface Requirements

User Interface Requirements

Screen Formats/Organization: The ATM Machine interface should have a clear and organized layout, displaying relevant information prominently. Screens should be designed to minimize clutter and prioritize essential information. Menu options and functions should be logically grouped and easily accessible.

Report Layouts: The system should provide clear and concise transaction receipts for users. Receipts should include transaction details such as date, time, transaction type, account balance, and any applicable fees.

Menu Structures: The main menu should provide clear options for different banking services, such as cash withdrawals, balance inquiries, and deposits. Submenus should be used to present additional options related to each service. Navigation within menus should be intuitive and user-friendly.

Error and Other Messages: Error messages should be displayed in a user-friendly and understandable format, clearly indicating the nature of the error and any corrective actions. Messages should be concise, informative, and presented in a way that minimizes confusion. The system should provide helpful prompts and suggestions to guide users through any errors or exceptional conditions.

Function Keys: The ATM Machine should support function keys for quick access to common operations. Function keys should be labeled appropriately and perform predefined actions, such as canceling a transaction or navigating back to the previous screen.

Visual Design: The user interface should have a visually appealing and professional design. Color schemes, fonts, and graphics should be chosen carefully to enhance readability and usability. Adequate contrast and font sizes should be used to ensure accessibility for users with visual impairments.

Input Validation: The system should validate user inputs to prevent errors and ensure data integrity. Input fields should have appropriate validation rules and provide feedback on any invalid or incorrect inputs.

Multi-Language Support: The ATM Machine should support multiple languages to accommodate users with different language preferences. Language selection options should be provided in the user interface.

3.2.2 Usability

Learnability: The ATM Machine system should be easy to learn for new users with minimal training. The user interface should be intuitive and self-explanatory, requiring minimal guidance to navigate and perform basic transactions. On-screen instructions and prompts should be clear and concise, guiding users through the process step-by-step. The system should provide informative tooltips or hints for complex or unfamiliar operations.

User Documentation: The user documentation should be comprehensive and provide clear instructions on how to use the ATM Machine system. It should cover all features and functions of the system, explaining their purpose and providing detailed step-by-step guides. The documentation should be easily accessible and available in both digital and printable formats.

Context-Sensitive Help: The help system should be context-sensitive, meaning it provides relevant assistance based on the current user's task or location within the system. It should offer clear explanations of system features and provide guidance on how to achieve common tasks. The help system should be easily accessible from within the user interface and provide assistance without disrupting the user's workflow.

Error Handling and Recovery: The system should provide clear error messages when users encounter errors or invalid inputs. Error messages should be informative, explaining the issue and suggesting possible solutions. The system should guide users through the error recovery process, offering clear instructions on how to correct errors and resume normal operations.

Efficiency: The ATM Machine system should be designed to perform transactions and provide information quickly and efficiently. Response times should be optimized to minimize user wait times and provide a smooth user experience. Reducing unnecessary steps or clicks required to complete tasks will enhance efficiency.

Consistency: The user interface elements, terminology, and interactions should be consistent throughout the system. Common operations, such as confirming transactions or navigating menus, should use consistent patterns to reduce cognitive load and enhance usability. Consistency across the system will make it easier for users to understand and navigate the ATM Machine.

Accessibility: The system should adhere to accessibility guidelines to ensure that users with disabilities can access and use the ATM Machine. It should support assistive technologies, such as screen readers and keyboard navigation, to accommodate users with visual or motor impairments. The user interface should have sufficient color contrast, resizable fonts, and other accessibility features.

3.2.3 Performance

Static Numerical Requirements

1. ***Number of Terminals:*** The system should support a minimum of X terminals simultaneously.
2. ***Simultaneous Users:*** The system should support a minimum of Y concurrent users accessing the ATM Machine.

3. ***Information Handling:*** The system should be capable of handling a maximum of Z amount of customer account information, transaction history, and other relevant data.

Dynamic Numerical Requirements

1. ***Transaction Processing Time:*** 95% of transactions should be processed and completed within 2 seconds.
2. ***Response Time:*** The system should respond to user inputs within 1 second for normal workload conditions.
3. ***Peak Workload Capacity:*** The system should handle a minimum of X transactions per minute during peak usage periods.
4. ***Data Processing:*** The system should process and handle a minimum of Y amount of data per hour, including transaction data, account balances, and other related information.
5. ***Concurrent Transactions:*** The system should be capable of handling a minimum of Z concurrent transactions without performance degradation.

These performance requirements provide measurable targets for the system's ability to handle user interactions, process transactions, and manage data. By specifying specific timeframes and capacities, these requirements ensure that the system meets the performance expectations of users and can handle both normal and peak workload conditions effectively.

3.2.4 Manageability/Maintainability

Monitoring Requirements

1. ***Health Monitoring:*** The ATM Machine should have a built-in health monitoring system that continuously checks the status of its hardware components, software processes, and network connectivity. It should be able to detect any abnormalities or malfunctions and provide alerts for timely maintenance or repairs.
2. ***Failure Conditions:*** The system should have mechanisms in place to detect and handle various failure conditions. This includes detecting hardware failures (e.g., card reader, cash dispenser), software errors, network connectivity issues, and power outages. The system should be designed to gracefully handle these failures and minimize disruption to users.
3. ***Error Detection:*** The ATM Machine should have robust error detection mechanisms to identify and report any errors or inconsistencies during user interactions, transaction processing, or system operations. This includes validating user inputs, verifying account information, and detecting any discrepancies in transaction records or balances.
4. ***Logging:*** The system should maintain comprehensive logs of all activities, transactions, and system events. This includes logging user interactions, error messages, system alerts, and audit trails. The logs should capture relevant information for troubleshooting, analysis, and auditing purposes.
5. ***Error Correction:*** When errors or inconsistencies are detected, the ATM Machine should be capable of taking appropriate corrective actions. This may involve displaying error messages to the user, initiating rollback procedures, notifying system administrators or maintenance personnel, and recovering from system failures or data inconsistencies.
6. ***System Performance Monitoring:*** The ATM Machine should have performance monitoring capabilities to track key performance indicators (KPIs) such as response time, transaction throughput, system utilization, and error rates. This monitoring helps identify performance bottlenecks, optimize system resources, and ensure efficient operation.

By incorporating these monitoring requirements, the ATM Machine can proactively detect and address issues, ensure system availability and reliability, and provide a seamless user experience. The monitoring mechanisms enable timely detection of failures, effective error handling, and accurate logging for troubleshooting and analysis purposes.

Maintenance Requirements

1. ***Modularity:*** The system should be designed with a modular architecture, allowing for easy maintenance and updates. Each module or component should have well-defined boundaries and dependencies, enabling individual modules to be modified or replaced without impacting the entire system. This promotes flexibility and simplifies maintenance tasks.
2. ***Documentation:*** The system should be thoroughly documented to provide clear and comprehensive information on its design, functionality, and maintenance procedures. This includes detailed documentation of code structure, APIs, database schemas, configuration settings, and any other relevant technical information. The documentation should be easily accessible and kept up-to-date to support efficient maintenance.
3. ***Code Maintainability:*** The system's codebase should adhere to coding best practices, emphasizing readability, maintainability, and modifiability. This includes writing clean, well-structured code with meaningful variable and function names, proper code commenting, and following established coding conventions. Code maintainability helps reduce the effort required for troubleshooting, bug fixing, and future enhancements.
4. ***Error Handling and Logging:*** The system should have robust error handling mechanisms in place to capture and log errors encountered during runtime. Error messages should be informative and enable quick identification of the root cause. Detailed logs should be maintained to facilitate troubleshooting and debugging, aiding maintenance personnel in diagnosing and resolving issues.
5. ***Compatibility and Upgradability:*** The system should be designed to ensure compatibility with future technology advancements and changes. This includes considering potential hardware and software upgrades, ensuring backward compatibility with existing components or interfaces, and minimizing dependencies on specific technologies or versions. Upgradability should be a key consideration to facilitate future maintenance and enhancements.

6. ***Scalability:*** The system should be scalable to accommodate future growth and increasing demands. This includes designing the system to handle larger datasets, higher user loads, and increased transaction volumes. Scalability ensures that the system can be easily expanded or modified to meet changing business requirements without significant disruptions or costly rework.
7. ***Testability:*** The system should be designed to facilitate effective testing and maintenance. This includes incorporating proper unit testing, integration testing, and system testing practices. Test cases and test data should be well-documented, and the system should provide tools or utilities to automate testing processes. Easy testability enables efficient identification and resolution of software defects or issues.

By addressing these maintenance requirements, the system can be built with a focus on long-term maintainability and supportability. The modular design, comprehensive documentation, code maintainability, error handling mechanisms, compatibility, scalability, and testability contribute to reducing maintenance efforts, enhancing system stability, and ensuring smooth ongoing operations.

Operations Requirements:

1. ***Periods of Interactive Operations and Unattended Operations:*** The system should support both periods of interactive operations, where users directly interact with the system, and periods of unattended operations, where the system performs automated tasks without user intervention. The system should be designed to handle these operational modes seamlessly, ensuring smooth transitions and appropriate behavior during each phase.
2. ***Data Processing Support Functions:*** The system should provide necessary data processing support functions to enable efficient and effective operations. This includes data input, validation, processing, storage, retrieval, and reporting capabilities. The system should handle data processing tasks accurately and efficiently to meet user requirements.
3. ***Backup and Recovery Operations:*** The system should have robust backup and recovery mechanisms in place to protect data and ensure system availability. Regular backups of critical data should be performed, and backup procedures should be documented and followed. In the event of system failures or data loss, the system should have procedures and tools to facilitate timely recovery and

restoration of data and services.

4. ***Safety Considerations and Requirements:*** The system should adhere to safety considerations and requirements to protect users, operators, and the environment. This includes compliance with applicable safety standards, incorporating safety features into the system design, and providing clear instructions for safe system operation. Safety measures should be implemented to mitigate risks and prevent accidents or hazards associated with system usage.
5. ***Disaster Recovery and Business Resumption:*** The system should have a comprehensive disaster recovery plan in place to ensure business continuity in the event of major disruptions or disasters. This includes strategies for data backup, off-site storage, redundant systems, and failover mechanisms. The plan should outline procedures for quickly resuming operations and minimizing downtime in critical situations.

By addressing these operations requirements, the system can effectively support normal and special operations, ensuring reliable and uninterrupted functionality. The system's ability to handle interactive and unattended operations, provide data processing support functions, implement backup and recovery operations, adhere to safety requirements, and enable disaster recovery and business resumption contributes to the overall operational effectiveness and user satisfaction.

3.2.5 Security

Protection Requirements:

1. ***Encryption:*** The system should employ encryption techniques to protect sensitive data during transmission and storage. Encryption should be used to secure communication channels between the ATM machine and external systems, such as banking networks or online services. Additionally, sensitive data stored within the system, including account information and transaction records, should be encrypted to prevent unauthorized access.
2. ***Activity Logging and Historical Data Sets:*** The system should maintain detailed activity logs that capture user interactions, system events, and transaction records. These logs should include information such as user identification, date and time stamps, and the nature of the activity. Historical data sets should be securely stored to provide a record of system activities for auditing, investigation, and forensic purposes.

3. ***Restrictions on Intermodule Communications:*** The system should enforce strict restrictions on intermodule communications to prevent unauthorized access or tampering. Only authorized modules or components should be allowed to communicate with each other, and appropriate access control mechanisms should be in place to verify the authenticity and authorization of communication channels.
4. ***Data Integrity Checks:*** The system should implement data integrity checks to ensure the accuracy and integrity of stored data. This can include checksums, hash functions, or other cryptographic techniques to verify the integrity of data during transmission and storage. Data integrity checks should be performed at critical points, such as during data input, processing, and retrieval, to detect and prevent any unauthorized modifications or tampering.
5. ***Access Control and User Authentication:*** The system should implement robust access control mechanisms and user authentication protocols to verify the identity and authorization of users. This can include the use of strong passwords, two-factor authentication, biometric authentication, or other secure authentication methods. Access to sensitive functionality, such as account management or system administration, should be restricted to authorized individuals only.
6. ***Security Auditing and Monitoring:*** The system should have mechanisms in place to perform security auditing and monitoring activities. This includes regular audits of system configurations, access logs, and security settings to identify any vulnerabilities or unauthorized activities. Real-time monitoring should be implemented to detect and alert on any suspicious or abnormal behavior that may indicate a security breach or unauthorized access.

Authorization and Authentication Factors:

1. ***User Identification:*** The system should require users to provide a unique identifier, such as an account number, username, or card number, to authenticate their identity and access the system. This ensures that only authorized users can perform transactions and access account information.
2. ***Password-Based Authentication:*** The system should support password-based authentication, where users are required to provide a password or PIN (Personal Identification Number) to verify their identity. The passwords should be securely stored and encrypted to protect against unauthorized

access.

3. ***Two-Factor Authentication:*** To enhance security, the system may support two-factor authentication, which requires users to provide additional authentication factors in addition to a password. This can include the use of one-time passwords (OTP) generated through mobile apps or hardware tokens, biometric authentication (fingerprint or facial recognition), or other secure authentication methods.
4. ***Account Lockout:*** To prevent unauthorized access through brute-force or repeated login attempts, the system should implement an account lockout mechanism. After a certain number of failed login attempts, the user's account should be temporarily locked or suspended to protect against unauthorized access.
5. ***Account Privileges and Role-Based Access Control:*** The system should implement a role-based access control mechanism to assign different privileges and access rights to users based on their roles and responsibilities. This ensures that users can only perform actions and access information that is appropriate for their assigned roles.
6. ***Session Management:*** The system should manage user sessions to ensure that users are automatically logged out after a period of inactivity. This helps prevent unauthorized access to the system if a user leaves their session unattended.
7. ***Secure Communication Channels:*** The system should use secure communication protocols, such as HTTPS, to encrypt data transmission between the ATM machine and external systems. This ensures that sensitive user information, such as account credentials, is protected during transmission.
8. ***Audit Trail:*** The system should maintain an audit trail that records user activities, including login attempts, transactions, and system configuration changes. This allows for traceability and accountability in case of any security incidents or unauthorized access.

By incorporating these authorization and authentication factors, the system ensures that only authenticated and authorized users can access the ATM machine, perform transactions, and access sensitive account information. User identification, password-based authentication, two-factor authentication, account lockout, role-based access control, session management, secure communication channels, and audit trails contribute to the overall security and integrity of the system.

3.2.6 Standards Compliance

Standards Compliance Requirements

1. ***Data Security and Privacy:*** The ATM machine should comply with relevant data security and privacy standards, such as the General Data Protection Regulation (GDPR) or local data protection laws. This includes protecting sensitive user information, implementing secure data transmission protocols, and ensuring appropriate access controls to prevent unauthorized access to personal data.
2. ***Financial Regulations:*** The ATM machine should adhere to financial regulations and standards, such as those set by the financial regulatory authorities in the respective jurisdiction. This may include requirements for transaction recording, reporting, and compliance with anti-money laundering (AML) and Know Your Customer (KYC) regulations.
3. ***Accessibility Standards:*** The user interface of the ATM machine should meet accessibility standards to ensure that individuals with disabilities can effectively use the system. This may include compliance with accessibility guidelines such as the Web Content Accessibility Guidelines (WCAG) for visually impaired users or support for alternative input methods for users with motor disabilities.
4. ***Network and Communication Standards:*** The ATM machine should comply with relevant network and communication standards, such as the Transmission Control Protocol/Internet Protocol (TCP/IP) for network connectivity, encryption protocols for secure communication, and network infrastructure standards.
5. ***Compliance Reporting:*** The ATM machine should be capable of generating compliance reports as required by applicable standards, policies, regulations, or laws. This may include generating audit logs, transaction reports, or regulatory compliance reports for auditing purposes.
6. ***Software Development Standards:*** The software development process for the ATM machine should follow established software engineering standards and best practices. This may include adherence to coding conventions, documentation standards, version control practices, and software testing methodologies.
7. ***Hardware and Interoperability Standards:*** The ATM machine should comply with relevant hardware standards to ensure compatibility and interoperability with other systems and devices. This may include standards for hardware interfaces, card readers, cash dispensers, and other peripheral devices.

8. ***Environmental Regulations:*** The ATM machine should comply with environmental regulations regarding power consumption, waste management, and disposal of electronic equipment.

By incorporating these standards compliance requirements, the ATM machine ensures that it meets the necessary regulatory, legal, and industry standards. Compliance with data security and privacy regulations, financial regulations, accessibility standards, network and communication standards, compliance reporting, software development standards, hardware and interoperability standards, and environmental regulations helps to ensure a reliable, secure, and compliant operation of the ATM machine.

3.2.7 Other Non-Functional Requirements

Other Non-Functional Requirements

1. ***Performance:*** The ATM machine should have a fast and responsive user interface, ensuring that transactions are processed within an acceptable time frame. For example, cash withdrawal transactions should be completed within a specified time limit (e.g., 30 seconds) to provide a smooth user experience.
2. ***Reliability:*** The ATM machine should be highly reliable and available to users. It should minimize system failures and ensure that the system can recover quickly from any unexpected errors or disruptions. This includes regular system maintenance and backup procedures to prevent data loss and ensure uninterrupted service.
3. ***Scalability:*** The ATM machine should be scalable to accommodate an increasing number of users and transaction volumes. As the number of users or transactions grows, the system should be able to handle the increased load without significant performance degradation.
4. ***Availability:*** The ATM machine should be available to users for a significant portion of the time. It should have a high uptime and minimize downtime for maintenance or system upgrades. This may involve redundant hardware configurations, fault-tolerant design, and backup power systems to ensure continuous operation.
5. ***Maintainability:*** The ATM machine should be designed for ease of maintenance and updates. It should have modular and well-documented code, making it easier for developers to understand, modify, and fix issues. The system should also support remote maintenance and software updates to

minimize disruptions to the service.

6. ***Compatibility:*** The ATM machine should be compatible with various hardware and software platforms. It should support different types of ATM hardware, operating systems, and network infrastructures to ensure interoperability with existing systems and future upgrades.
7. ***Internationalization and Localization:*** The ATM machine should support multiple languages, currencies, and local regulations to accommodate users from different regions. This includes providing localized user interfaces, currency conversion capabilities, and adherence to local banking regulations.
8. ***Usability:*** The ATM machine should be intuitive and easy to use for users of varying technical backgrounds. The user interface should be clear, logically organized, and provide helpful prompts and error messages. The system should also consider accessibility requirements, such as support for assistive technologies and accommodating users with disabilities.
9. ***Security:*** The ATM machine should have robust security measures to protect user information and prevent unauthorized access or fraudulent activities. This includes encryption of sensitive data, secure authentication mechanisms, monitoring of suspicious activities, and compliance with industry security standards.
10. ***Documentation:*** The ATM machine should have comprehensive documentation that includes user manuals, system administration guides, troubleshooting guides, and any other relevant documentation to assist users and system administrators in understanding and operating the system effectively.

These non-functional requirements ensure that the ATM machine not only meets the functional needs of the users but also provides a reliable, performant, and secure system that is easy to maintain, compatible with existing infrastructure, and accessible to a wide range of users.

3.3 Domain Requirements

Domain requirements refer to the specific needs and considerations related to the industry or field in which the project operates. These requirements may include regulations, industry standards, best practices, or specific domain knowledge that must be considered during the development of the project. Here are some examples of domain requirements for an ATM machine project:

1. Banking Regulations: The ATM machine must comply with banking regulations and guidelines set by regulatory authorities, such as transaction limits, security measures, and data privacy requirements.
2. Payment Network Integration: The ATM machine should be compatible with the payment network(s) used by the banking institution, allowing for seamless transaction processing and communication with the banking system.
3. Currency Handling: The ATM machine should support the specific currency or currencies used in the target market, including accurate denomination recognition, counting, and dispensing of notes or coins.
4. Accessibility Standards: The ATM machine should comply with accessibility standards to ensure it is usable by individuals with disabilities, including features such as braille keypads, audio instructions, and suitable user interface design.
5. Security Standards: The ATM machine should adhere to industry security standards and best practices to prevent fraud, unauthorized access, and data breaches. This may include encryption of sensitive data, secure PIN entry, physical security measures, and secure communication protocols.
6. Banking Domain Knowledge: The development team should possess a good understanding of banking processes, terminology, and common practices to ensure the functionality and user experience of the ATM machine align with industry norms.
7. Network Connectivity: The ATM machine should be designed to operate reliably in various network environments, considering factors such as intermittent connectivity, network latency, and error handling during communication with the banking system.
8. Transaction Processing: The ATM machine should support real-time transaction processing, ensuring accurate and timely updates to account balances and transaction records.
9. Account Integration: The ATM machine should integrate with the banking institution's account management systems, allowing users to access their accounts, view balances, and perform

transactions seamlessly.

10. ATM Network Interoperability: If the ATM machine is part of a larger network of ATMs, it should be designed to interoperate with other machines in the network, enabling features such as card sharing, shared transaction logs, and centralized monitoring and management.

These domain requirements help ensure that the ATM machine aligns with the specific needs and considerations of the banking industry and provides a secure and reliable banking experience for users.

4. Design thinking methodologies

Provide all the Used Design Muscles in Software product

1. Empathy: Understanding the needs and experiences of the users through user research, interviews, observation, and immersion.
2. Define: Analyzing the gathered information to define the core problem or challenge and formulating a clear problem statement or design challenge.
3. Ideate: Generating a wide range of ideas and solutions to address the defined problem through brainstorming and encouraging divergent thinking.
4. Test: Iteratively testing and gathering feedback on prototypes from users to validate ideas, uncover usability issues, and refine the design based on user insights.
5. GUI (Graphical User Interface): Designing the visual interface of the software, including the layout, screens, icons, and other visual elements to create an intuitive and user-friendly experience.

These design muscles work together throughout the software development process to ensure that the product meets the needs of the users and provides a positive user experience.

5. Software Design

5.1 Use Case

Manage ATM Transactions

Business Scenario:

1. *Business Need:*
   - The bank wants to provide its customers with a convenient and secure way to perform

ATM transactions, including cash withdrawals, balance inquiries, and deposits.

- The bank wants to maintain accurate transaction history for each account.

2. *Problem:*
    - Customers need to access their accounts securely and perform various transactions efficiently.
    - The bank needs to track and record all transactions accurately.
3. *Business and Technical Environment:*
    - The bank operates multiple ATMs connected to a centralized banking system.
    - Customers have unique IDs and PINs to access their accounts.
    - The ATMs have a user interface for customers to interact with the system.
    - The system manages customer accounts, transaction history, and transaction limits.
4. *Objectives:*
    - Provide customers with a user-friendly interface to perform ATM transactions.
    - Ensure secure authentication and authorization of customers.
    - Enable customers to withdraw cash within the specified limits.
    - Allow customers to check their account balance.
    - Facilitate cash and check deposits within the specified limits.
    - Maintain accurate transaction history for each customer.
5. *Actors:*
    - Customer: The bank's customer who interacts with the ATM system to perform transactions.
    - Bank System: The centralized system that manages customer accounts, transaction history, and limits.

Use Cases:

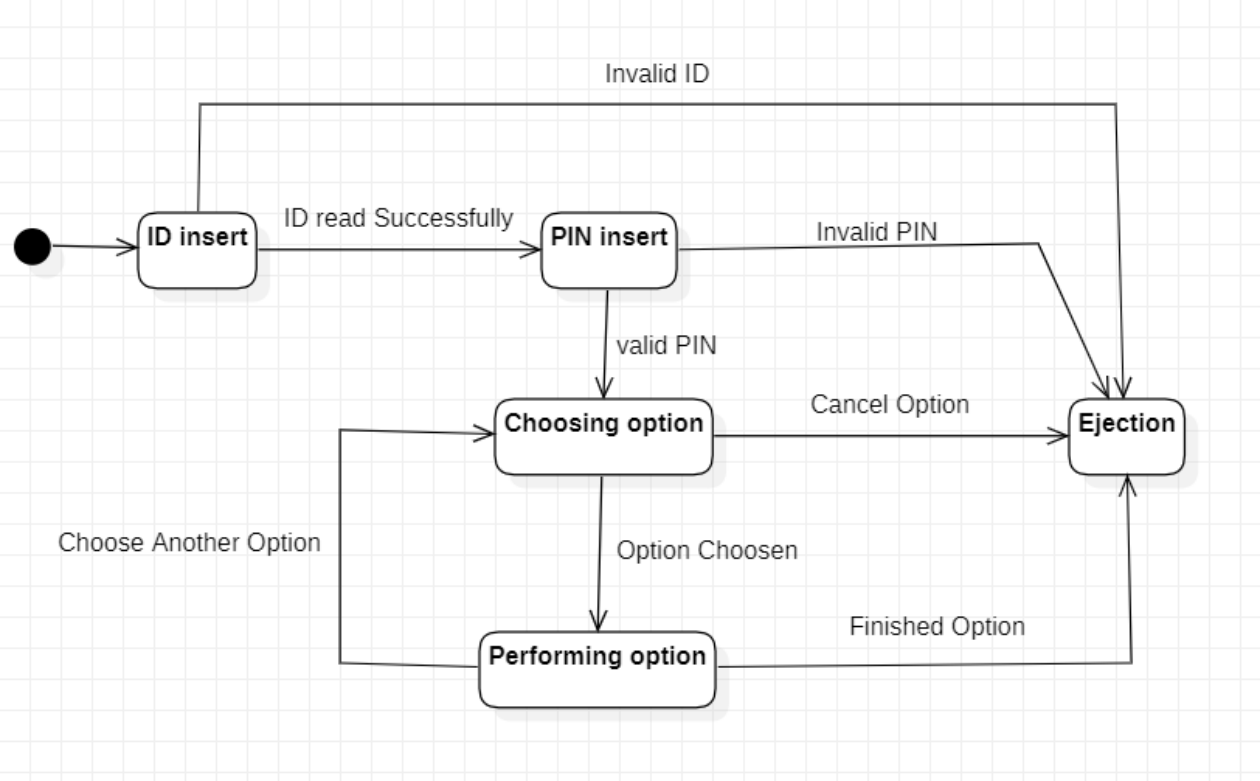
6. *Withdraw Cash:*
    - Description: The customer wants to withdraw a specific amount of cash from their account.
    - Actors: Customer, Bank System
    - Precondition: Customer is authenticated and has sufficient funds in their account.
    - Postcondition: Customer's account balance is updated, and cash is dispensed.

- Success Metrics: Customer receives the requested cash, and the account balance is updated accurately.

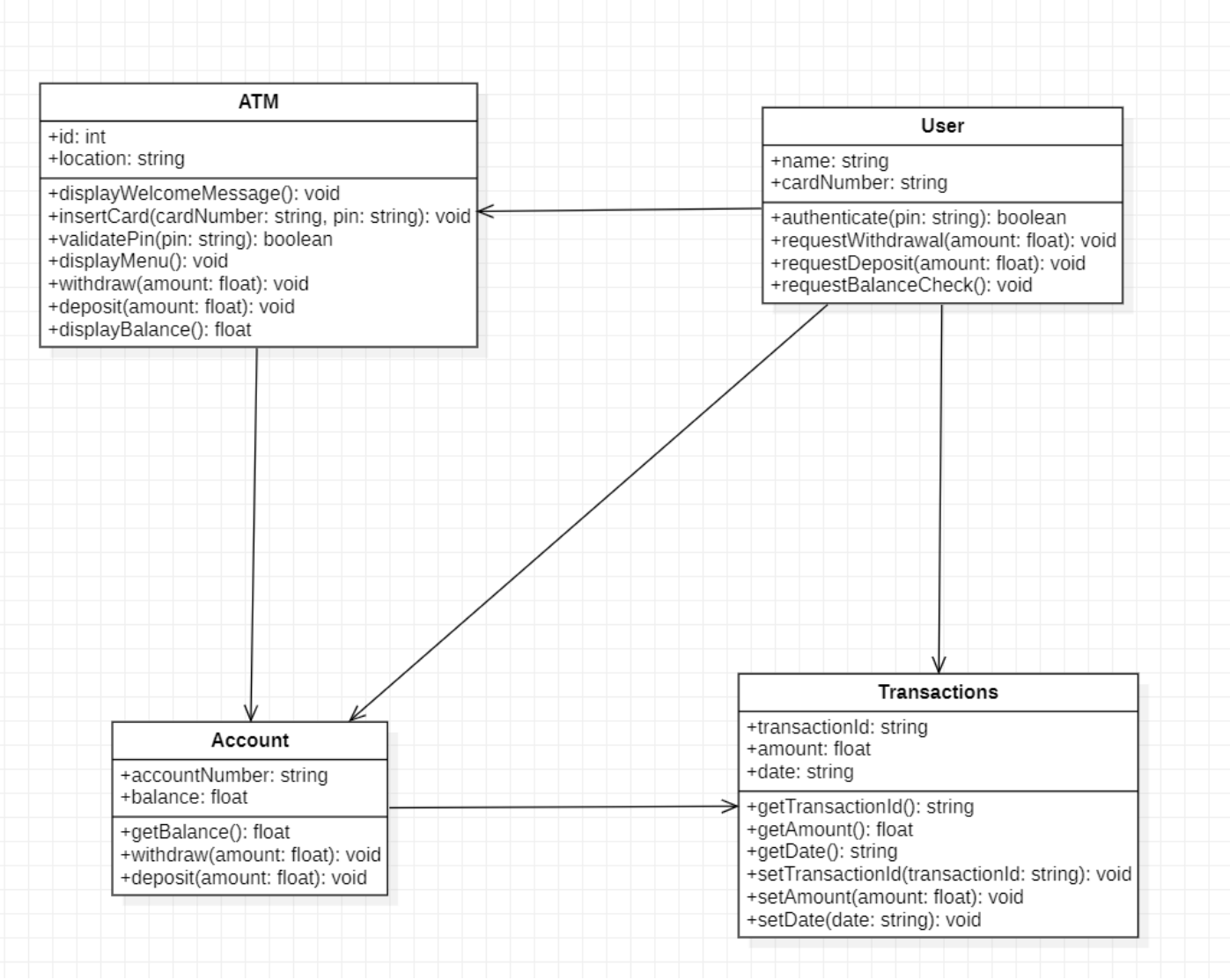
7. *Check Account Balance:*
    - Description: The customer wants to check the current balance of their account.
    - Actors: Customer, Bank System
    - Precondition: Customer is authenticated.
    - Postcondition: Customer receives the account balance information.
    - Success Metrics: Customer receives accurate account balance information.
8. *Deposit Cash:*
    - Description: The customer wants to deposit a specific amount of cash into their account.
    - Actors: Customer, Bank System
    - Precondition: Customer is authenticated and the cash deposit amount is within the specified limit.
    - Postcondition: Customer's account balance is updated.
    - Success Metrics: Customer receives confirmation of the successful deposit, and the account balance is updated accurately.
9. *Deposit Check:*
    - Description: The customer wants to deposit a check into their account.
    - Actors: Customer, Bank System
    - Precondition: Customer is authenticated and the check deposit amount is within the specified limit.
    - Postcondition: Customer's account balance is updated.
    - Success Metrics: Customer receives confirmation of the successful deposit, and the account balance is updated accurately.
10. *Display Transaction History:*
    - Description: The customer wants to view their transaction history.
    - Actors: Customer, Bank System
    - Precondition: Customer is authenticated.
    - Postcondition: Customer receives the transaction history information.
    - Success Metrics: Customer receives accurate transaction history information.

11. *Update Accounts File:*
    - Description: The system needs to update the accounts file with the latest account and transaction information.
    - Actors: Bank System
    - Precondition: None
    - Postcondition: The accounts file is updated with the latest data.
    - Success Metrics: The accounts file reflects the accurate account and transaction information.

5.2 State Diagram



5.3 Class Diagram



APPENDIX

Appendix A. Definitions, Acronyms, and Abbreviations

Appendix B. References

Appendix C. Organizing the RequirementS