ATM Cash Withdrawal Management System

Name: M nazim

Class : Bscs 4th semester

Roll no: 21305/411705

Course code: swe-401

Submitted by: M nazim

Submitted to: Ma’am ANZA

Software Requirements Specification Documents

Category: Web application

Purpose:

The purpose of this document is to outline the requirements for thr development of an ATM
Cash Withdrawal Management System. This System aims to provide users with secure and convenient cash withdrawal services through automated teller machines

Scope:

The ATM Cash Withdrawal Management System will facilitate user authentication cash withdrawal balance inquiry transaction history retrieval user profile management and ATM management functionalities

Introduction:

The ATM Cash Withdrawal Management System is a software engineering model made to make cash withdrawals from automated teller machines (ATM) more secure and efficient. The banking industry relies heavily on this system because it ensures customers have easy access to cash and preserves the integrity of financial transactions.
ATM have become an essential part of our day-to-day lives in today's fast-paced world, providing us with 24/7 access to our funds. Banks must manage and optimize ATM cash withdrawals in order to meet customer expectations and keep their edge over competitors. In this manner, the improvement of a solid and

vigorous programming framework for ATM cash withdrawal the executives is fundamental.

Key Features and Applications:

Authentication of Users:

Through a combination of PINs (Personal Identification Numbers), biometric verification (fingerprint or iris scanning), and card validation, the system ensures secure user authentication.

2. Exchange Observing:

Every cash withdrawal is tracked and recorded by the system, including the date, time, and amount. It keeps a review trail to work with following and examination of any dubious exercises or disparities.

3. Management of Cash Availability:

Each ATM's cash level is continuously monitored by the system to ensure sufficient availability to satisfy customer demands. When cash levels fall below predetermined thresholds, it sends real-time alerts to bank administrators, allowing for proactive replenishment.

4. Limits and restrictions on transactions:

Banks can use the system to limit transactions and impose restrictions based on user profiles, such as daily or weekly limits, maximum withdrawal amounts, or currency preferences.

5. Notification and monitoring of faults: In order to identify ATM hardware or software failures, the system makes use of a variety of sensors and monitoring mechanisms. It notifies designated support teams immediately, allowing them to address any issues promptly and reduce downtime.

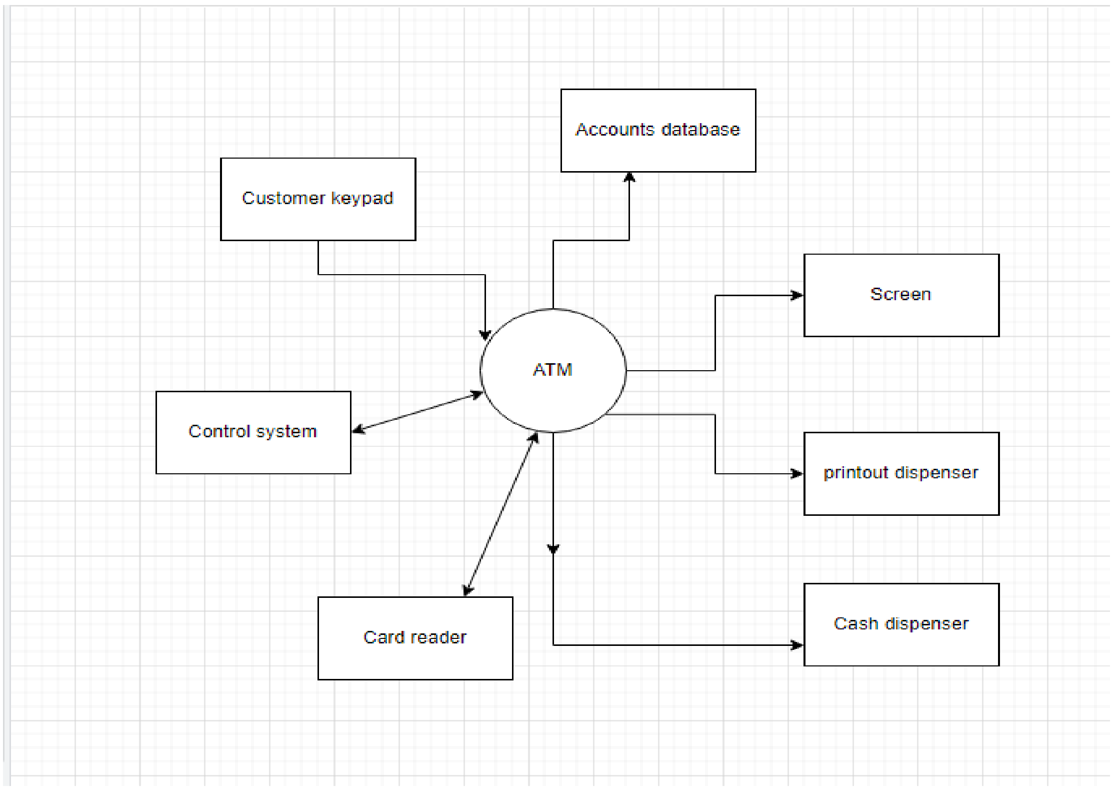
6. Prevention and detection of fraud:

The framework utilizes progressed calculations and AI methods to recognize and forestall deceitful exercises, for example, skimming, card cloning, or unapproved access endeavors. It constantly looks at the patterns of transactions, finding oddities and triggering the right security measures.

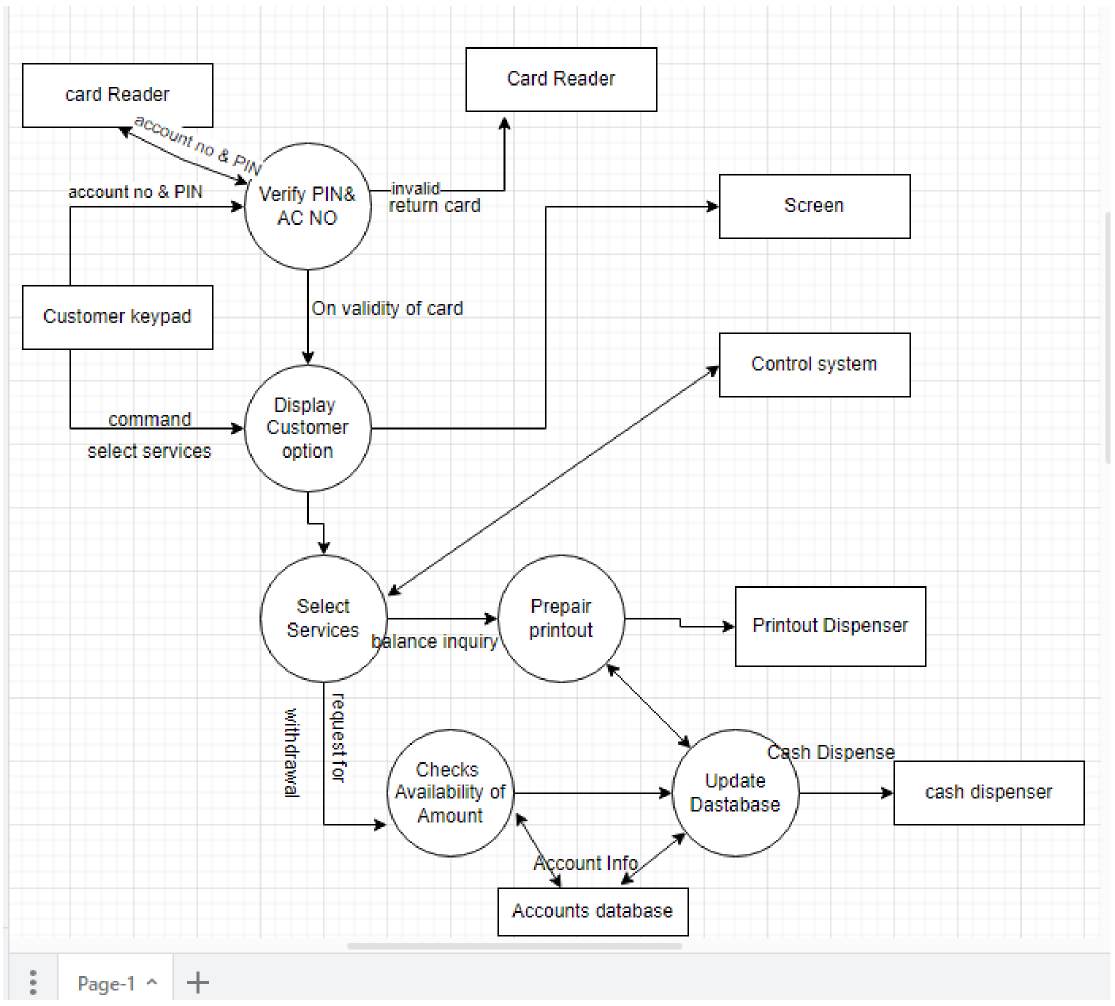
7. System Integration with Banking:

Customer account balances, transaction records, and authentication procedures can all be updated in real time thanks to the system's seamless integration with the bank's core banking software.

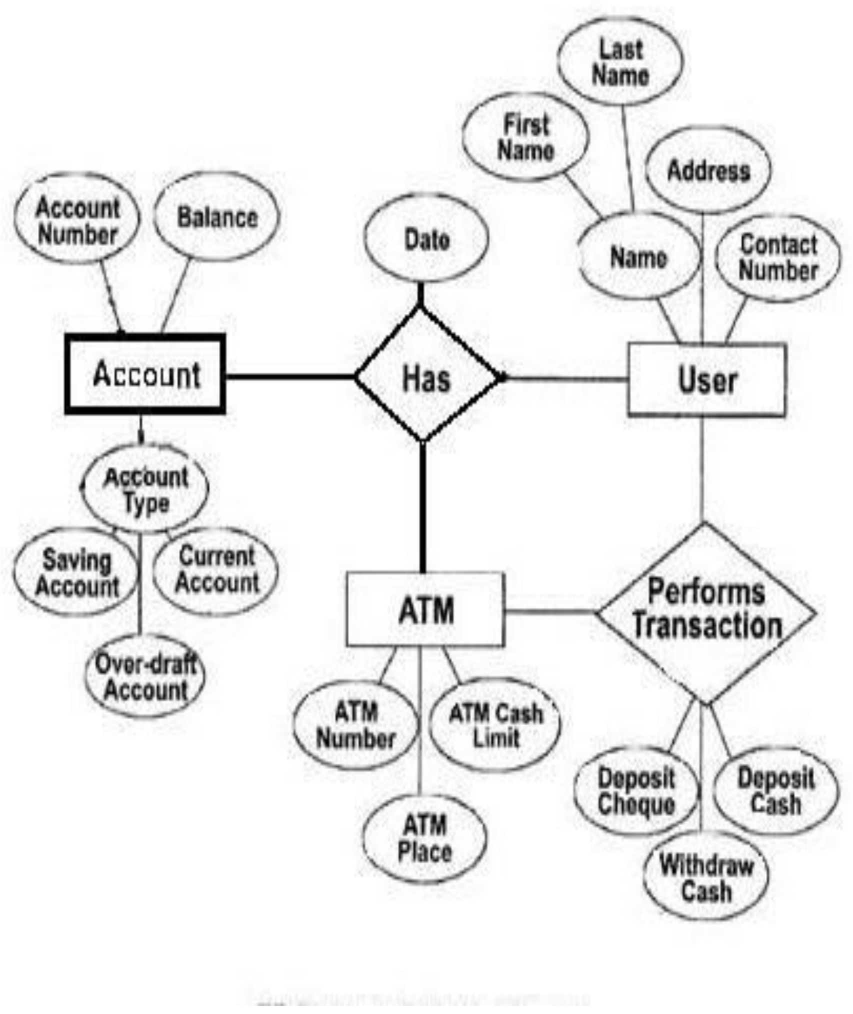
Data flow diagram for ATM management system (level 0)



Data flow diagram for ATM management system (level 1)



Entity relationship diagram for ATM Cash Withdrawal Management system



Descripion:

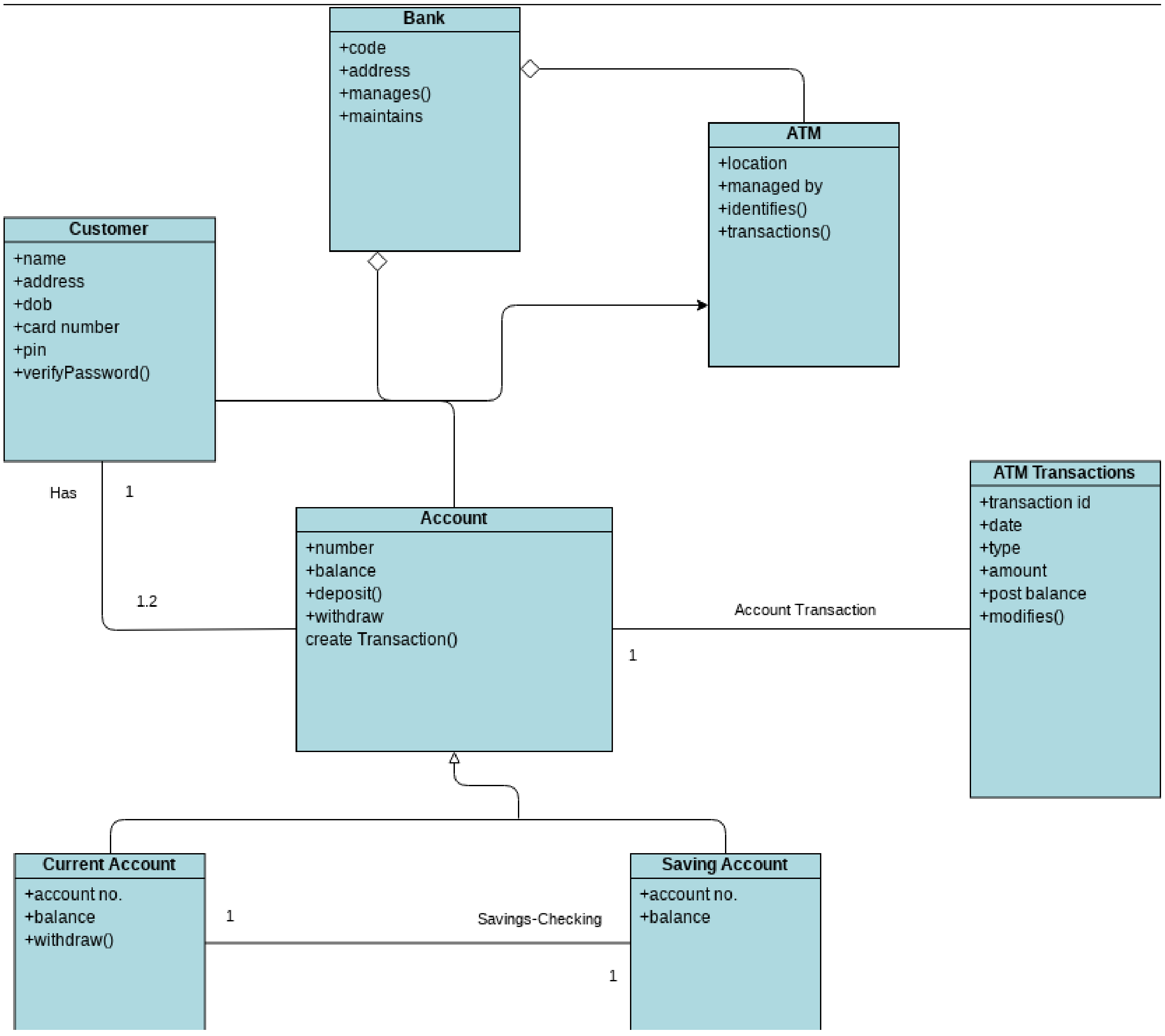
The General Things needed in a ATM system are: –

1. Person Opens an Account

2. Person using ATM for Transaction

The person opens an Account in a Bank and gets a account number and ATM card. The person can make transactions in ATM centres. The Details of the Transaction has to be maintained Between Three Entitys. i.e. User, Account, ATM

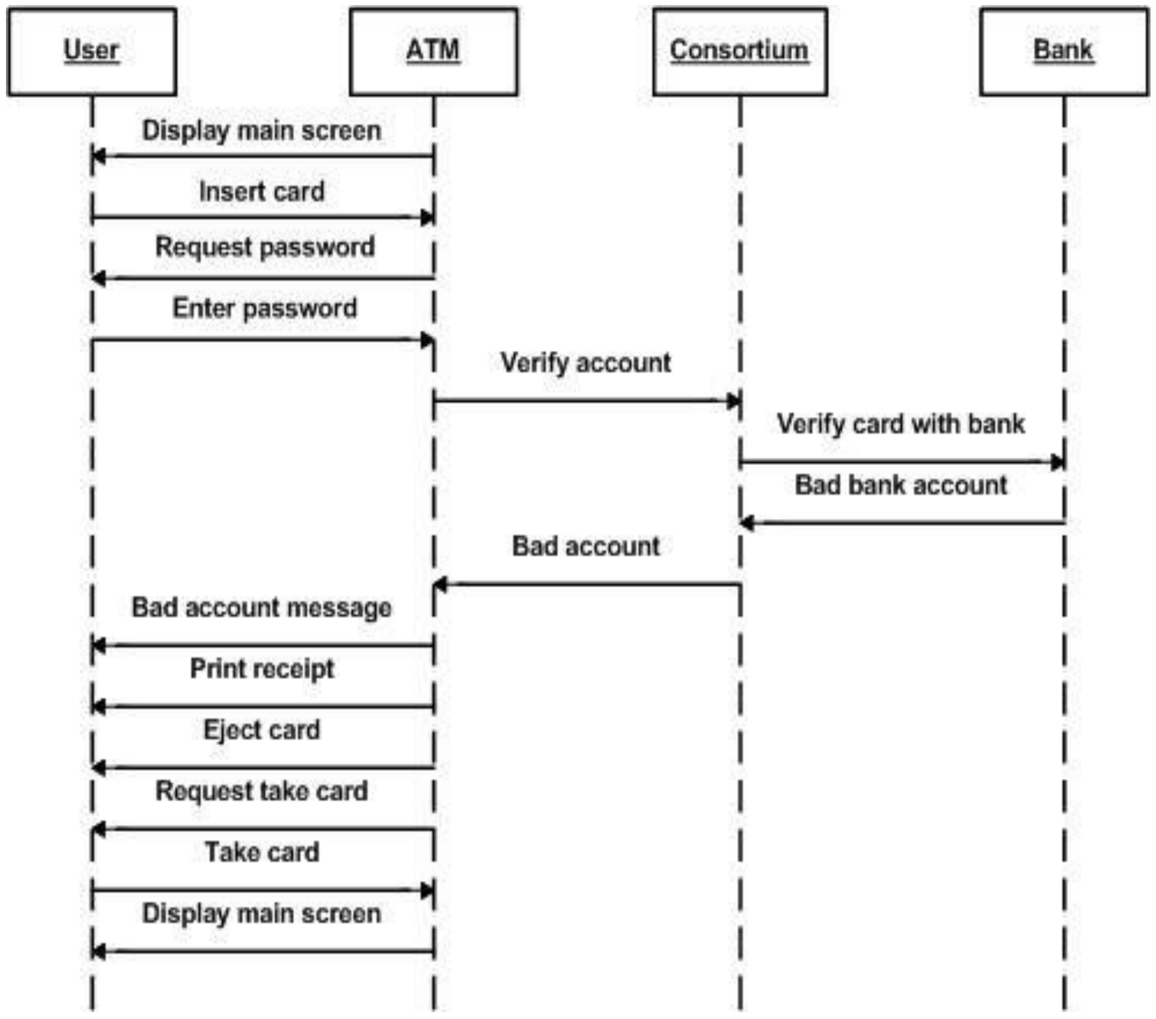
Class diagram:



Description:

The ATM Cash Withdrawal Management System class diagram illustrates the essential classes and their relationships. It includes classes for the ATM Machine, Bank Account, Transactions, Card, Bank, Branch, and Administrator. These classes encapsulate attributes and methods for key functionalities such as dispensing cash, managing accounts, logging transactions, and system administration. This diagram serves as a blueprint for building the system.

Sequence diagram :

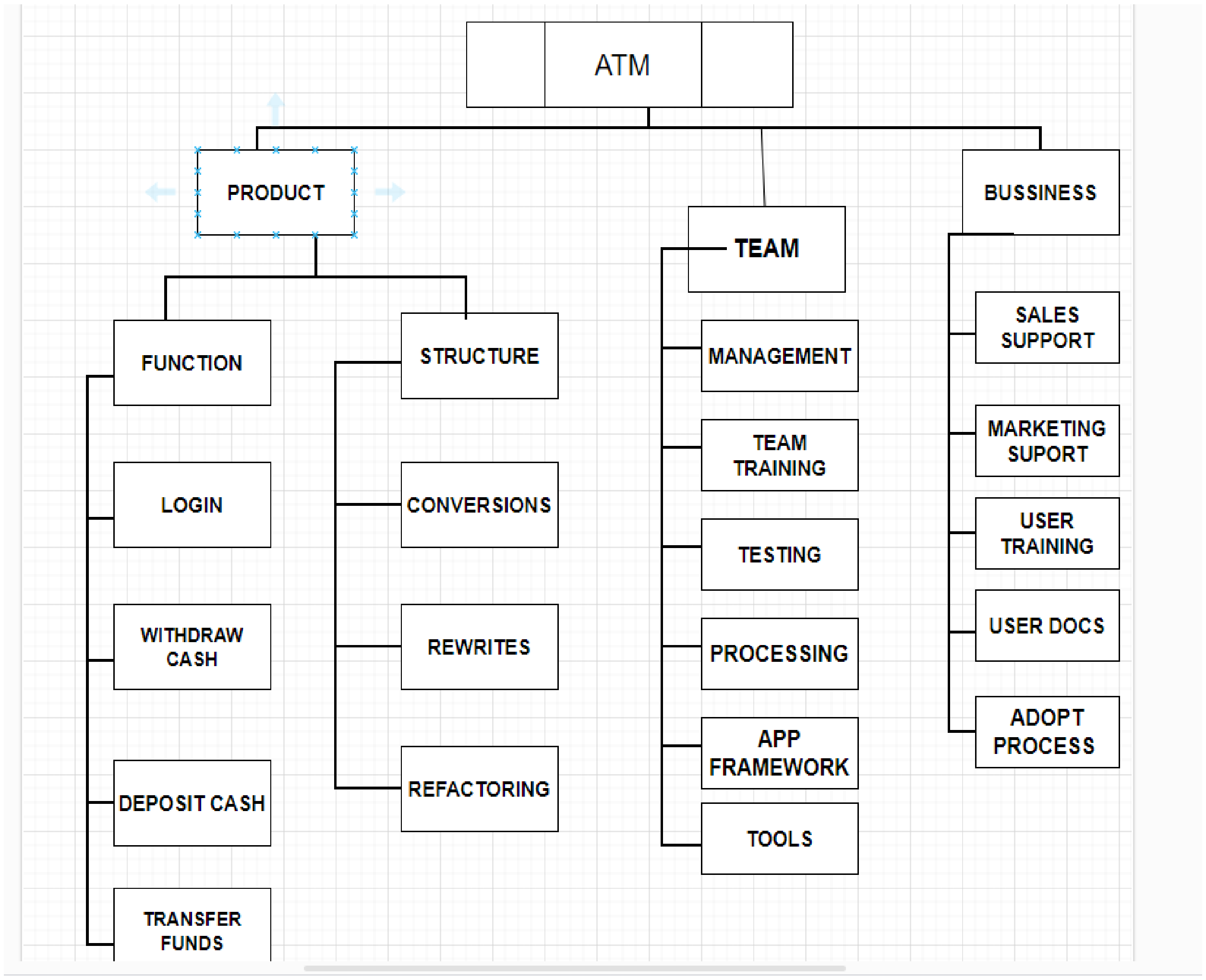


Description:

Figure shows an example of a UML sequence dia-gram, which illustrates a cash withdraw scenario in an auto-matic teller machine (ATM). The diagram shows a possible interaction between instances of classes Cash Dispenser,Card Reader, AT M and Session, when a user decides to make a withdraw and the amount of cash is dispensed successfully.

A message defines a particular communication between life lines of an interaction. It can be either asynchronous (represented by an open arrow head) or synchronous (rep-resented by a filled arrow head). Additionally, there are two special kinds of messages, lost and found, with the obvious meaning, which are described by a small black circle At the arrow head or origin respectively.

Work Breakdown Structure Diagram:



Description:

Certainly! Here's a simplified Work Breakdown Structure (WBS) for the ATM Cash Withdrawal Management System:

1. **Product**

- Define Scope
- Assemble Team

- Stakeholder Meetings

2. Requirements

- Gather Functional and Non-Functional Requirements
- Prioritize and Document

3. Structure

- Class Diagram
- Database Schema
- UI Layout

4. Management

- ATM Functions
- Bank Account Management
- Transaction Logging
- Card Handling
- Bank and Branch Operations
- Administrator Features

5. Testing

- Unit and Integration Testing
- System Testing
- User Acceptance Testing

6. Bussiness

- System Documentation
- User Manuals
- Training Sessions

Functional Requirements for ATM Management System:

1. User Authentication: The system should provide secure and reliable user authentication methods, such as PIN verification or biometric identification, to ensure that only authorized individuals can access the ATM services.

2. Cash Withdrawal Transaction: The system should facilitate the process of cash withdrawal for customers, including validating account information, verifying available balance, dispensing cash, and updating the account balance after the transaction.

3. Deposit Transaction: The system should support deposit transactions, allowing customers to deposit cash or checks into their accounts through the ATM. It should provide proper validation and acknowledgment of the deposited amount.

4. Balance Inquiry: The system should enable customers to check their account balance through the ATM, providing accurate and up-to-date information.

5. Funds Transfer: The system should allow customers to transfer funds between their accounts or to other designated accounts within the bank. It should ensure secure and accurate transfer of funds.

6. Bill Payment: The system should support bill payment functionalities, allowing customers to pay bills such as utilities, credit cards, or loans through the ATM. It should validate payment details and ensure secure transactions.

7. Account Statement: The system should provide customers with the ability to request and receive an account statement from the ATM, displaying transaction history, account activity, and balances.
8. PIN Management: The system should allow customers to change their PINs securely through the ATM, providing a straightforward process for managing their account security.

Non-Functional Requirements for ATM Management System:

1. Usability: The system should have a user-friendly interface and intuitive navigation, ensuring that customers can easily understand and operate the ATM services.
2. Performance: The system should provide fast and responsive performance, ensuring minimal latency and quick transaction processing to deliver a seamless user experience.
3. Security: The system should incorporate robust security measures to protect customer data and transactions. It should include encryption techniques, secure communication protocols, and mechanisms to detect and prevent fraud.
4. Reliability: The system should be highly reliable and available at all times, minimizing downtime and disruptions in ATM services. It should have backup and recovery mechanisms to ensure continuous operation.
5. Scalability: The system should be salable to handle increasing user demands and accommodate a growing number of transactions without compromising performance.

6. Compatibility: The system should be compatible with various ATM hardware and software configurations, ensuring seamless integration and interoperability.
7. Maintainability: The system should be easily maintainable, allowing for efficient updates, bug fixes, and enhancements. It should have well-documented code and proper version control to facilitate future maintenance activities.
8. Compliance: The system should comply with relevant regulatory and compliance standards, such as PCI-DSS (Payment Card Industry Data Security Standard), ensuring the security and privacy of customer information.
9. Audibility: The system should maintain an audit trail of all transactions and activities performed through the ATM, allowing for auditing, monitoring, and compliance purposes.
10. Disaster Recovery: The system should have robust disaster recovery mechanisms in place to recover quickly in case of system failures, natural disasters, or other unforeseen events. It should include data backups, redundant systems, and contingency plans.
These functional and non-functional requirements form the foundation for designing and developing an effective and reliable ATM management system. They address both the core functionalities of the system as well as important considerations related to usability, performance, security, and compliance.

Use Case Model

A Use Case Model describes the proposed functionality of new system. A Use Case represents a discrete unit of interaction between a user (human or machine) and the system.

List of Actors

- Customer: This person perform transaction using card
- Bank: This person manages transaction records
- Technician: This person maintain the machinery

List of Use Cases:

Withdraw cash:

The user can withdraw money by adding PIN

Withdraw checking:

The user can also withdraw checking by adding PIN

Withdraw saving:

The user can also withdraw saving by adding PIN

Deposit cash/check:

The user can deposit cash/check after authentication

Transfer funds:

The user can transfer funds after authentication

Pay bill:

The user can pay bill after authentication

Print receipt:

The user can print statement after authentication

Authentication:

The user has to be authenticate by inserting card and valid PIN

Insufficient balance:

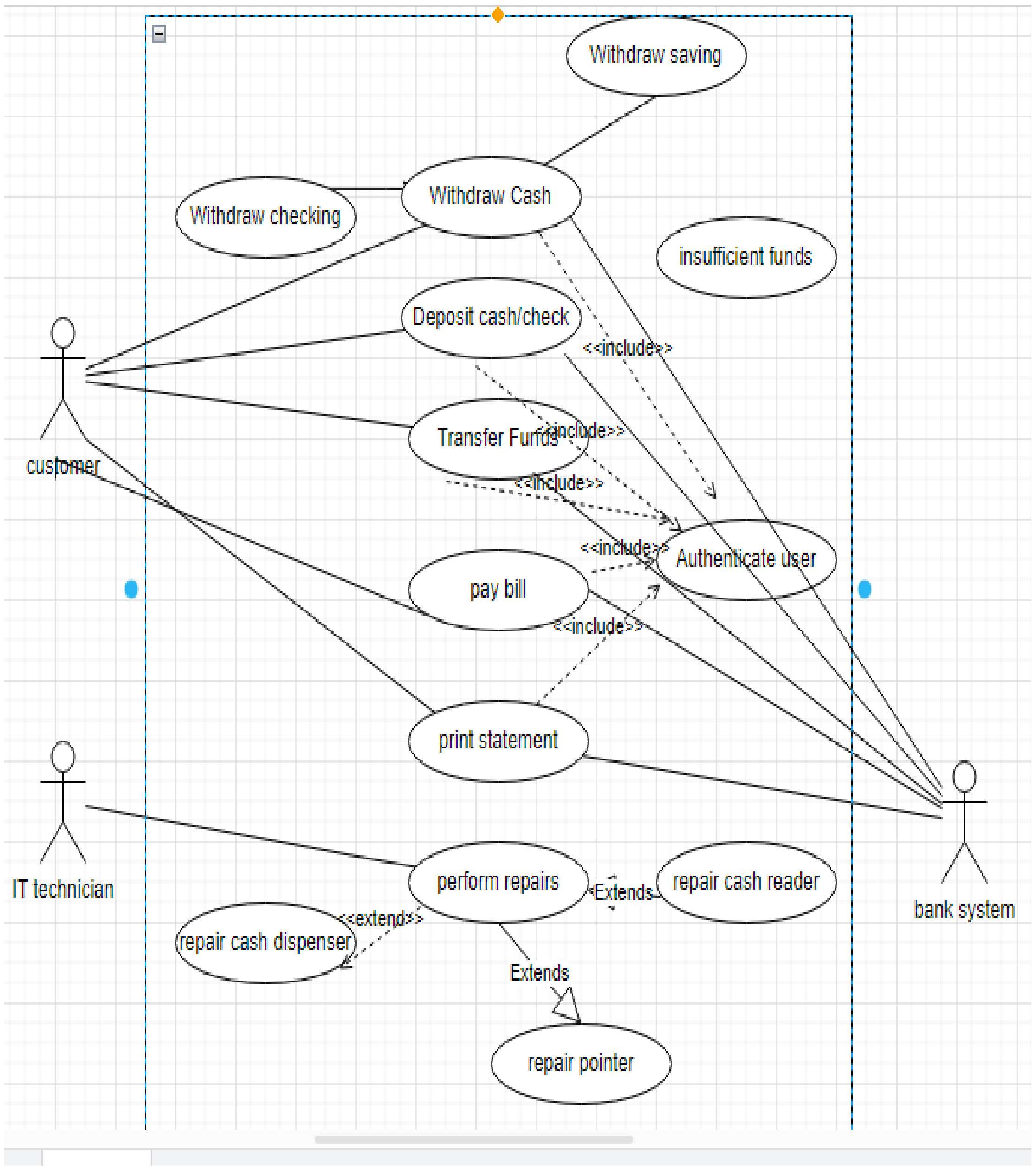
Sometimes times the balance could be insufficient

Cash dispense repair:

The technician can repair dispense cash

Card reader repair: The technician can repair the card reader

Use Case Diagram :



Adopted Methodology:

Agile Development (Scrum) Model

Overview:

Agile methodologies prioritize collaboration, flexibility, and continuous improvement. Scrum is a popular Agile framework that organizes work into time-boxed iterations called "sprints." Each sprint typically lasts 2-4 weeks, during which a cross-functional team works on a set of prioritized features. Scrum fosters adaptability, rapid development, and the delivery of incremental value to users.

Applicability to ATM Cash Withdrawal Management System:

1. Iterative Development:

The ATM system can be built in increments, allowing for early and continuous delivery of essential functionalities like user authentication, cash withdrawal, and balance inquiry. This allows stakeholders to see progress quickly and provide feedback.

2. User Feedback:

Agile methodologies encourage regular user involvement. Frequent demonstrations of working software during sprint reviews allow users to provide feedback and make adjustments early in the development process.

3. Flexibility:

The banking industry and technology landscape can evolve rapidly. Agile's flexibility allows the development team to respond to changes in regulations, security measures, or technology updates.

4. Requirement Refinement:

In an ATM system, requirements might evolve as stakeholders gain a deeper understanding of the system's needs. Agile supports ongoing requirements refinement and adjustment in response to changing business needs.

5. Cross-Functional Teams:

A Scrum team consists of members with diverse skills (developers, testers, UI/UX designers, etc.). This composition is ideal for a complex system like an ATM, where multiple aspects need attention.

6. Prioritization and Time Management:

Scrum's prioritization of features through the Product Backlog helps ensure that the most valuable functionalities are developed first. Timeboxing sprints also aids in managing development timelines.

7. Quality Assurance:

Regular testing and continuous integration practices in Agile ensure that the system is thoroughly tested and stable at the end of each sprint.

Implementation Steps:

1. Product Backlog Creation:

Collaboratively identify and prioritize system features (user stories) with stakeholders.

2. Sprint Planning:

Select a set of high-priority user stories for the upcoming sprint based on their complexity and value. Break them into smaller tasks.

3. Sprint Execution:

The development team works on implementing the selected user stories, focusing on completing the tasks within the sprint timeframe.

4. Daily Standups:

Short daily meetings where team members discuss progress, challenges, and plan their work for the day.

5. Sprint Review:

At the end of each sprint, present the completed user stories to stakeholders and gather feedback. Adjust the backlog based on this feedback.

6. Sprint Retrospective:

Reflect on the sprint process, discuss what went well and what could be improved, and make adjustments for the next sprint.

7. Repeat:

Iterate through sprint cycles, gradually building and refining the ATM Cash Withdrawal Management System.

Considerations:

Stakeholder Involvement:

Frequent communication with stakeholders is crucial to ensure that the system aligns with business goals and user needs.

Technical Debt:

While Agile promotes flexibility, managing technical debt (shortcuts taken to meet deadlines) is important to maintain system quality.

Change Management:

Agile accommodates changes, but proper change management processes should be in place to evaluate and incorporate modifications effectively.

Team Collaboration:

Effective collaboration and communication within the cross-functional team are essential for the success of Agile development

Software tools:

- Frontend: html, css, javascript
- Backend: php, mysql
- Databease server: Apache

Deployment:

Operating system server: Windows Linux and Unix

Hardware specifications:

- Processor:Intel core i5
- Ram :4GB
- Hard disk :1TB