Bank Management System

ITSE302

Group Members:

Shakeel Mohammad 202009190

Contents

1 Business Case:

1.1 Requirements Descriptions:

The Bank Management System, also known as "BMS," has been designed to provide users/customers with convenient access to their account information and allow them to manage their accounts privately from any location, eliminating the need to physically visit bank branches. By utilizing this system, users/customers can carry out a variety of banking transactions and activities online, without the requirement of being physically present. The main objective of the bank management system is to identify the essential features and requirements that will aid software developers in constructing the system, while considering the overall perspective of the clients.

The primary purpose of the "BMS" is to assist customers with financial transactions within a banking environment, offering multiple options for conducting banking tasks. Additionally, it maintains records of clients, employees, and other pertinent information within the bank. Customers have the ability to create accounts, make deposits and withdrawals, transfer funds, and view reports pertaining to their accounts. The Banking Management System ensures the smooth operation of management tasks and securely stores information about employees and their salaries.

The initial step to engage with the system involves registration, which empowers users to manage their finances without physically visiting a bank. Registration grants users' complete control over their accounts, enabling them to perform transactions and processes at their convenience. Online bank registration is a time-saving process that ensures secure operations through password protection and restricted access to critical system functions. Users can either log into an existing account or create a new one, recover or reset their data, and must verify and validate their information to finalize the registration or login process.

The system places significant emphasis on facilitating transactions through online operations. Users can easily transfer funds from their account to another by entering the recipient's bank IBAN or phone number, and they can also review their current balance before initiating a transfer. It can be challenging for bank customers to keep track of their deposit and withdrawal

transactions solely through messages. To address this issue, the system allows users to access a record of their deposit and withdrawal transaction details, provided they enter their account password for enhanced security and privacy.

1.2 Use Case Descriptions:

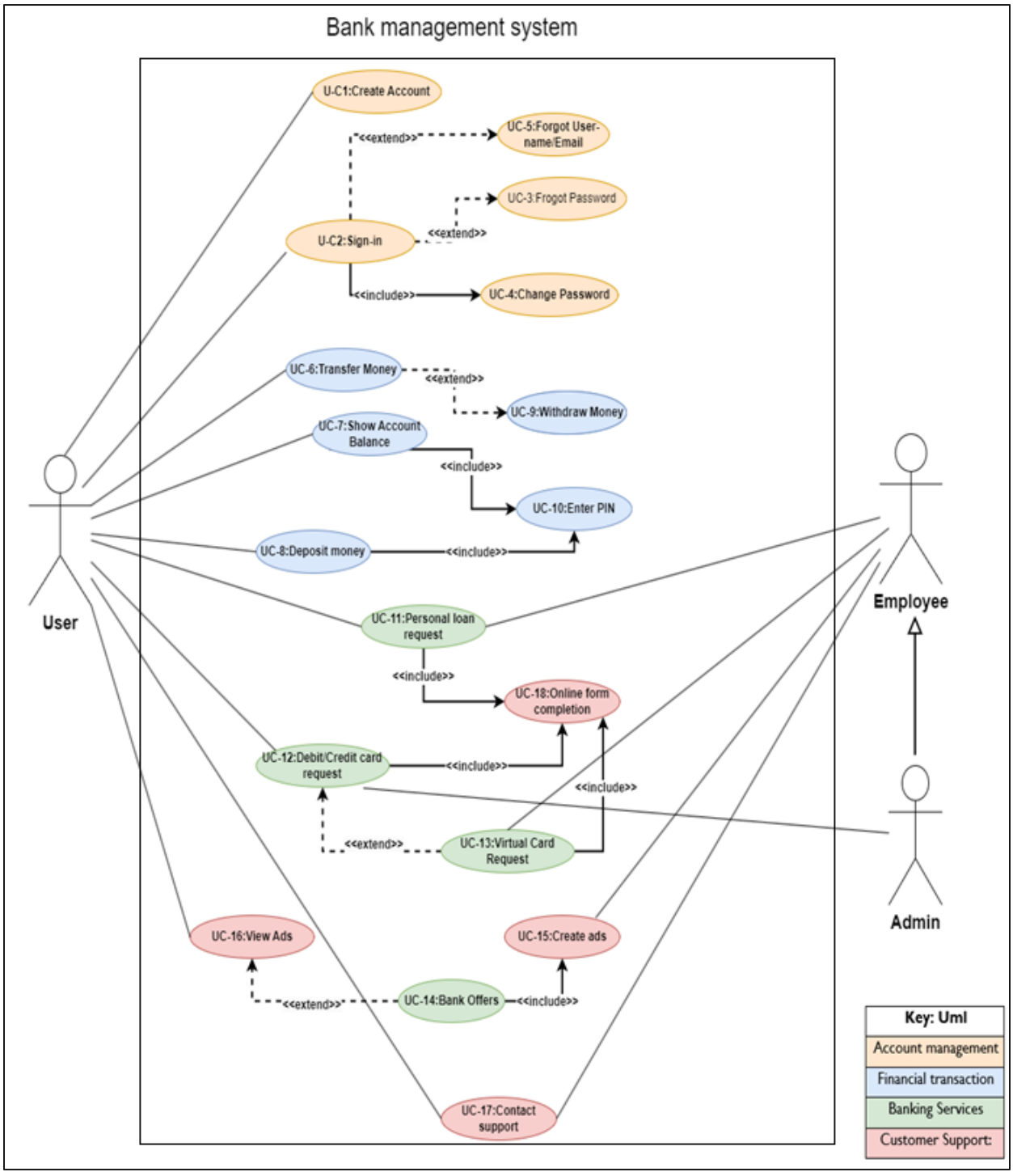
Use case	Description
UC-1: Create Account	This use case allows users to create a new account in the system. Users will be required to provide their personal information to the system. Once the information is validated, users can create a unique username and password to access their account.
UC-2: Sign-in	This use case permits users to access their accounts by providing their username and password. The system will then authenticate the credentials to verify that the user is authorized to access the account.
UC-3: Forgot password	This use case allows password recovery for users who have forgotten their passwords. The user must provide their registered email address or phone number, and the system will send them a link or code to reset their password.
UC-4: Change Password	This use case provides the functionality of changing account's password. Users will provide their current password, then enter and confirm their new password. After validation, the system will update the user's account with the new password.
UC-5: Forgot Username/Email	In case a user forgets their username or email, they can retrieve it by providing their registered phone number or email address. The system will send a message containing their username or email.

UC-6: Transfer Money	This use case allows users to transfer funds between their account and another user's account by providing the recipient's account IBAN or phone number, transfer amount, and transaction description. The system authenticates the transaction and processes it accordingly.
UC-7: Show Account Balance	To view their current account balance, users authenticate themselves by entering their password. The system then displays the account balance.
UC-8: Deposit money	Users specify the deposit amount, and the system generates a virtual account number for transferring funds. After transaction verification, the deposited amount is added to the user's account balance.
UC-9: Withdraw Money	Users can withdraw money from their bank account by specifying the withdrawal amount and ATM location. After account authentication, the system deducts the withdrawal amount from the user's balance and dispenses cash from the chosen ATM.
UC-10: Enter PIN	This use case compels the user to enter their PIN number to authenticate their access to specific bank management system features such as money transfers, withdrawals, and password changes.
UC-11: Personal loan request	Users can request a personal loan directly through the bank management system by specifying the amount, and the preferred repayment period. Bank employees evaluate eligibility and authorize requests.

UC-12: Debit/Credit card request	This use case allows users to request a new debit or credit card through the bank management system. Users specify the type of card, card limit, and other relevant details; The bank employee will process the request and generate the new card. The user will be notified about the availability of the new card and how to retrieve it.
UC-13: Virtual Card Request	Users can request virtual debit or credit cards through the system. They specify card details, and bank employees generate virtual cards. Users can access these details and use them for online purchases.
UC-14: Online Form Completion	This use case allows users to complete various online bank-related forms through the system. They provide required information, attach necessary documents, and submit the forms. After processing the form, the bank employee will notify the user of the outcome.
UC-15: Advertisement Creation	This use case allows bank employees to create advertisements through the bank management system to promote products or services. The employee can specify the target audience, the message, and the medium of the advertisement. The system will then display the ads to the intended audience.
UC-16: Advertisement Viewing	Users can view relevant advertisements through the system based on their interests or banking behavior. They have the option to interact with or ignore these advertisements.
UC-17: Contact Support	This use case enables users to contact the bank through the management system for assistance or inquiries. The user can choose the type of inquiry or issue they are experiencing, and the

	system will connect them with a bank employee who can provide help.
UC-18: Bank Offers	The bank system displays various offers and promotions to users, which may include discounts, rewards, or incentives for using the bank's products or services. Users can choose to interact with or ignore these offers.

1.3 Use Case Model:



2 System Requirements:

2.1 Quality Attributes:

ID	Quality Attribute	Scenario	Associated Use case
QA- 01	Security	The system employs robust security measures to safeguard customer data. This includes encryption of sensitive information, two-factor authentication, and continuous monitoring for any suspicious activities.	UC-2, UC-5, UC-6, UC-7, UC-8, UC-9, UC-10, UC-14
QA-02	Usability	The system is designed with a user-friendly interface, ensuring that customers can easily navigate and complete tasks. User feedback and intuitive design elements are prioritized.	ALL
QA-3	Performance	The system boasts exceptional performance, enabling swift transaction processing and quick access to account information within 15 ms. It can handle a high volume of simultaneous transactions without delays.	UC-6, UC-7, UC-8, UC-9, UC-11. UC-12
QA-4	Availability	The system is available 24/7, ensuring that customers can access their accounts and perform transactions at any time, day or night, without service interruptions.	ALL
QA-5	Reliability	The system is highly reliable, providing accurate processing of financial transactions and ensuring that balances and account information are up to date and error-free.	UC-6, UC-7, UC-8, UC-9, UC-11. UC-12, UC-13
QA-6	Scalability	The system is designed to scale seamlessly to accommodate a growing number of users and an increasing volume of transactions.	UC-6, UC-7, UC-8, UC-9
QA-7	Responsiveness	The system responds swiftly to customer requests, ensuring that actions like creating accounts, signing in, and changing passwords are processed without delay.	UC-1, UC-2, UC-3, UC-4, UC-6. UC-11
QA-8	Marketing and Customer Engagement	The system engages customers through targeted advertisements and promotions, enhancing their banking experience and encouraging them to explore bank offers.	UC-15

The priorities of the quality attribute scenarios is discussed in the following table:

Scenario ID	Importance to Customer	Difficulty of Implementation
QA-01	High	High
QA-02	Medium	Low
QA-03	High	High
QA-04	High	High
QA-05	Medium	Medium
QA-06	Low	Medium
QA-07	Medium	Medium
QA-08	Low	Low

2.2 Constraints:

ID	CONSTRAINTS
CON-1	The system should be available at least 99.99% of the time to ensure continuous access to banking services.
CON-2	The system must support multiple currencies and provide accurate exchange rates.
CON-3	The system should respond quickly to customer requests, ensuring swift calculations and transactions.
CON-4	The system should easily connect with external systems for smooth transactions and regulatory compliance.
CON-5	The system must maintain accurate and consistent customer data.
CON-6	Strict security measures must be in place to protect customer information.
CON-7	The system should work seamlessly on various platforms, including web, mobile, and desktop.

2.3 Concerns:

ID	Concern	Concern Description
CRN-1	Data Security	Ensure robust data security measures to protect customer information, such as encryption, authentication, and monitoring.
CRN-2	Transaction Performance	Achieve optimal transaction performance for quick processing and account access, with low latency and high efficiency.
CRN-3	System Availability	Maintain 24/7 system availability to ensure uninterrupted access to banking services for customers at any time.
CRN-4	Data Accuracy	Guarantee the accuracy and consistency of transaction and customer data stored within the system to prevent errors.
CRN-5	Scalability	Design the system to scale seamlessly, accommodating a growing user base and increasing transaction volumes effectively.
CRN-6	Integration with External Systems	Ensure the system can integrate with external systems and services as required for banking operations

3 Design Process:

We will now design the system by implementing the Attribute Driven Design process.

3.1 Iteration 1: Establishing an Overall System Structure

3.1.1 ADD step 1: Review Inputs

Categories	Details
Design purpose	The project is a mature Greenfield engineering system with the initial goal of identifying the overall system structure to support system development.
Primary functional requirements	The primary functionality is derived from the following use cases presented before: • UC-2: Sign-In: Critical for ensuring secure access to the system, protecting user accounts and sensitive information.

<table>
<tr><td></td><td>

- UC-6: Transfer Money: Core functionality supporting the primary purpose of banking.
- UC-8, 9: Deposit and Withdraw money: Fundamental banking transactions.

</td></tr>
<tr><td>Quality attribute scenarios</td><td>The quality attribute scenarios that were listed have now been prioritized in the following order:
<table>
<tr><th>Scenario ID</th><th>Importance to Customer</th><th>Difficulty of Implementation</th></tr>
<tr><td>QA-01</td><td>High</td><td>High</td></tr>
<tr><td>QA-02</td><td>Medium</td><td>Low</td></tr>
<tr><td>QA-03</td><td>High</td><td>High</td></tr>
<tr><td>QA-04</td><td>High</td><td>High</td></tr>
<tr><td>QA-05</td><td>Medium</td><td>Medium</td></tr>
<tr><td>QA-06</td><td>Low</td><td>Medium</td></tr>
<tr><td>QA-07</td><td>Medium</td><td>Medium</td></tr>
<tr><td>QA-08</td><td>Low</td><td>Low</td></tr>
</table>
</td></tr>
<tr><td>Constraints</td><td>All the constraints listed before are included as drivers.</td></tr>
<tr><td>Concerns</td><td>All the architectural concerns have been chosen as drivers.</td></tr>
</table>

3.1.2 ADD Step 2: Establish Iteration Goal by Selecting Drivers

The iteration goal is to achieve an overall structure supporting key constraints and quality attributes.

Prioritized drivers derived from use cases and quality attributes:

- UC-2: Sign in functionality.
- UC-6: Transfer money functionality.
- UC-11: Allows user to request a personal loan directly through the bank management system.
- CON-5: Data accuracy.
- CRN-3: Maintain 24/7 system availability.
- CRN-6: Integration with external systems.
- QA-1: Strict security measures.
- QA-2: Usability.
- QA-3: Performance.

- QA-7: Responsiveness.

3.1.3 ADD Step 3: Choose One or More Elements of the System to Refine

The whole Online Bank Management System is chosen as the element for refinement as this is a greenfield development effort. Below is the context diagram of how the system works generally.

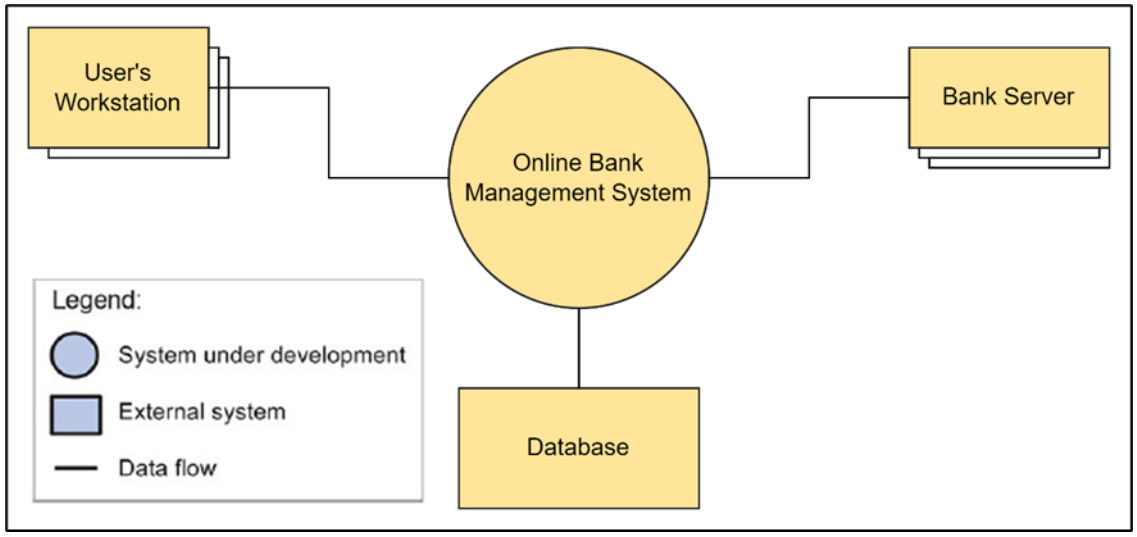


Figure 1.1: Context Diagram of the System

3.1.4 ADD Step 4: Choose One or More Design Concepts That Satisfy the Selected drivers

Below is a summary of design decisions made regarding the choice of design concepts:

Design Decision and Location	Rationale
The client-side of the system will be structured using a mobile application reference architecture.	Opting for a mobile application as the client-side structure will elevate the overall user experience, aligning with the essence of providing seamless and user-friendly banking interactions. The presentation layer in the reference architecture is essential for providing a user-friendly interface (QA-2). Recognizing the growing trend of mobile-centric user interactions in the digital landscape ensures that our system adapts to modern user preferences.
Alternative	**Reason for Discarding**

Web Application	The web application conflicts with QA-2, providing a user-friendly interface, which is of high importance to the customer.
Rich Client Application	Our decision prioritizes user-friendly interfaces, adaptability to contemporary trends, and robust security measures. In contrast, the rejection of rich client architecture is based on concerns related to security and portability.
Logically structure the server part (data source) of the system using the Service Application reference architecture	Service Application offers services provided by the server-side without a user interface, aligning with the system's requirements. No other alternatives considered; Service Application architecture is considered fully adequate.
Physically structure the application using the three-tier deployment pattern.	A three-tier deployment is appropriate for accessing the system via handheld devices, and it aligns with CRN-5. Adopting a three-tier architecture ensures a secure separation of concerns. With critical data residing in the middle-tier, we reduce the risk of unauthorized access and enhance overall system security. Two-tier is discarded as it does not support data source needs. All n>3 alternatives are unnecessary and may bring additional costs and complexity.
Build the user interface of the mobile application using the Spring framework.	The Spring framework is chosen for its reliability. It is a basic framework, and most developers are familiar with its syntax. Alternatives, like Ionic, were considered but not chosen due to unfamiliarity.
Exception Management	Implement user-friendly error messages (CON-3, QA-2) to inform users about errors. Implement features that guide users to avoid errors, contributing to the overall usability and aligning with QA-2.

3.1.5 ADD Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Below is a summary of the design decisions considered and made and their reasoning in the instantiation process:

Design Decision and Location	Rationale
Instantiate a data encryption mechanism for sensitive customer information.	We employ some encryption algorithms to maintain data integrity and confidentiality (CRN-1) and (CON-6).
Implement a continuous monitoring system to detect and respond to any suspicious activities without delay.	Security is a major concern as this is a banking system. Implementing a continuous monitoring system to detect and respond to any suspicious activities promptly will contribute to maintaining the security and reliability of the overall banking system (QA-5).

3.1.6 ADD Step 6: Sketch Views and Record Design Decisions

The diagram presented below (Figure: 1.2) depicts the adapted sketches of the module view of the reference architectures that were selected for our system. These diagrams were adapted according to the design decisions made.

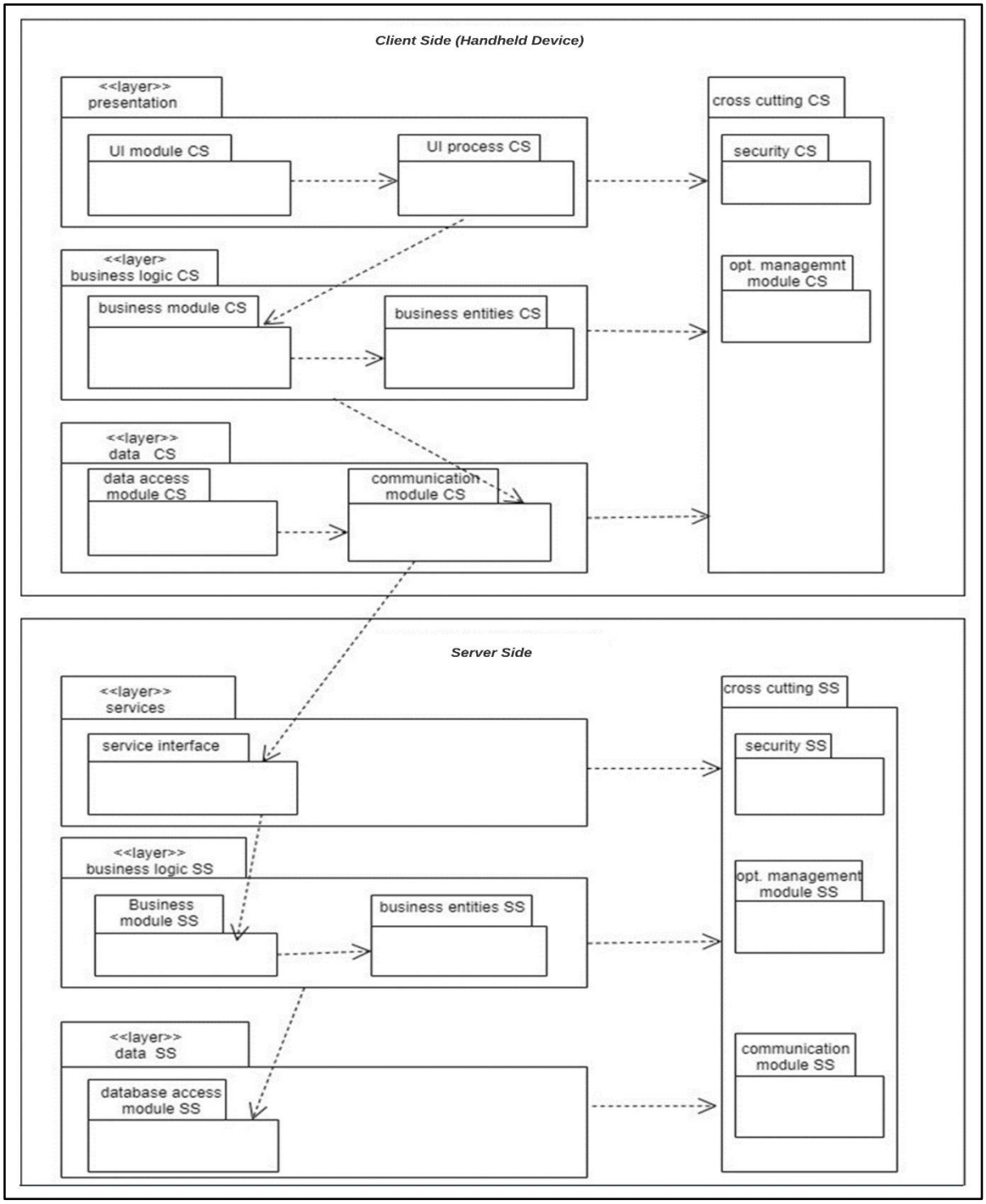


Figure 1.2: Module Sketches

These sketches were made using the aid of draw.io, an online tool. Each element and a short description of its responsibilities is captured. Note that the descriptions at this point are quite raw, just indicating major functional responsibilities, with no details. The following table summarizes the information that is captured in the sketch.

Elements	Responsibility
Presentation Layer (Client Side)	This layer contains modules that control user interaction and UC control flow, contributing to QA-2.
Business Logic (CS)	This layer contains modules that enable core business logic functionalities directly on the client-side.
Data (CS)	Contains modules that provide a bridge between client and server sides. These modules are essential for maintaining data accuracy and consistency
Cross-Cutting Layer	This layer contains modules that provide functionality across different layers. This layer enables robust security measures (QA-1).
UI Module (CS)	These modules directly interact with the client and receive user input.
UI Process Module (CS)	These modules manage user input flow between use cases, ensuring a smooth and intuitive UI.
Business Module (CS)	These modules either implement business operations locally or import server-side business functionality.
Business Entities (CS)	These entities manage transactions and provide the domain model.
Communication Module (CS)	These modules use services provided by the server-side application.
Data Access Module (CS)	This module is used to access local data.
Security (CS)	This is a cross cutting module. It supervises and ensures proper security measures are taken at both server and client side.
Services (Server Side)	Contains modules that provide interfaces to the services consumed by the clients.
Business Logic (SS)	This layer contains modules that perform client requests (business logic operations) coming from the CS.
Data (SS)	This layer manages data access securely with Database Access and Communication SS modules.
Cross-Cutting (SS)	These modules have functionality across different layers such as security, opt management etc.
Service Interfaces (SS)	Exposes services that are consumed by the clients and facilitates communication between client and server.
Business Module (SS)	Performs business operations in parallel with service interfaces.
Business Entities (SS)	Constitutes the domain model.
Database Access Module (SS)	Accesses the database securely and converts object-oriented data into records.

Communication Module (SS)	Communicates with the servers.

The deployment diagram in Figure 3 sketches an allocation view that illustrates where the components associated with the modules in the previous diagram will be deployed.

In figure 1.3, a deployment diagram sketch is illustrated for our system. It is an allocation view that illustrates where the components associated with the modules in the previous diagram will be deployed.

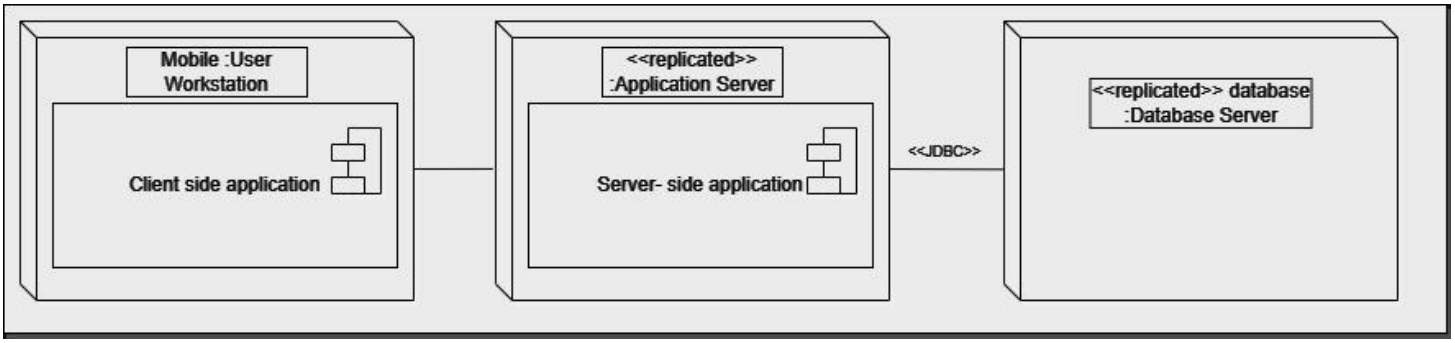


Figure 1.3: An initial deployment diagram for our system

Element responsibilities are described in a summary below:

Elements	Responsibility
User Workstation	User's mobile which hosts the client-side logic of the application.
Application server	The server that hosts the server-side logic of the application.
Database server	The server that hosts the database.

Information about relationships between some elements in the diagram that is worth recording is summarized in the following table:

Relationship	Description
Connection from Application Server to Database Server	We utilize the JDBC protocol for effective communication with the database.

3.1.7 ADD Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

We now summarize the design process for the first iteration using the Kanban board technique.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions made during the Iteration
	UC-2		The authentication process is supported by the selected reference architecture modules.
	UC-6		Money transfer functionality is supported by the implemented modules.
UC-8, UC-9			No relevant decisions taken yet.
		QA-1	Decision to implement a continuous monitoring system to detect and respond to any suspicious activities promptly at both mobile and service applications.
		QA-2	The reference architecture contributes significantly to achieving accessibility goals.
	QA-3		The chosen reference architecture fully supports the performance requirements of the system.
QA-7			No relevant decisions made.
CRN-3			No relevant decisions made.
	CRN-6		Initial steps have been taken to consider integration with external systems, but the depth of integration and compatibility need further evaluation.
		CON-5	The current design thoroughly addresses the data accuracy constraint through the implementation of a three-tier architecture. The dedicated data layer ensures effective measures for maintaining accurate information in the online banking system.

3.2 Iteration 2: Identifying Structures to Support Primary Functionality.

3.2.1 ADD step 2: Establish Iteration Goal by Selecting Drivers

In this iteration, the focus is on improving the user cases of Debit/Credit card request (UC-12), Sign-in (UC-2), and Transfer Money (UC-6) while addressing critical requirements. Two key critical requirements that will be considered are data security (CRN-1) and transaction performance (CRN-2). Robust data security measures, such as encryption and authentication, will be implemented to protect customer information during sign-in and transaction activities. Additionally, efforts will be made to optimize transaction processing speed and response time to ensure efficient and timely execution of debit/credit card requests and money transfers. By prioritizing these critical requirements, this iteration aims to enhance the security and performance aspects of the banking system, providing customers with a secure and seamless banking experience.

3.2.2 Step 3: Choose One or More Elements of the System to Refine.

In this iteration, we will concentrate on improving specific components found in the two reference architectures selected in the last iteration. At this point, our main goal is to enable the primary functionality, which is UC-2, UC-6, and UC-12. To do this, pieces in different layers must work together.

3.2.3 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers.

Design Decisions and Location	Rationale and Assumptions
Create a Domain Model for the application	Making an initial domain model that lists the primary players in the domain and their connections is crucial to ensuring that the system is developed correctly. The domain model should also include infrastructural components that support the system, business functionalities, and domain ideas. There is no other practical option except to create a domain model. In the absence of a domain model, the resulting architecture could be challenging to understand and maintain.
Identify Domain Objects that map to the functional requirements	Making an initial domain model that lists the primary players in the domain and their

	connections is crucial to ensuring that the system is developed correctly. The domain model should also include infrastructural components that support the system, business functionalities, and domain ideas. There is no other practical option except to create a domain model. In the absence of a domain model, the resulting architecture could be challenging to understand and maintain.
Decompose Domain Objects into general and specialized Components	Domain objects are essential elements that encompass all of the functionalities in the system along with the relationships that go along with them. These objects depend on components that are arranged in various layers and further subdivided into functionally specific modules, like user interface modules. There are no other options; breaking down the layers into modules is necessary to enable the necessary functionality.
Use Spring Java framework for IOS and Android	For developing applications for the iOS and Android operating systems, developers use the Java-based Spring framework. The JDBC technology, which increases efficiency and lowers system errors, is supported by the framework. Additionally, it provides testability and backward compatibility, which lowers the risk in subsequent versions. The main factor in the development team's decision to use the Spring framework over other options was their familiarity with Java, which allowed for increased efficiency.

3.2.4 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces.

Many instantiation design decisions have been made as a result of this iteration, and they are summarized in the following table:

Design Decisions and Location	Rationale

Create an initial domain model	The entities that take part in the use cases that were established in the earlier stage must be identified and modeled in this step. For this, a visual depiction of the entities and their connections is provided by the Class Diagram.
Map the system use cases to domain objects	We will generate domain objects for each of the previously chosen use cases to satisfy CON-5. Sequence diagrams will be used by us to locate and examine these domain objects.
Decompose the domain objects across the layers to identify layer-specific modules with an explicit interface	To guarantee that the modules cover every defined functionality and assign tasks to team members for UC-2, UC-6, and UC-12, we must employ a specific methodology.
Associate frameworks with a module in the data layer	Our system will integrate ORM, an existing solution, to satisfy CON-5 standards. The transformation of things into relational objects that can be kept in a database is the responsibility of this solution. Our development team will save time and ensure database consistency by employing this solution instead of having to design custom code for this activity.

3.2.5 Step 6: Sketch Views and Record Design Decisions.

Multiple diagrams have been created. Figure 2.1, which shows the initial domain object, is one of these.

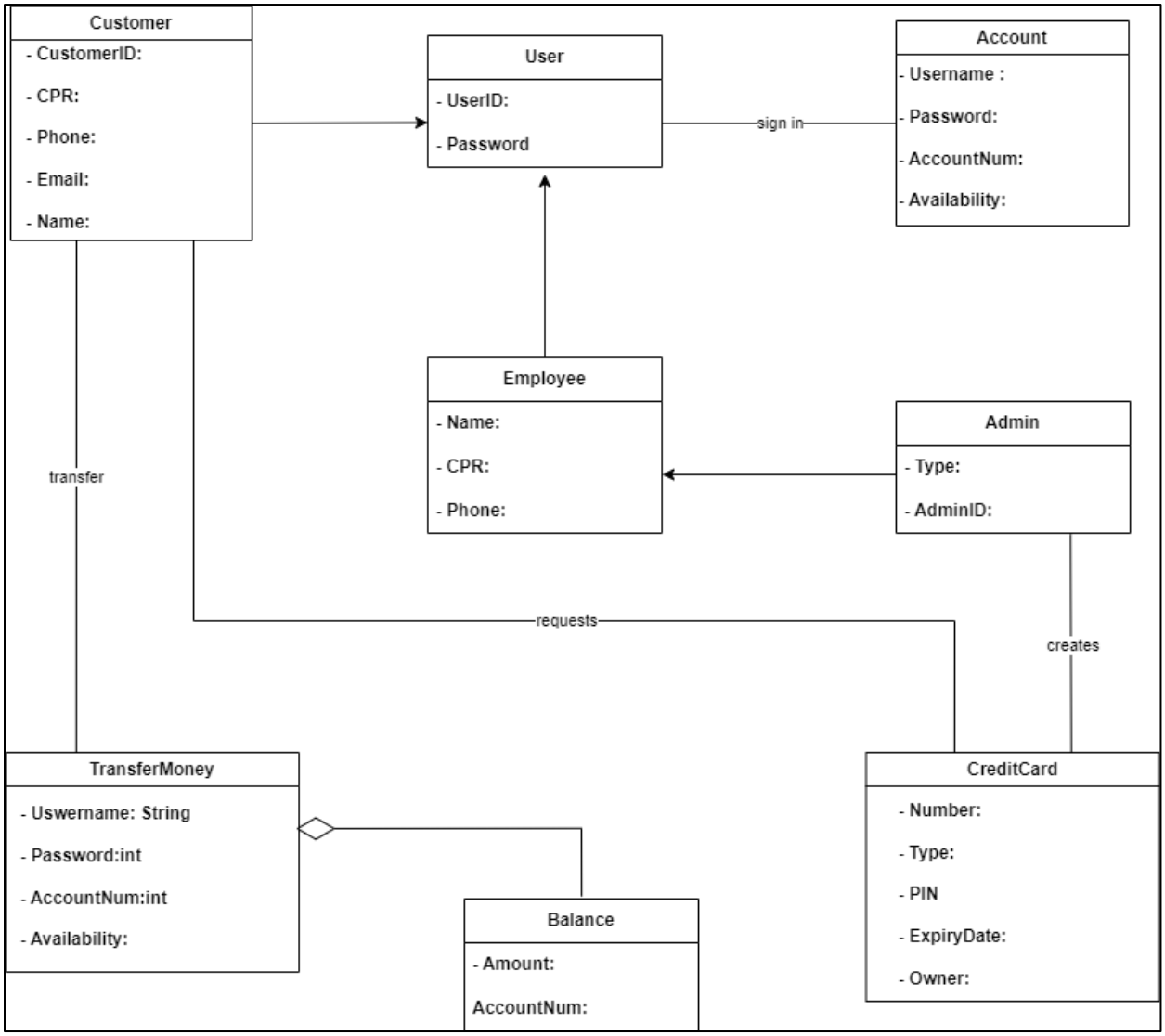


Figure 2.1: UML class diagram

This is the domain objects for the use case

model (Figure 2.2)

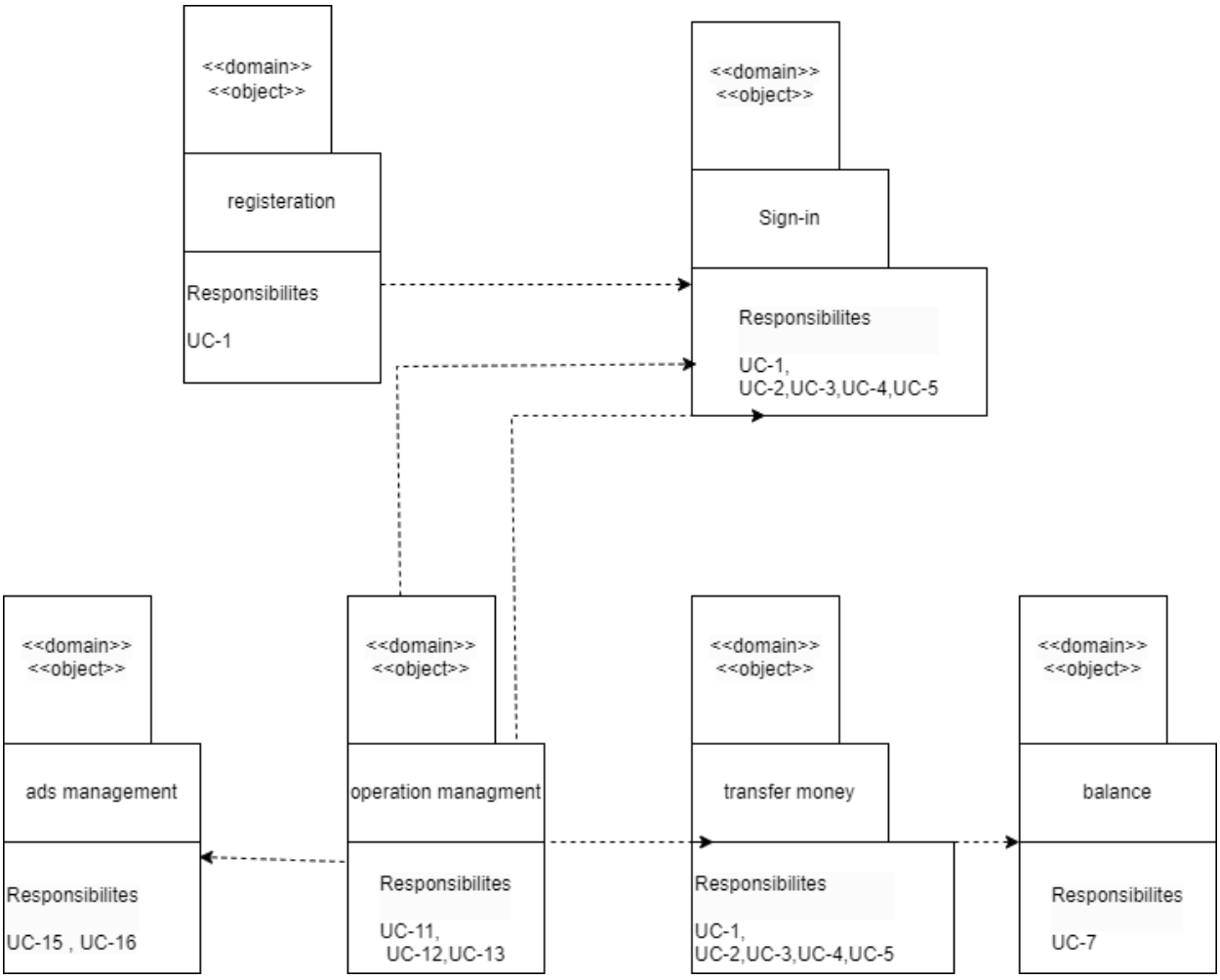


Figure 2.2: Use cases Domain objects

Modules that support the primary use cases **Figure 2.3**

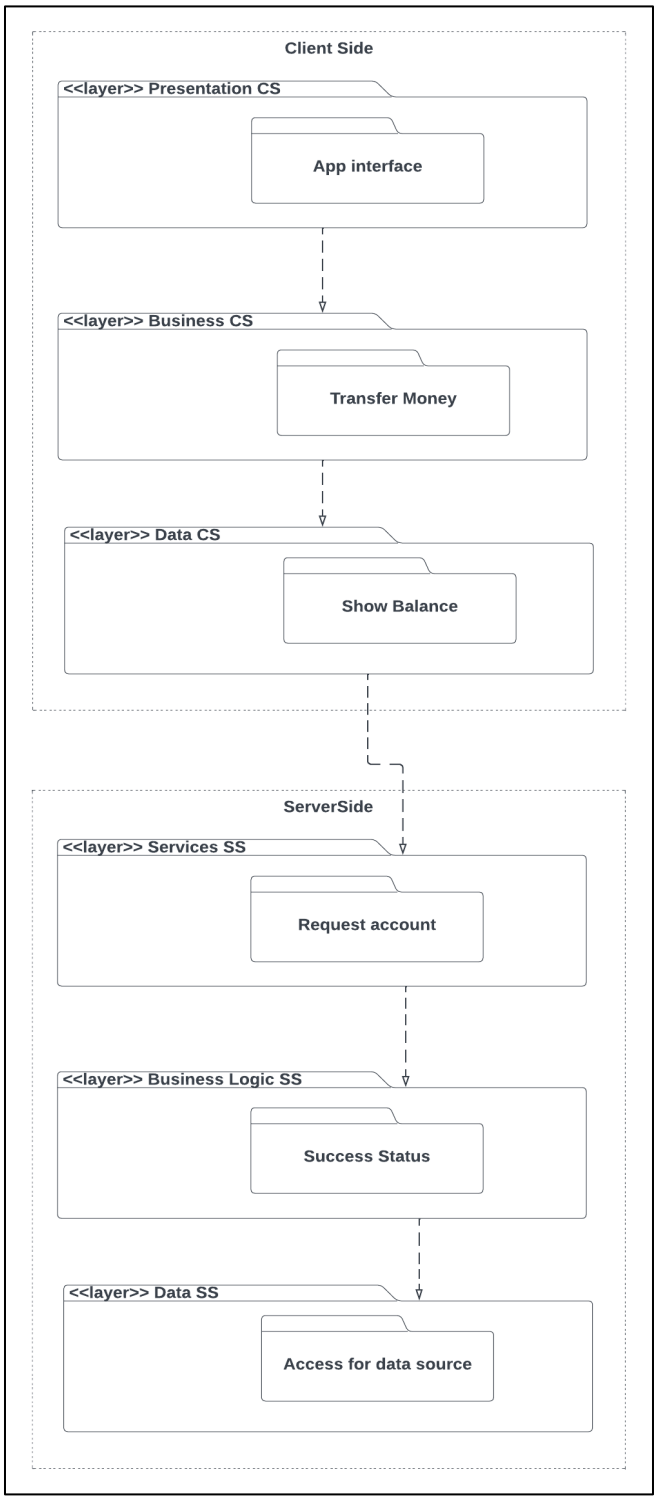


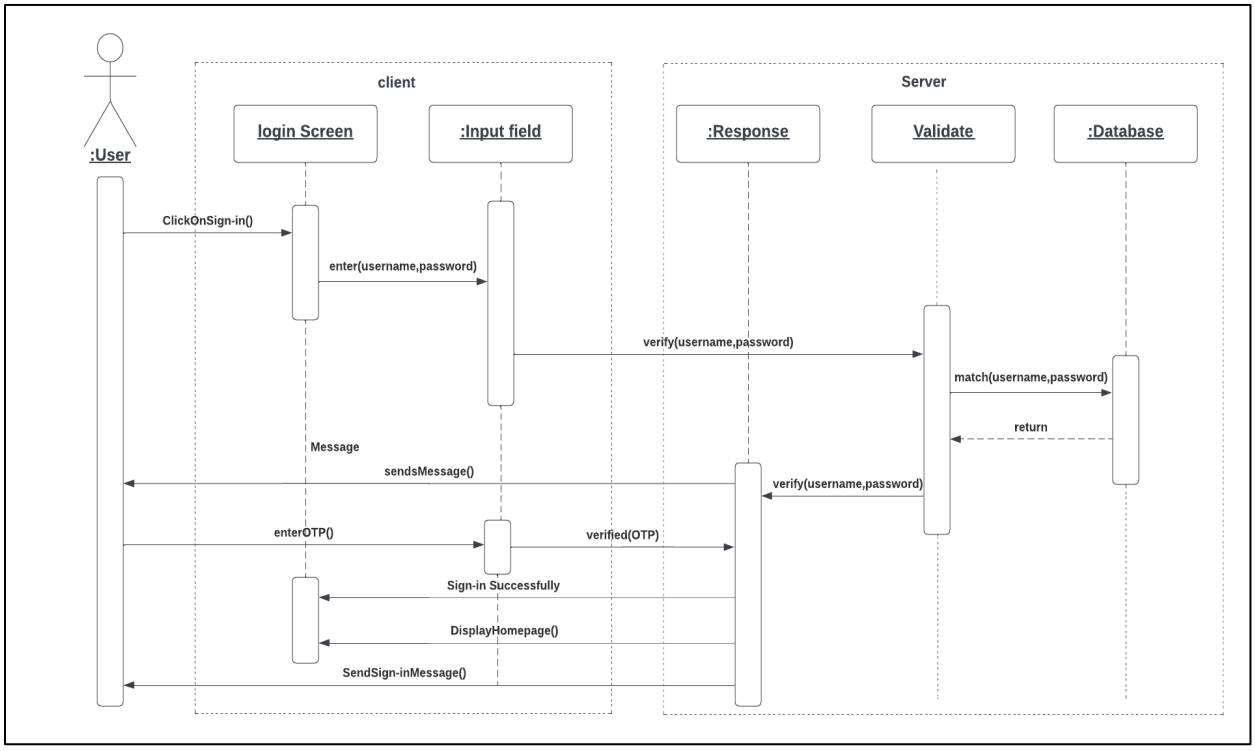
Figure 2.3

The functional responsibilities of the components depicted in Figure 2.3 are outlined in the following table:

Element	Responsibility
App interface	This module oversees displaying the user interface to the user so they can choose from the alternatives that are offered to them.
Transfer Money	This module oversees producing the necessary data or information, such as the user's entered amount of money, for the presentation layer.
Balance	This module is responsible for handling the processing and storage of data through its logical operations.
Request account	Through its logical functions, this module manages the processing and storing of data.
Success Status	This part oversees taking requests from the client and giving the underlying system logic a more straightforward interface.
Access for data source	This module oversees monitoring server-side data storage as well as system-level functions.

The following sequence diagrams for UC-2, UC-6 and UC-12 were created in the previous step of the method to define interface

UC-2: Sign-in:

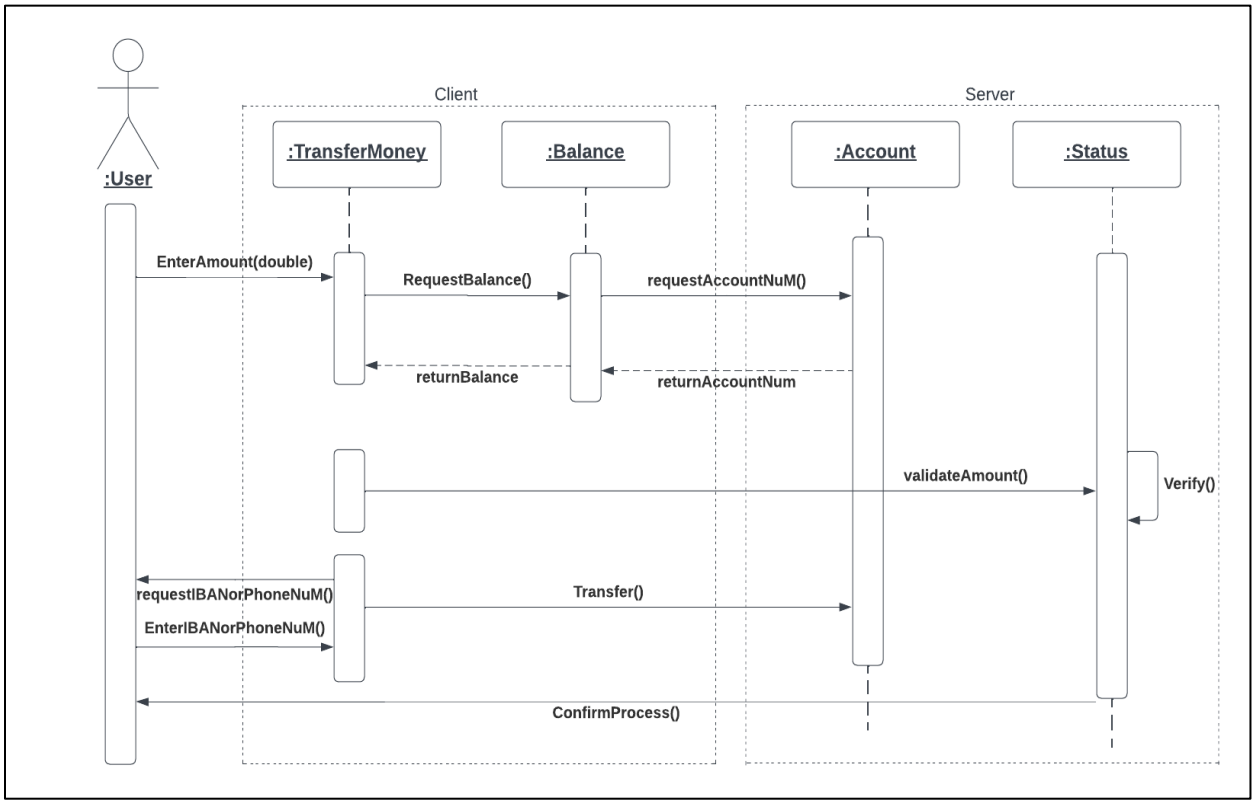


Sequence diagram for use case UC-2

Method Name	Description
clickOnSign-In()	The user initiates the sign-in process by clicking on the sign-in option.
enter(username,password)	The user enters their username and password into the input fields on the login screen.
verify(username,password)	The client verifies if the entered username and password are correct.
sendMessage()	If verification is successful, a message is sent to initiate OTP (One Time Password) authentication
enterOTP()	The user enters received OTP for further verification.

verified(OTP)	The OTP is verified by the server to ensure it matches with sent OTP.
Sign-in Successfully	If OTP verification is successful, user signs in successfully and gains access to their account.
DisplayHomepage()	After successful sign-in, users are redirected to their homepage or dashboard within application or website.
SendSignIn-Message()	A confirmation message of successful sign-in may be sent to notify users they have successfully accessed their accounts.

UC-6: Transfer Money

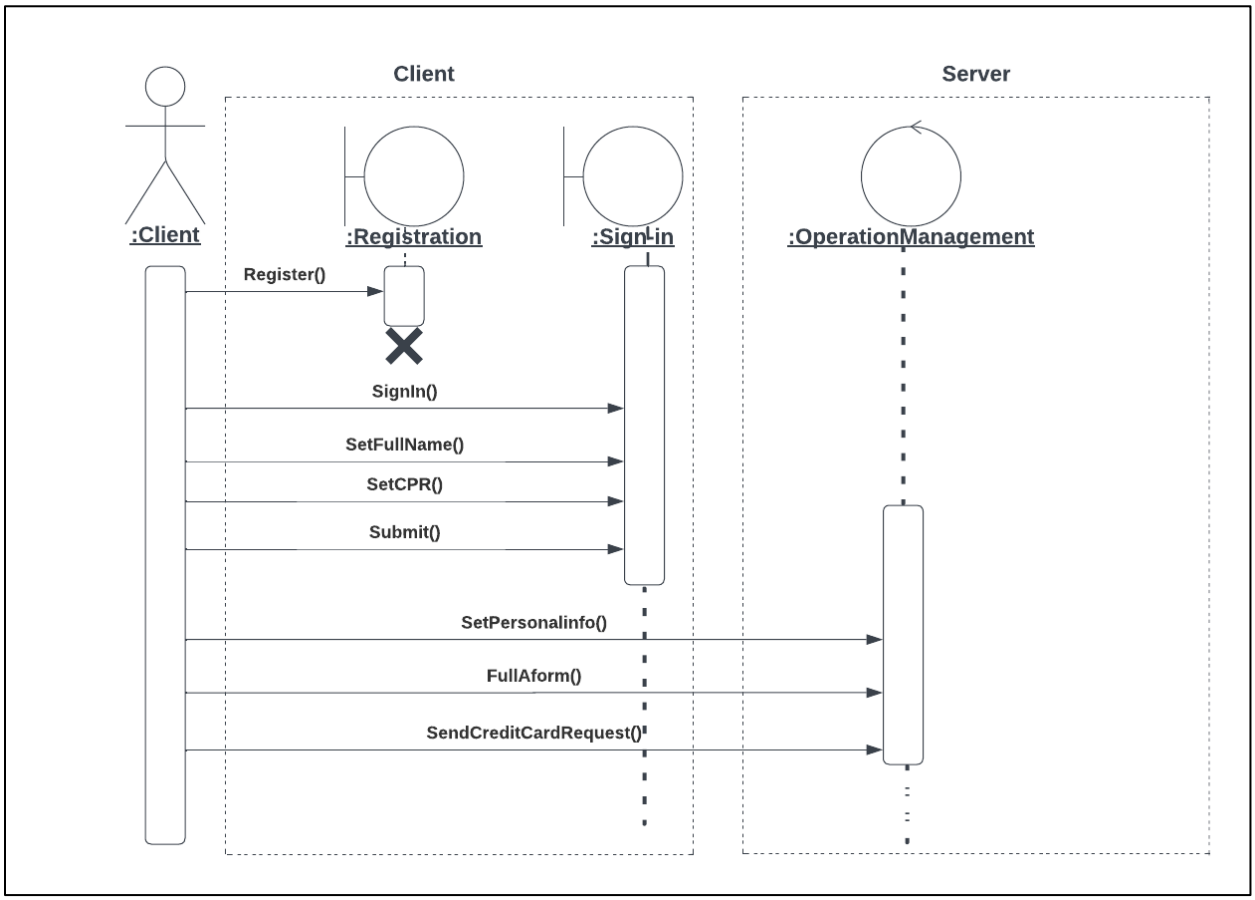


Sequence diagram for use case UC-6

Element Description is given below:

Method Name	Description
EnterAmount(double)	The user enters the amount of money to be transferred.
RequestBalance()	The client requests the current balance of the user's account from the server.
requestAccountNum()	The client requests the account number of the user from the server.
returnBalance	The server returns the current balance of the user's account to the client.
returnAccountNum	The server returns the account number of the user to the client.
requestIBANorPhoneNum()	The client requests the IBAN or phone number of the recipient from the user.
EnterIBANorPhoneNum()	The user enters the IBAN or phone number of the recipient.
Transfer()	The client initiates the transfer of money to the recipient.
ConfirmProcess()	The client confirms the transfer process with the user.
validateAmount()	The server validates the amount of money to be transferred.
Verify()	The server verifies the transfer details and updates the database.

UC-12: Debit/Credit card request



Sequence diagram for use case UC-12

Method Name	Description
Register()	The client initiates the registration process by clicking on the register option.
SignIn()	The client attempts to sign in with their credentials.
SetFullName()	The client enters their full name into the input field on the registration screen.
SetCPR()	The client enters their CPR (Civil Personal Registration) number into the input field on the registration screen.
Submit()	The client submits their registration details to the server.
SetPersonalInfo()	The client enters their personal information such as address, phone number, email, etc.

	into the input fields on the operation management screen.
FullAForm()	The client fills out a form with their credit card request details such as amount, duration, interest rate, etc. on the operation management screen.
SendCreditCardRequest()	The client sends their credit card request to the server for approval.

3.2.6 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose.

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During the Iteration
		UC-2	To support this use case, modules have been defined across various layers and diagrams.
		UC-6	To support this use case, modules have been defined across various layers and diagrams.
		UC-12	In support of this use case, modules have been defined across different layers and diagrams.
CON-2			No relevant decision has been made.
		CRN-2	The selected reference architecture is aligned with and supported by the identified use case.
	QA-1		Both the reference architecture for the mobile application and the service application in CS and SS incorporate the use of IT to safeguard users' data and all system files and data sources.

3.3 Iteration 3: Addressing Quality Attribute Scenario Drivers

(QA-1: Security) (QA-3: Performance)

We are expanding on the basic structural choices made in the first and second iterations in this section. This iteration's goal is to address QA-1 (Security) and QA-3 (Performance), another crucial quality attributes, and the scenario that goes along with them

3.3.1 Step 2: Establish Goad by selecting drivers

The protection of sensitive financial data, preventing unauthorized access, and maintaining regulatory compliance are the main drivers behind the iterative aim of (QA-1 security) in a bank management system. To secure the integrity of the system and protect client data, the goal involves putting strong authentication, encryption, access controls, and frequent security audits into place.

In order to ensure optimal system (QA-3 Performance) for efficient banking operations, a bank management system's iteration aim for performance is influenced by elements which includes successful transaction processing, fast response times, scalability, optimal resource use, and little system downtime, and to accommodate read operations for ad-hoc queries as well as frequent user queries

3.3.2 ADD Step 3: Choose One of More Elements of the System to Refine

- Bank Server
- Database Server
- Application Server

3.3.3 ADD Step 4: Choose One or More Design Concepts That Satisfy the Selected Driver

Design Decisions and location	Rationale and Assumptions
Encryption of sensitive Data in (Data storage)	To prevent unauthorized access, encrypt sensitive financial data while it's at rest. A strong encryption process and key management system will be used.
Implementation of role-Based access control in (application layer)	Implement role-Based access control to ensure authorized access to sensitive bank data and functionality, users will be assigned specific roles with different levels of access

3.3.4 ADD Step 5: Instantiate Architectural Elements, Allocate

Responsibilities, and Define Interfaces

Design Decisions and location	Rationale
Deploy message queue on separate node	Using active redundancy as a tactic You can ensure that traps are not lost by deploying the message queue on a separate node.
Use load balancer and active redundancy in application server	By distributing traffic among several application servers using active redundancy, load balancers can improve security by guaranteeing consistent availability and reducing the consequences of server failures or attacks.
Implement load balancing and redundancy using technology support	Instead of creating an ad hoc, less developed, and more difficult to maintain solution, there are numerous technological methods for load balancing and redundancy that can be put into action.
Implement encryption of sensitive data at rest in Data storage	Accept incoming sensitive data to be stored securely , apply a strong encryption process to the received data before storing it ,persistently store the encrypted data in secure manner

Implement role-based access control in application layer	Verify the identity of users trying to access sensitive bank data and functionality ,allocate specific roles authenticated users based on their authorization level ,apply access control rules restrictions to prevent unauthorized access

3.3.5 ADD Step 6: Sketch Views and Record design Decisions

Refined deployment diagram

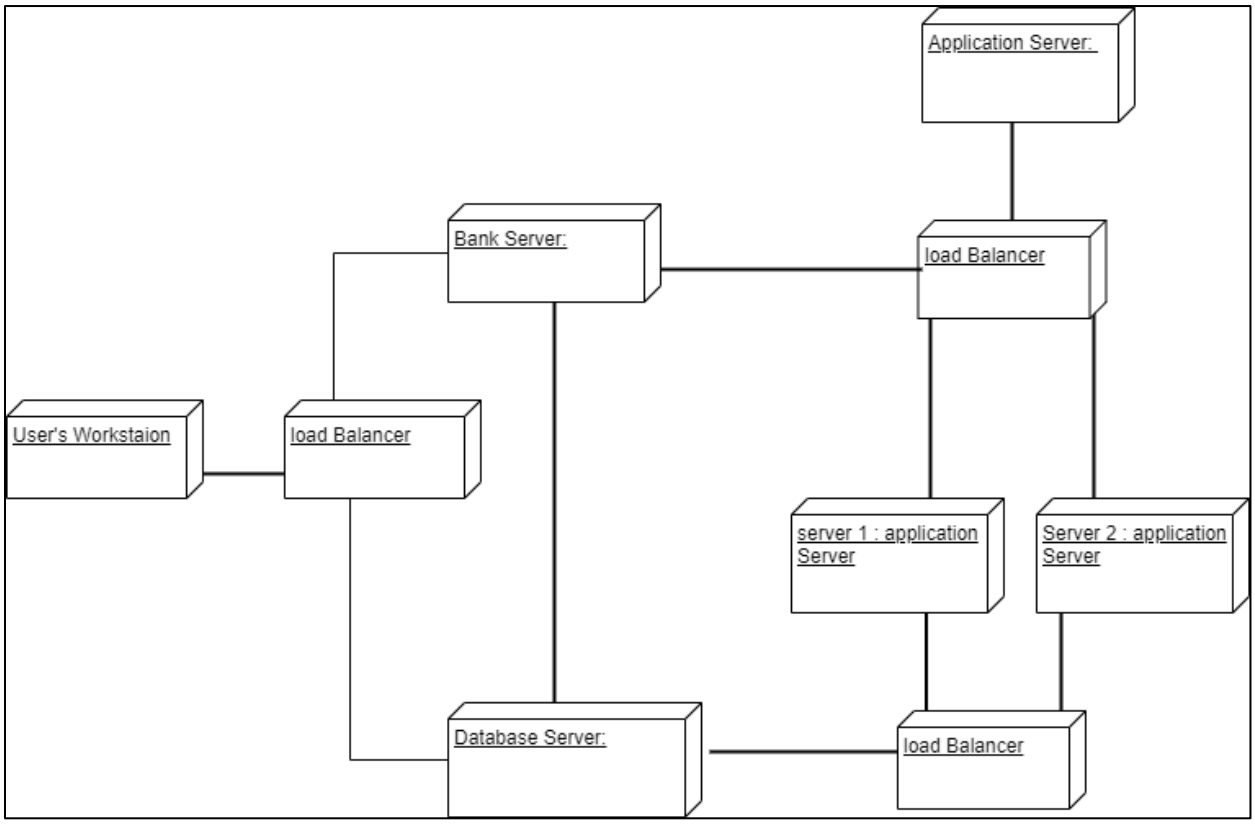


Figure 3.1 the refined deployment diagram

UML sequence diagram

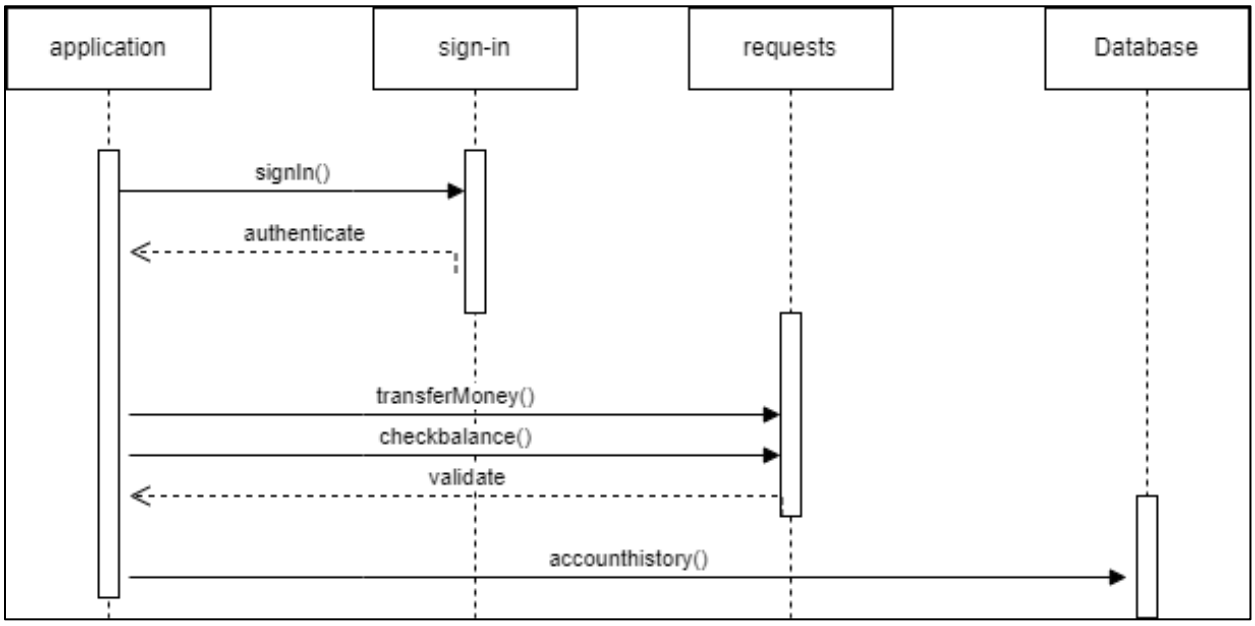


Figure 3.2 UML sequence diagram

3.3.6 ADD Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

QA-3 has been addressed in this iteration through significant design choices that also influenced QA-1. A summary of the decisions made during the iteration and the current state of each driver can be found in the following table.

Not addressed	Partially addressed	Completely Addressed	Design Decisions made during the iteration
		QA-3	The performance was an important element during the iteration it was supported in deployment diagram
		QA-1	The Security was fully addressed during this iteration in most of the steps
	CRN-2		During the sequence diagram Transaction Performance was addressed