

COMP 2663 – Software Engineering 1, Fall 2024

Student Name(s): Neer Patel, Mantaj Singh, Shivam Sangwan

Student ID(s): 0302826p, 0303830s, 0302959s

Contributions of each student:

Shivam-> Q1

Mantaj->Q2

Neer-> Q3

Date: Oct 10, 2024.

Project Component 2 – Requirements: SRS and Use Cases

You may complete this in a group of size up to 3.

1. Write the high-level requirements for your course project. Follow the form of IEEE830-1993 that is provided in Chapter 11. Track the amount of time spent doing this exercise. Decide what fraction of the requirements are understood by the team. Estimate how long it would take to obtain 95 percent of the requirements. State how the process you used could have been improved. Be specific and provide examples.



High-Level Requirements (System for Bank Administration)

1. Overview

The purpose of ATMs is to facilitate efficient and secure banking transactions.

Scope: The system permits deposits, withdrawals, transfers, and balance inquiries.

2. Product Features: Cash deposits and withdrawals.

Transfer funds across accounts.

In real time, display the account balance and transaction history.

Both non-technical users and bank customers who use ATMs are considered users.

3. Specific Requirements

Permit cash withdrawals and deposits at any ATM.
Permit transfers of funds and balance checks.
Transactions must be completed in under 20 seconds.
Support a maximum of 1000 users concurrently.

Security Requirements: Encrypt transactions using industry-standard techniques.

4. Evaluation and Tracking Exercise

Record the amount of time you spent creating these specifications.
Team Knowledge: Evaluate the team's comprehension of these demands.

A 95% estimation Understanding: Calculate how long it will take to get to a 95% comprehension level.

Enhance the process by using visual aids to explain difficult ideas during requirement talks.

2. Create high-level use cases and provide UML use case diagrams that correspond to the essential requirements of your project.

Answer 2:

A) Use Case: Withdraw Money

Primary Actor: Customer

Description: The user arrives at an ATM, inserts their bank card, and enters their PIN. They then request a money withdrawal. The system validates their account, checks the balance, and dispenses the cash.

Success Scenario:

1. Customer inserts card.
2. Customer enters PIN.
3. Customer selects withdrawal and enters the amount.
4. System verifies balance and dispenses cash.

B) Use Case: Deposit Money

Primary Actor: Customer

Description: The user can deposit money by inserting cash or checks into the ATM. The system processes the deposit and updates the user's balance.

Success Scenario:

1. Customer inserts card and enters PIN.
2. Customer selects deposit and inserts cash or checks.
3. The system validates the deposit and updates the balance.

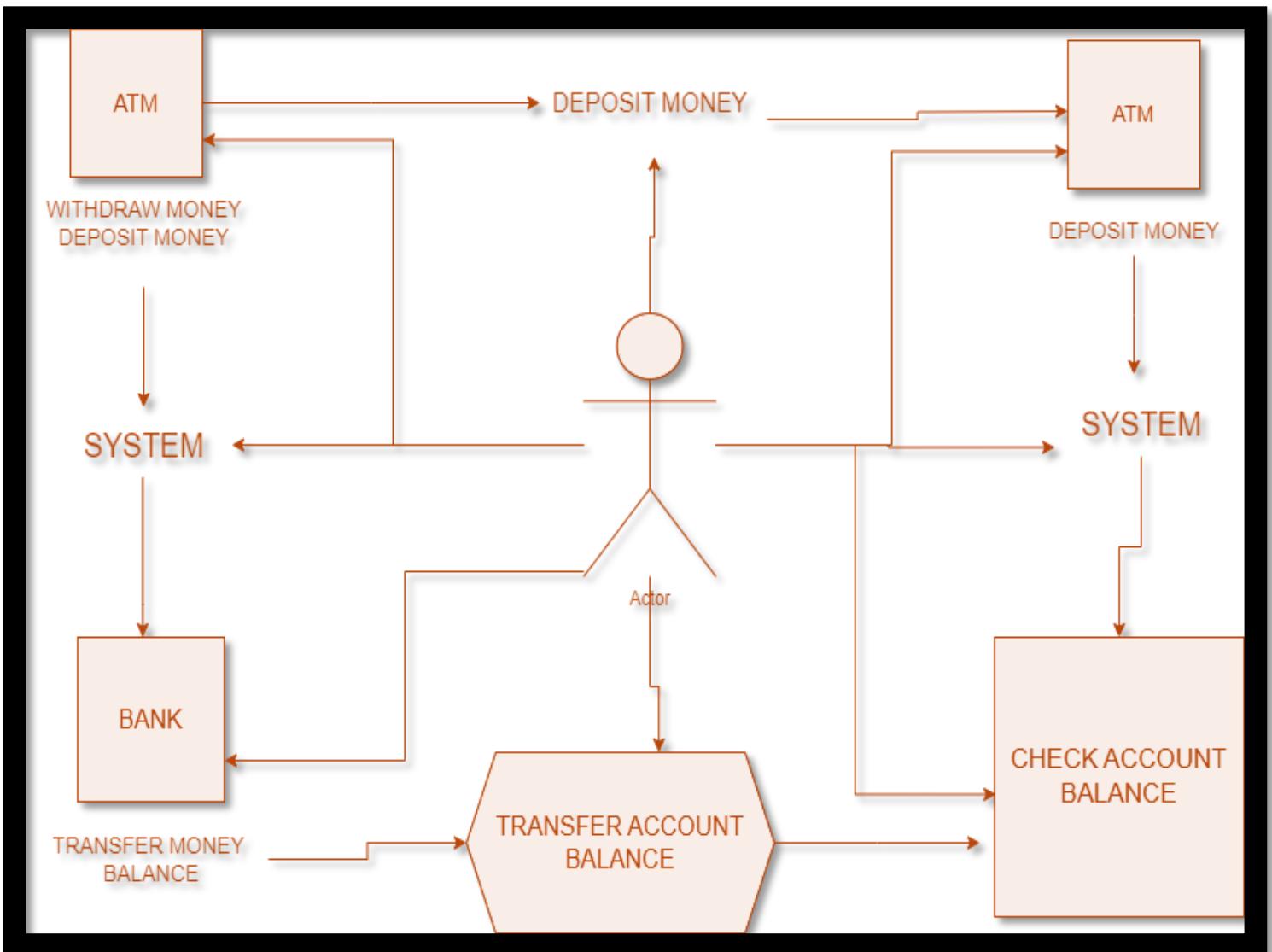
C) Use Case: Transfer Money

Primary Actor: Customer

Description: The user transfers money between accounts.

Success Scenario:

1. Customer inserts card and enters PIN.
2. Customer selects "Transfer Money."
3. Customer chooses source and destination accounts.
4. Customer enters the amount, and the system verifies the balance and processes the transfer.



3. Create a draft of the Gantt chart that will be used for your project.

Answer 3:

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
------	--------	--------	--------	--------	--------	--------	--------	--------

Project Planning	●	●			
Requirements Gathering	●	●			
System Design	●	●			
Use Case & Diagrams	●	●			
Database Design	●	●			
Front-End Design (ATM UI)	●	●			
Back-End Development		●	●		
Integration (Back-End & UI)		●	●		
Testing			●	●	
Bug Fixing & Optimization			●	●	
Final Testing & Deployment				●	●
Documentation & Report					●

1. Project Planning (Weeks 1-2):

- We start with planning in the first two weeks. This is where we figure out the project's goals, set timelines, and assign roles. It's all about getting everyone on the same page so we can move forward smoothly.

2. Requirements Gathering (Weeks 1-2):

- Alongside planning, we're also gathering requirements. This means talking to everyone involved, like the stakeholders, to make sure we understand what they need from the system. It's crucial for building something that actually works for the users.

3. System Design (Weeks 2-3):

- After we know what's needed, the system design phase kicks in. In Weeks 2 and 3, we'll be laying out the blueprint for the system, like how everything fits together—from the user interface to the backend. This is where the system starts to take shape, at least on paper.

4. Use Cases & Diagrams (Weeks 2-3):

- During these weeks, we're also working on use cases and diagrams. These are like visual guides that show how the system should function and how different parts will interact. Think of them as the roadmap for making sure the system behaves as expected.

5. Database Design (Weeks 3-4):

- In Weeks 3 and 4, we'll dive into the database design. We'll be setting up the structure that holds all the important data, making sure it's organized and efficient. The goal here is to build a database that can handle everything the system needs.

6. Front-End Design (ATM UI) (Weeks 4-5):

- Starting from Week 4 and continuing into Week 5, the focus shifts to designing the front-end, specifically the user interface (UI) for the ATM system. This involves creating a clean, user-friendly design that meets the requirements gathered earlier. The UI is an essential part of the system, as it ensures users have an intuitive experience when interacting with the ATM.

7. Back-End Development (Weeks 5-6):

- In Weeks 5 and 6, the team will work on developing the back-end of the system. This is where all the core logic and processing takes place, including handling transactions, communicating with the database, and managing user data. The back-end development ensures that the system operates correctly behind the scenes.

8. Integration (Back-End & UI) (Weeks 6-7):

- Once the front-end and back-end are built, the integration phase begins in Week 6 and continues into Week 7. During this time, the two parts of the system will be connected to ensure seamless communication between the user interface and the back-end

processes. This is a critical step to make sure that data flows smoothly and the system functions as intended.

9. Testing (Weeks 6-7):

- Testing also begins in Week 6 and extends into Week 7. The system will undergo various tests to identify and resolve any bugs or issues. This phase ensures that the system meets the expected quality standards and works without errors. Testing may include unit testing, integration testing, and user acceptance testing.

10. Bug Fixing & Optimization (Weeks 7-8):

- During Weeks 7 and 8, the team focuses on fixing any bugs found during testing and optimizing the system's performance. This phase ensures that the system runs efficiently and is free of any critical issues before final deployment.

11. Final Testing & Deployment (Week 8):

- In the final week, the system undergoes a last round of testing to ensure everything is functioning properly after the bug fixes. Once the system passes all tests, it is deployed, making it available for actual use by the client or end-users.

12. Documentation & Report (Week 8):

- The last task, which also happens in Week 8, is writing up the final documentation and project report. This includes all the details of the system design, development process, testing, and deployment. Proper documentation is important for future maintenance and ensuring that any new developers or users can understand the system.

Each of these tasks is marked on the Gantt chart with a filled circle (●), so we can easily see when we'll be working on each part of the project. Some tasks

overlap to make sure we're moving forward without unnecessary delays. All in all, this timeline helps keep the project on track and ensures that we're moving from one phase to the next as smoothly as possible.