CSE232 (Advanced Software Engineering) Report

Presented to: Dr. Gamal Abdel Shafy and Eng. Sally Shaker

Junior CESS

Project Name: Game Guard System

Names and student IDs:

Ahmed Mohamed Ibrahim Mohamed Elsheikh	21p0109
SAIFELDIN MOHAMED HATEM	21P0186
Basel Ashraf Fikry	22P0122
Ahmad Sabry Issa	17P8030
Mazen Ahmed Galal Selim	21P0103

Table Of Contents

Abstract

This project is about an internet café management system called Game Guard System. The system allows users to request account creation with the national ID, email and password. Those requests are processed by the admins via the system. Additionally the system is designed for users to access PCs or PlayStations based on predefined rates, Users prepay to add money which is converted to credits for usage. In addition, the admins have a management system where they can create/delete user accounts, manage details and view transaction history. Admins can also add new games which are synced across all PCs connected to the system.

1 Introduction

Figure 1.1 Logo of GGS

1.1 Purpose

The purpose of this document is to provide a detailed description of the Internet café management system, GGS, short for Game Guard System. This document will outline the system's design, architecture, and implementation details to facilitate the development team's understanding of the system's functionalities. The intended audience of this document includes project stakeholders, developers, quality assurance (QA) personnel, and anyone involved in the development and maintenance of the system. It provides a comprehensive overview of the system's design and serves as a reference for developers to ensure that the final product aligns with the requirements and specifications. The document will also aid in identifying potential design flaws and make modifications as needed, ultimately leading to the development of a high-quality and efficient system.

1.2 List of Definitions

GGS: An abbreviation for Game Guard System.
OMT: An abbreviation for Object Modeling Technique
DFD: An abbreviation for Data Flow Diagram

1.3 Scope

The scope of this document is to provide an in-depth description of GGS. It outlines the system's design, architecture, and implementation details, including software requirements, system constraints, and user functionalities. This document covers the system's scope, including the three types of users: normal users, employees, and administrators, and the functionalities available to them. It also covers the system's main features, such as user technical support, management of users, and user credit system. The document will provide guidance for the development team to ensure that the system meets the specified requirements and functions as intended. It also helps to identify any potential design flaws, which can be rectified during the development process, resulting in a high-quality and efficient system.

1.4 List of References

Use Case Diagrams: Use case diagrams were used to identify the system's functionalities and the interactions between the different types of users.

Activity Diagrams: Activity diagrams were used to model the system's workflow and help identify any potential design flaws or inefficiencies.

1.5 Overview

The Mobile App Functionality Overview for Game Guard System (GGS) offers a thorough exploration of the intricacies inherent in the proposed software. By using various functionalities, this document aims to provide a comprehensive understanding of the Game Guard System. Notably, the inclusion of diagrams such as use case, CRC (Class Responsibility Collaboration), and OOAD (Object-Oriented Analysis and Design) enhances its value for both developers and stakeholders, which aids effective communication between them. Also the document also contains the time plan for the project, it also contains the architectural model of the GGS. The document also contains the classes and methods used and their relationships between them. Those can be found in the class diagram in the document

2 General Description

2.1 Product Perspective

The internet management café system interacts many other systems to ensure so it can work as intended.

User Authentication Systems

The Internet Cafe Management System collaborates with user authentication systems to ensure secure and verified access. The specifics of this authentication system are yet to be determined, but integration with existing authentication mechanisms enhances overall security.

Communication and Support System

Support Requests feature links the system with communication and support platforms. Users can can seek technical assistance, request WiFi passwords, or place orders. Integration with communication channels streamlines user-admin interactions.

Frontend and Backend Technologies

The Internet Cafe Management System features a user-friendly mobile app (Frontend) developed with Android Studio. The Backend, implemented with Java simulation using OOP principles, interacts with Android ecosystems, ensuring a cohesive user experience.

2.2 General Capabilities

Sign-Up and Login

Users can ask to request to make an account using their national ID, email, and password. Admins take care of these requests.

Registered users can then later log into the system with the email/username and password that they used to register with.

Adding Money and Getting Credits

Users can put money in early, and it changes into credits for later use.

Getting Help and Ordering

Users can ask for help, request Wi-Fi passwords, or order snacks/drinks.

Adding New Games

Admins can put new games, and all computers will have them.

Saving Information

User details and transactions are kept in a pretend database in the computer's memory.

Admin Account Control

Admins can create or delete user accounts, manage details, and see transaction history.

2.3 General Constraints

Technology Limitations

The system's functionality is limited by the technology it has, such as the Java simulation on RAM without a traditional database. The absence of a database may affect scalability and data storage capabilities.

Authentication Undetermined

The method for user authentication is setters and getters in class objects, which puts a constraint on system security until a more suitable authentication mechanism is established.

2.4 User Characteristics

Regular Users

Account Creation

Regular users can request account creation by providing essential details such as national ID, email, and password.

Prepaid Credits

They have the ability to prepay, converting money into credits for usage within the system.

Credit Usage

Regular users experience credits deducted when accessing PCs or PlayStations, adhering to predefined rates.

Support Requests

They can submit requests for technical support, WiFi passwords, and order snacks/drinks.

Admins

Admins are divided into three types these are System Admins, IT/Tech Support and Cade Workers.

System Admins

This type of Admin controls users, passwords, signups, blacklisting and also processes requests from regular registrations. They also can blacklist regular users.

IT or Tech Support

These admins control syncing games across pcs and answering tech support calls and providing wifi passwords.

Cafe workers

The café workers are responsible for the approval and delivery of snacks or drinks to the regular user's station.

2.5 Environment Description

GGS operates through a user-friendly mobile app developed with Android Studio, catering to both regular users and administrators. Regular users engage in account creation, prepaid credits, and support requests, while administrators navigate the Java-simulated backend for system control. Financial transactions, communication, and gaming enhancements are seamlessly integrated. The system offers administrative operations, such as account management, and ensures operational flexibility with features like pausing time. User data is stored in a pseudo database, and authentication mechanisms are yet to be determined.

2.6 Assumptions and Dependencies

User Availability and Connectivity

The system assumes that users have reliable access to the internet and maintain connectivity throughout their interactions with the mobile app.

Hardware and Software Compatibility

The system assumes compatibility with the hardware and software configurations of the devices running the mobile app.

Storage and Data Handling

The system's functionality depends on the successful implementation of the storage structure for user data and transactions in the database.

3 Specific Requirements

3.1 Capability Requirements

A1: User Registration and Authentication

The system enables users to request account creation with national ID, email, and password. A secure and user-friendly authentication mechanism, yet TBD will be implemented to safeguard user accounts.

A2: Prepaid Credits Management

Users can prepay to add money, which is converted to credits for usage within the system. Admins have the capability to manage and adjust user credits based on financial transactions.

A3: Credit Usage and Transaction Tracking

Credits are deducted when users access PCs or PlayStations, following predefined rates. The system maintains a comprehensive transaction history for each user, allowing for transparent tracking and auditing.

A4: Support Requests and Communication

Users can submit support requests for technical assistance, WiFi passwords, and orders for snacks/drinks. This communication feature enhances user satisfaction and engagement.

A5: Account Management for Administrators

Admins have the authority to create and delete user accounts, manage user details, and view transaction history. This capability ensures efficient administrative control and user account oversight.

A6: Blacklisting and User Restriction

Admins can blacklist users to restrict their access, ensuring a secure and controlled user environment within the internet cafe.

A7: Game Server and Service Management

Admins can add new games or delete games and IT workers can add new services or delete services .This capabilities enriches the gaming experience for users.

A8: Café management

Café workers have the ability to add or remove items in the menu. Also they manage snack orders.

3.2 Constraint Requirements

B1: Connectivity Dependency

Constraint: The system is dependent on users having reliable internet connectivity for seamless interactions with the mobile app. Limited or inconsistent connectivity may impact the user experience.

B2: Administrative Expertise

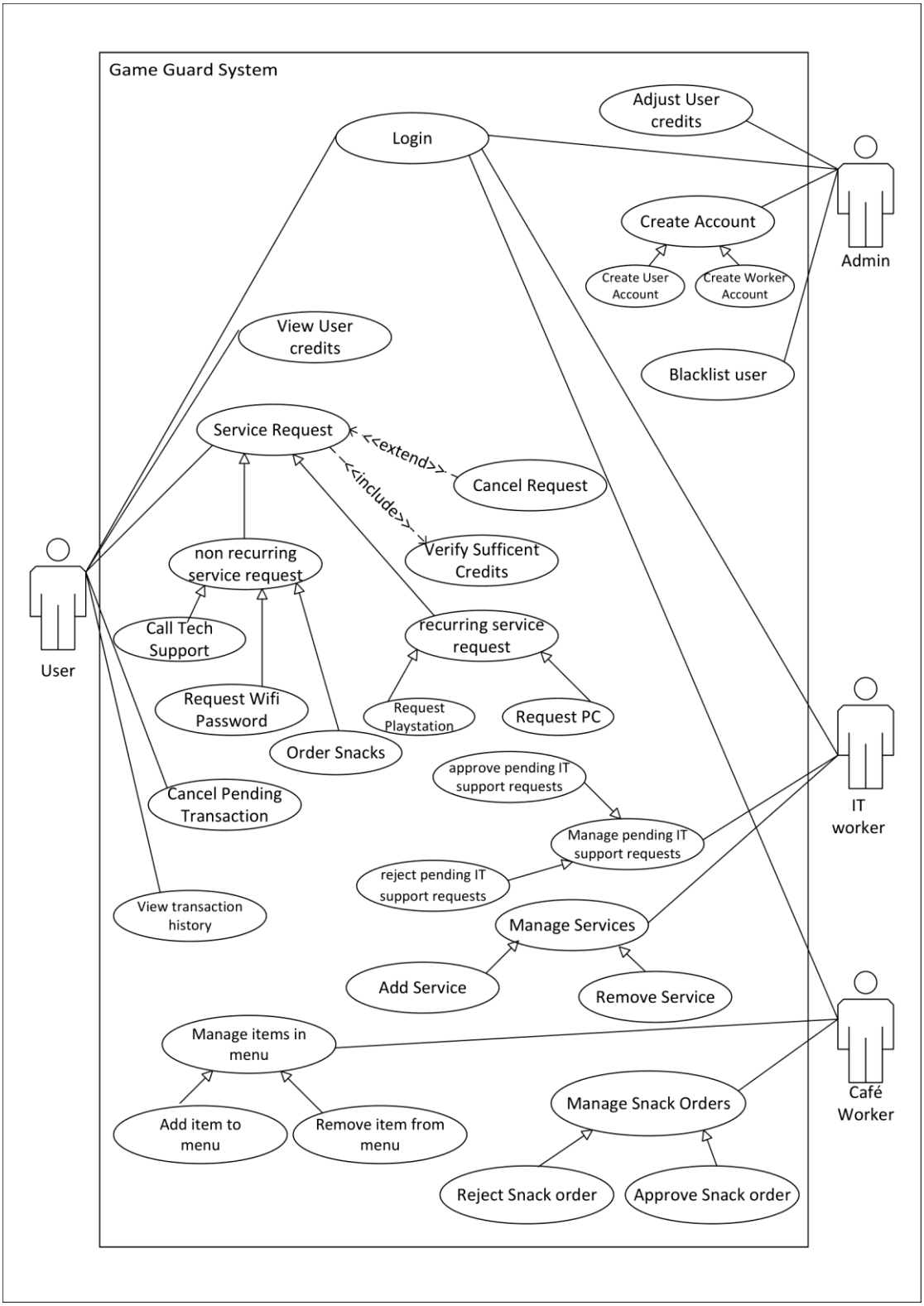
Efficient system administration relies on administrators possessing the necessary expertise and authorization to perform tasks such as account management, blacklisting, and system control. Admins with no experience might disrupt the system unintentionally and could cause problems.

B3: Communication Reliability

The functionality of support requests and user communication relies on reliable channels. Communication interruptions may impact users' ability to seek assistance or place orders like snacks.

4 Use Case Diagrams and Use Case Descriptions

4.1 Use Case Diagram



4.2 Use Case Descriptions

Figure 2 Use Case Diagram of GGS

Use case name: Login
Related Requirements: A.1.
Goal in Context: An existing user(regular user,admin,IT worker or café worker) enters their email and password
Preconditions: The User must be registered by an admin.
Successful End Condition: The User is directed into their user page.
Failed End Condition: The user gets a message telling them to attempt entering their credentials again.
Primary Actors: User,Admin,IT Worker and Café worker
Secondary Actors: None
Trigger: The User requests to log into the system.
Main Flow:

1. The User requests to log into the system
2. The User Enters username and password
3. The User's credentials are checked by the authenticator
4. The authenticator fetches accounts from the server
5. The server gives to approval to the authenticator
6. The User is directed to their user page

Extensions:
5.1 The server finds a mismatch in the credentials entered by the user and stored in the system
5.2 authenticator prompts user to re-enter credentials

Use case name: View User Credits
Related Requirements: A.3.
Goal in Context: The User sees their credits
Preconditions: The User must be logged in
Successful End Condition: The User's credits are displayed
Failed End Condition: None
Primary Actors: User
Secondary Actors: None
Trigger: The User requests to view their credits
Main Flow:

1. The User requests to view their credits from the system
2. The User credits are displayed

Use case name: Call Tech Support
Related Requirements: A.4.

Goal in Context: The User requests technical support.
Preconditions: The User must be logged in and the User must have sufficient funds
Successful End Condition: The User's request is fulfilled.
Failed End Condition: The User fails to get their service.
Primary Actors: User
Secondary Actors: None
Trigger: The User calls technical support
Included Use Cases: Verify Sufficient Credits
Main Flow:

1. The User requests a non recurring service
2. The user selects call technical support
3. **Include::Verify Sufficient Credits**
   The user's credits are checked for sufficiency
4. Credits are deducted from the user's credits based on the amount of credits the service costs
5. The User's service request is fulfilled.

Extensions:
1.1 The User cancels the service request
3.1 The User's service request is denied due to insufficient credits
3.2 The User is directed to their home page

Use case name: Request Wifi password
Related Requirements: A.4.
Goal in Context: The User requests Wifi password.
Preconditions: The User must be logged in and the User must have sufficient funds
Successful End Condition: The User's request is fulfilled.
Failed End Condition: The User fails to get their service.
Primary Actors: User
Secondary Actors: None
Trigger: The User requests wifi password.
Included Use Cases: Verify Sufficient Credits
Main Flow:

1. The User requests a non recurring service
2. The user requests wifi password
3. **Include::Verify Sufficient Credits**
   The user's credits are checked for sufficiency

4. Credits are deducted from the user's credits based on the amount of credits the service costs
5. The User's service request is fulfilled.

Extensions:

1.1 The User cancels the service request
3.1 The User's service request is denied due to insufficient credits
3.2 The User is directed to their home page

Use case name: Order Snacks
Related Requirements: A.4.
Goal in Context: The User order snacks.
Preconditions: The User must be logged in and the User must have sufficient funds
Successful End Condition: The User's request is fulfilled.
Failed End Condition: The User fails to get their service.
Primary Actors: User
Secondary Actors: None
Trigger: The User orders snack(s) .
Included Use Cases: Verify Sufficient Credits
Main Flow:

1. The User requests a non recurring service
2. The user selects wifi password
3. **Include::Verify Sufficient Credits**

    The user's credits are checked for sufficiency
4. Credits are deducted from the user's credits based on the amount of credits the service costs
5. The User's service request is fulfilled.

Extensions:

1.1 The User cancels the service request
3.1 The User's service request is denied due to insufficient credits
3.2 The User is directed to their home page

Use case name: Request Playstation
Related Requirements: A.3.
Goal in Context: The User requests to use playstation.
Preconditions: The User must be logged in and the User must have sufficient funds
Successful End Condition: The User's request is fulfilled.
Failed End Condition: The User fails to get their service.
Primary Actors: User
Secondary Actors: None
Trigger: The User selects the request playstation service.
Included Use Cases: Verify Sufficient Credits
Main Flow:

1. The User requests a recurring service
2. The user selects playstation service
3.

Include::Verify Sufficient Credits

The user's credits are checked for sufficiency

4. Credits are deducted from the user's credits based on the amount of credits the service costs
5. The User's service request is fulfilled.

Extensions:

1.1 The User cancels the service request

3.1 The User's service request is denied due to insufficient credits

3.2 The User is directed to their home page

Use case name: Request PC
Related Requirements: A.3.
Goal in Context: The User requests to use PC.
Preconditions: The User must be logged in and the User must have sufficient funds
Successful End Condition: The User's request is fulfilled.
Failed End Condition: The User fails to get their service.
Primary Actors: User
Secondary Actors: None
Trigger: The User selects the request PC service.
Included Use Cases: Verify Sufficient Credits
Main Flow:

1. The User requests a recurring service
2. The user selects PC service
3.

Include::Verify Sufficient Credits

The user's credits are checked for sufficiency

4. Credits are deducted from the user's credits based on the amount of credits the service costs
5. The User's service request is fulfilled.

Extensions:

1.1 The User cancels the service request

3.1 The User's service request is denied due to insufficient credits

3.2 The User is directed to their home page

Use case name: Verify Sufficient credits
Related Requirements: A.3.
Goal in Context: The User's credits are checked for sufficiency.
Preconditions: The User's account must be registered

Successful End Condition: The User's credits are sufficient for a service.
Failed End Condition: The User's credits are insufficient for a service.
Primary Actors: None
Secondary Actors: None
Trigger: The User requests a service.
Main Flow:
1. The User requests a service
2. The user's credits are checked
3. The user successfully gets the service requested.

Extensions:
2.1 The user's credits are insufficient
2.2 The user is directed to the user's homepage.
Use case name: Verify Sufficient credits
Use case name: Cancel Request
Related Requirements: A.3.
Goal in Context: The User requests to cancel a service request.
Preconditions: The User's account must be registered and must have a service request.
Successful End Condition: The service request is cancelled.
Failed End Condition: The service request is not cancelled.
Primary Actors: User
Secondary Actors: None
Trigger: The User requests to cancel a service request
Main Flow:
1. The User requests to cancel a service request
2. The system verifies the service has been requested
3. The service request is cancelled
4. The user is directed to the user's homepage.

Extensions:
2.1 The system fails to verify that the service has been requested
2.2 The cancellation attempt fails
2.3 The user is directed to the user's homepage

Use case name: Cancel Pending Transaction
Related Requirements: A.3.
Goal in Context: The User requests to cancel a pending transaction.
Preconditions: The User's account must be registered and must have a pending transaction.
Successful End Condition: The User's pending transaction is cancelled.
Failed End Condition: The User's pending transaction is not cancelled.
Primary Actors: User
Secondary Actors: None
Trigger: The User requests to cancel a pending transaction

Main Flow:

5. The User requests to cancel a transaction
6. The system verifies that is a pending transaction
7. The pending transaction is cancelled.
8. The user is directed to the user's homepage.

Extensions:

2.4 The system verifies that the transaction is not pending

2.5 The cancellation attempt fails

2.6 The user is directed to the user's homepage

Use case name: View transaction history
Related Requirements: A.3.
Goal in Context: The User requests to view transaction history
Preconditions: The User's account must be registered.
Successful End Condition: The User's transaction history is displayed.
Failed End Condition: The User's transaction history is not displayed.
Primary Actors: User
Secondary Actors: None
Trigger: The User requests to view transaction history
Main Flow:

1. The User requests to view transaction history
2. The User's transaction history is displayed.

Use case name: Adjust User Credits
Related Requirements: A.5. and B.2.
Goal in Context: The admin requests to adjust a user's credits
Preconditions: The User's account must be registered and the admin must be logged in.
Successful End Condition: The User's credits are modified by the admin.
Failed End Condition: The admin fails to modify the user's credits.
Primary Actors: Admin.
Secondary Actors: None
Trigger: The admin adjusts the user's credits
Main Flow:

1. The admin requests to modify a user's credits.
2. The admin enters the username of the desired user
3. The admin enters the amount of credits to change with the operation desired
4. The User credits are modified.
5. The admin is directed to the admin's homepage
6. The User is notified with their credit's modification

Extensions:

2.1 The admin enters an invalid username

2.2 The target user's credits are not modified
2.3 The admin gets to try to enter the username again

Use case name: Create User account
Related Requirements: A.1. and A.5.
Goal in Context: The admin processes a user account creation request
Preconditions: The User must have requested an account creation request
Successful End Condition: The user's account is created
Failed End Condition: The user's account is not created
Primary Actors: Admin.
Secondary Actors: None
Trigger: The User requests an admin to process his account creation request
Main Flow:

1. The User requests an admin to process his account request
2. The admin selects create account
3. The admin selects create user account
4. The admin enters the credentials of the user
5. The user is notified that his account creation request is successful
6. The User's account is created

Extensions:

1.1 The account creation request is invalid because of reasons like invalid credentials or late registration.
1.2 The account creation request is rejected
1.3 The User is notified with the rejection

Use case name: Create Worker account
Related Requirements: A.1. and A.5.
Goal in Context: The admin processes a worker account creation request
Preconditions: The worker must have requested an account creation request
Successful End Condition: The worker's account is created
Failed End Condition: The worker's account is not created
Primary Actors: Admin.
Secondary Actors: None
Trigger: The worker requests an admin to process his account creation request
Main Flow:

1. The worker requests an admin to process his account request
2. The admin selects create account
3. The admin selects create worker account
4. The admin enters the credentials of the worker
5. The worker is notified that his account creation request is successful
6. The worker's account is created

Extensions:

1.4 The account creation request is invalid because of reasons like invalid credentials or late registration.
1.5 The account creation request is rejected
1.6 The worker is notified with the rejection

Use case name: Blacklist user
Related Requirements: A.6.
Goal in Context: The admin requests to blacklist a user
Preconditions: The user must be registered and the admin must be logged in.
Successful End Condition: The user is blacklisted
Failed End Condition: The user is not blacklisted
Primary Actors: Admin.
Secondary Actors: None
Trigger: The admin requests to blacklist a user
Main Flow:

1. The admin requests to blacklist a user
2. The admin enters the username of the user to blacklist
3. The system verifies a registered user with that username.
4. The user is blacklisted

Extensions:

3.1 The system fails to verify the entered username
3.2 The user is not blacklisted
3.3 The admin gets to enter the username again

Use case name: Approve pending IT requests
Related Requirements: A.4.
Goal in Context: A new support request is approved by an IT worker.
Preconditions: The IT worker must be logged in and the support request is available.
Successful End Condition: The support request is approved.
Failed End Condition: The support request is denied.
Primary Actors: IT worker.
Secondary Actors: None
Trigger: An IT worker approves a support request
Main Flow:

1. A new support request is sent to an IT worker
2. The IT worker manage IT support requests
3. The IT worker selects approve pending IT support request
4. The User is notified with the request approval

Use case name: Reject pending IT requests
Related Requirements: A.4.
Goal in Context: A new support request is rejected by an IT worker.

Preconditions: The IT worker must be logged in and the support request is available.
Successful End Condition: The support request is rejected.
Failed End Condition: The support request is not rejected.
Primary Actors: IT worker.
Secondary Actors: None
Trigger: An IT worker approves a support request
Main Flow:
1. A new support request is sent to an IT worker
2. The IT worker select manage IT support requests
3. The IT worker selects reject pending IT support request
4. The User is notified with the request rejection

Extensions:
3.1 The IT worker fails to reject the pending IT support request
3.2 The User is notified with the failed attempt to reject the request

Use case name: Change wifi password
Related Requirements: A.4.
Goal in Context: The IT worker requests to change wifi password
Preconditions: The IT worker must be logged in.
Successful End Condition: The wifi password is changed.
Failed End Condition: The IT worker fails to change the wifi password
Primary Actors: IT worker.
Secondary Actors: None
Trigger: An IT worker attempts to change the wifi password
Main Flow:
1. The IT worker requests to change the wifi password
2. The IT worker enters the current wifi password
3. The System verifies the entered wifi password
4. The IT worker types the new wifi password
5. The wifi password is changed

Extensions:
3.1 The System fails to verify the entered password.
3.2 The IT worker is asked to re-enter the current wifi password.
3.3 The IT worker fails to change the wifi password

Use case name: Add service
Related Requirements: A.7.
Goal in Context: An attempt to add a new service
Preconditions: The IT worker must be logged in.
Successful End Condition: The service is added.
Failed End Condition: The service is not added.

Primary Actors: IT worker.
Secondary Actors: None
Trigger: An IT worker requests the system to add a new service
Main Flow:
1. An IT worker requests the system to add a new service
2. The IT worker selects the service type
3. The IT worker enters the service details
4. The service is created

Extensions:
3.1 The IT worker enters the name of an exisiting service
3.2 The system notifies the IT worker with a name conflict
3.3 The system prevents the IT worker from continuing the process
3.4 The IT worker gets redirected to his homepage.

Use case name: Remove service
Related Requirements: A.7.
Goal in Context: An attempt to remove an existing service
Preconditions: The IT worker must be logged in.
Successful End Condition: The service is removed.
Failed End Condition: The service is not removed.
Primary Actors: IT worker.
Secondary Actors: None
Trigger: An IT worker requests the system to remove a service
Main Flow:
1. An IT worker requests the system to remove a service
2. The IT worker enters the service name
3. The system verifies that the service exists
4. The service is removed

Extensions:
3.4 The System fails to verify that the service exists
3.5 The IT worker is notified that the service doesn't exist
3.6 The IT worker is directed to the IT worker's homepage.

Use case name: Add item to menu
Related Requirements: A.8.
Goal in Context: An attempt to add a new item to the menu
Preconditions: The café worker must be logged in.
Successful End Condition: The item is added to the menu.
Failed End Condition: The item is not added to the menu.
Primary Actors: Café worker.
Secondary Actors: None
Trigger: A Café worker requests the system to add a new item in the menu.
Main Flow:

1. A Café worker requests the system to add a new item in the menu.
2. The Café worker enters the details of the item
3. The café worker enters the pricing of the item
4. The item is added to the menu

Extensions:

2.1 The system notifies a conflict in the item being added(item already exists)
2.2 The café worker is directed back to the café worker's homepage
2.3 The item is not added.

Use case name: Remove item from menu
Related Requirements: A.8.
Goal in Context: An attempt to add a remove item in the menu
Preconditions: The café worker must be logged in.
Successful End Condition: The item is removed from the menu.
Failed End Condition: The item is not removed from the menu.
Primary Actors: Café worker.
Secondary Actors: None
Trigger: A Café worker requests the system to add a remove item from the menu.
Main Flow:

1. A Café worker requests the system to add a remove an item in the menu.
2. The Café worker enters the name of the item
3. The item is removed from the menu

Extensions:

2.1 The system notifies the worker that the item does not exist
2.2 The café worker is directed back to the café worker's homepage
2.3 The item is not removed from the menu.

Use case name: Approve Snack Order
Related Requirements: A.8.
Goal in Context: a new Snack Order request is sent to the café worker for approval
Preconditions: The café worker must be logged in.
Successful End Condition: The snack order is approved.
Failed End Condition: The Snack Order is rejected.
Primary Actors: Café worker.
Secondary Actors: None
Trigger: A Snack Order request is sent to the café worker for approval.
Main Flow:

1. A User orders a snack
2. The café worker gets notified with the snack order
3. The café worker selects manage snack order
4. The café worker selects approve snack order
5. The snack order is approved

Extensions:

3.1 The snack order gets cancelled by the user before approval

3.2 The snack order is not approved

Use case name: Reject Snack Order
Related Requirements: A.8.
Goal in Context: a café worker attempts to reject a snack order
Preconditions: The café worker must be logged in and the snack order must be valid.
Successful End Condition: The snack order is rejected.
Failed End Condition: The Snack Order is not rejected.
Primary Actors: Café worker.
Secondary Actors: None
Trigger: A café worker rejects a snack order.
Main Flow:

1. A User orders a snack
2. The café worker gets notified with the snack order
3. The café worker selects manage snack order
4. The café worker selects reject snack order
5. The snack order is rejected

Extensions:
3.3 The snack order gets cancelled by the user before rejection
3.4 The snack order is not rejected.

5 Swimlane Diagrams

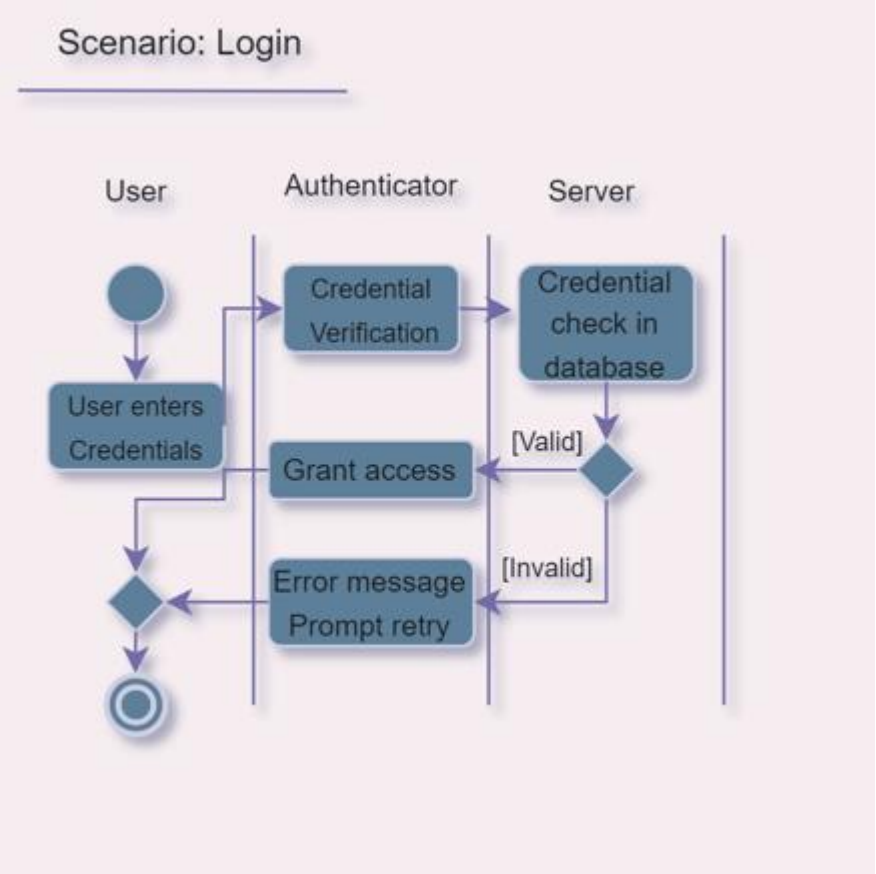


Figure 3.1 Login's swimlane diagram

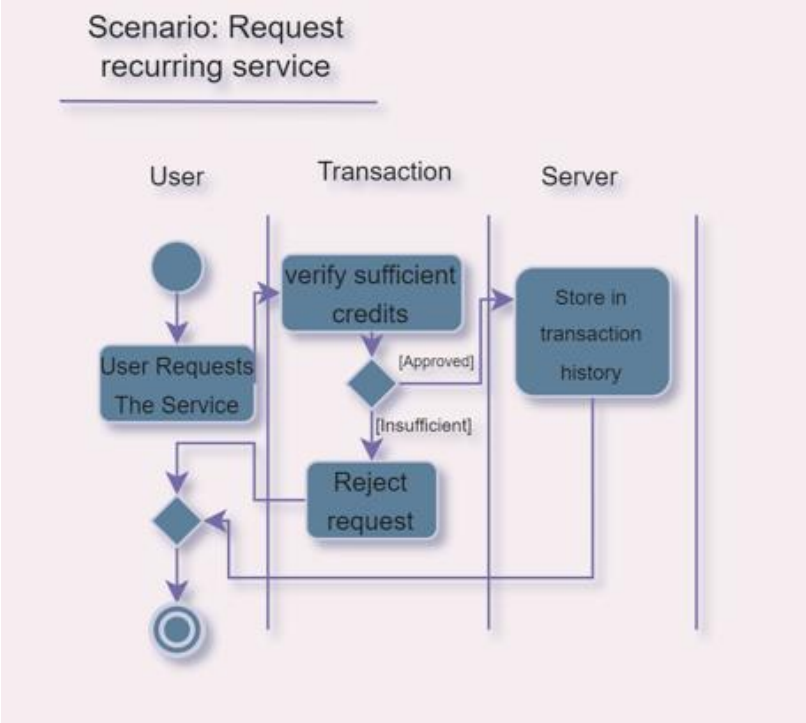


Figure 3.2 Request Recurring Service's swimlane diagram

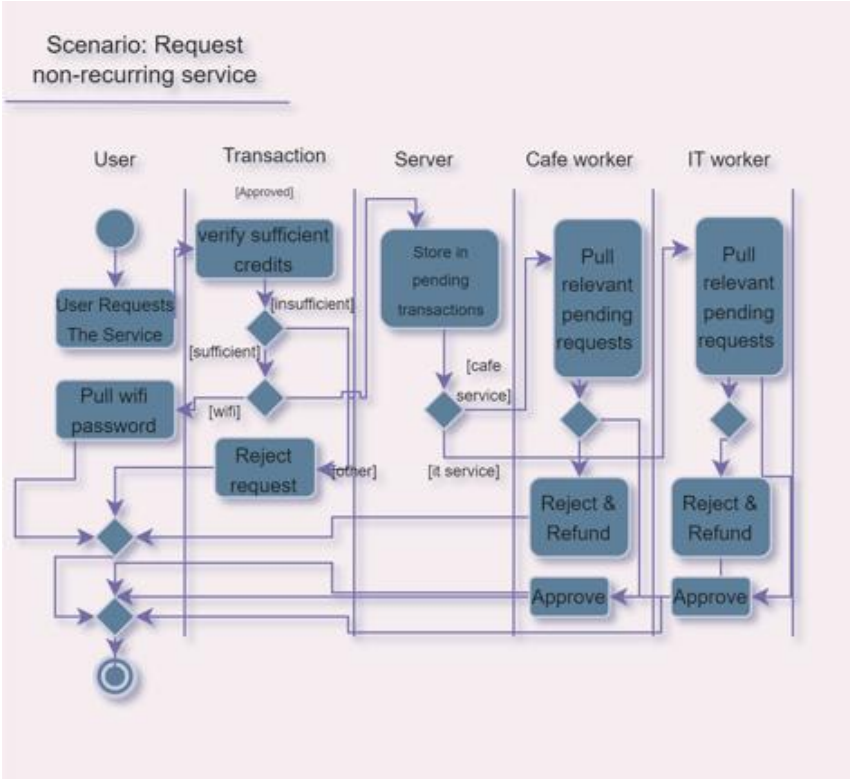


Figure 3.3 Request Non Recurring Service's swimlane diagram

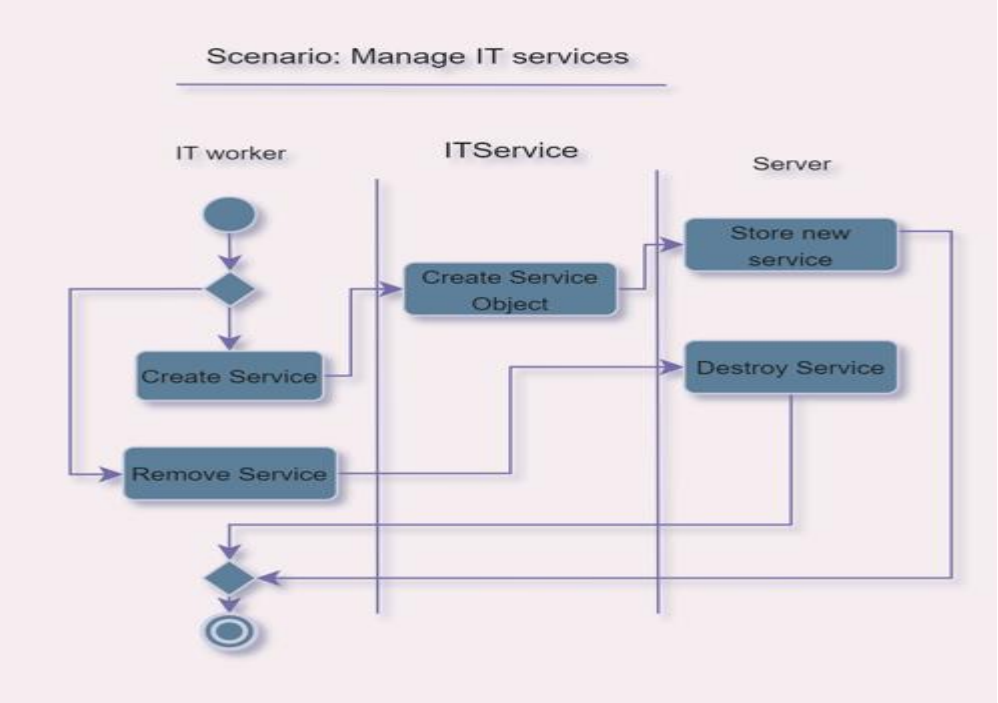


Figure 3.4 Request Manage IT services'
swimlane diagram

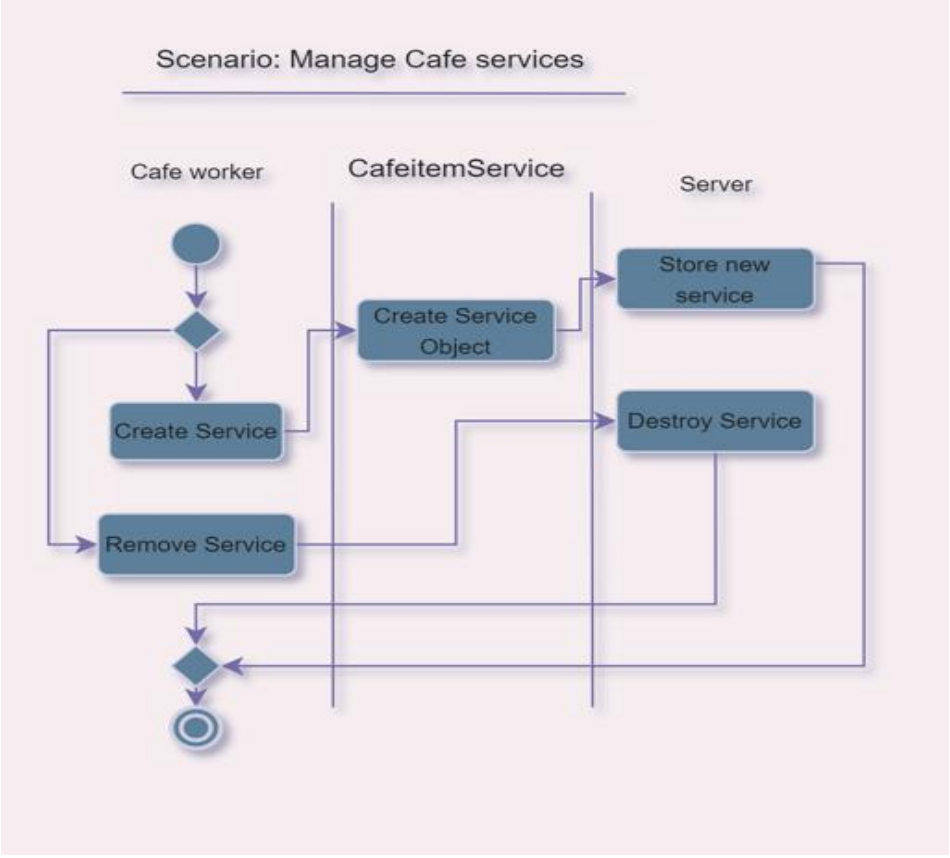


Figure 3.5 Manage Café services'
swimlane diagram

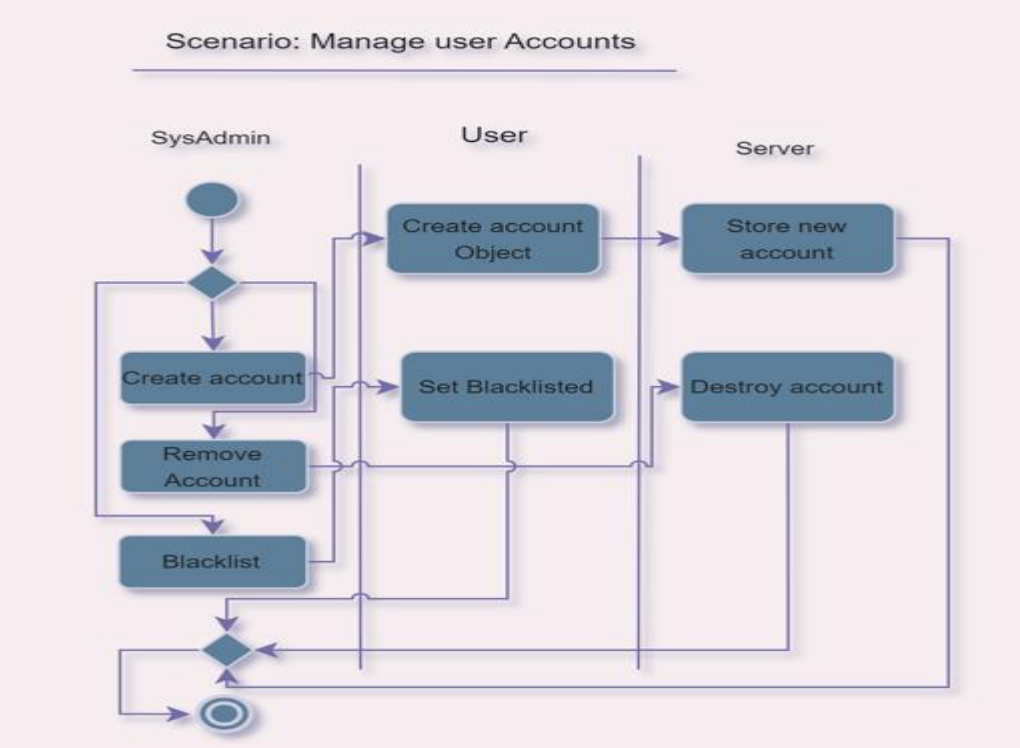


Figure 3.6 Manage user accounts' swimlane diagram

6 Sequence Diagrams

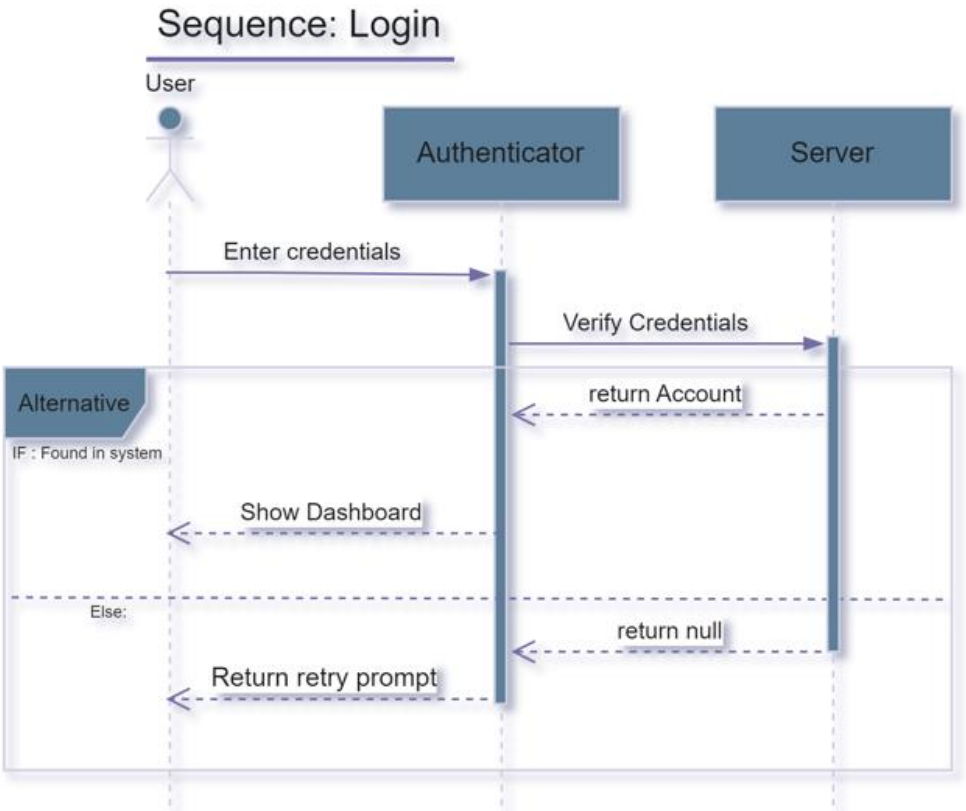


Figure 4.1 Login's sequence diagram

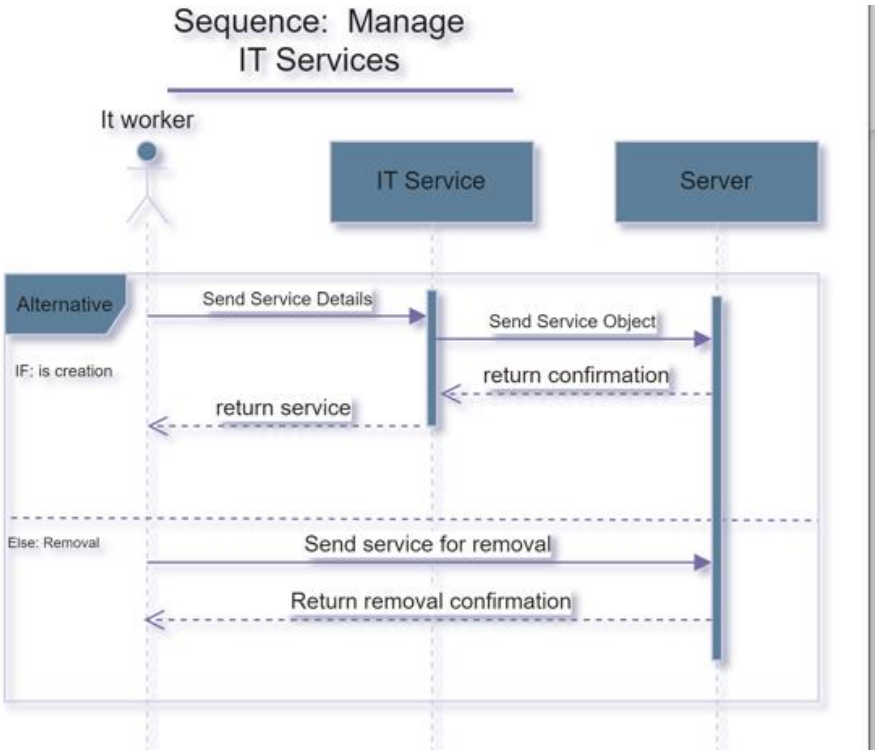


Figure 4.2 Manage IT Services' sequence diagram

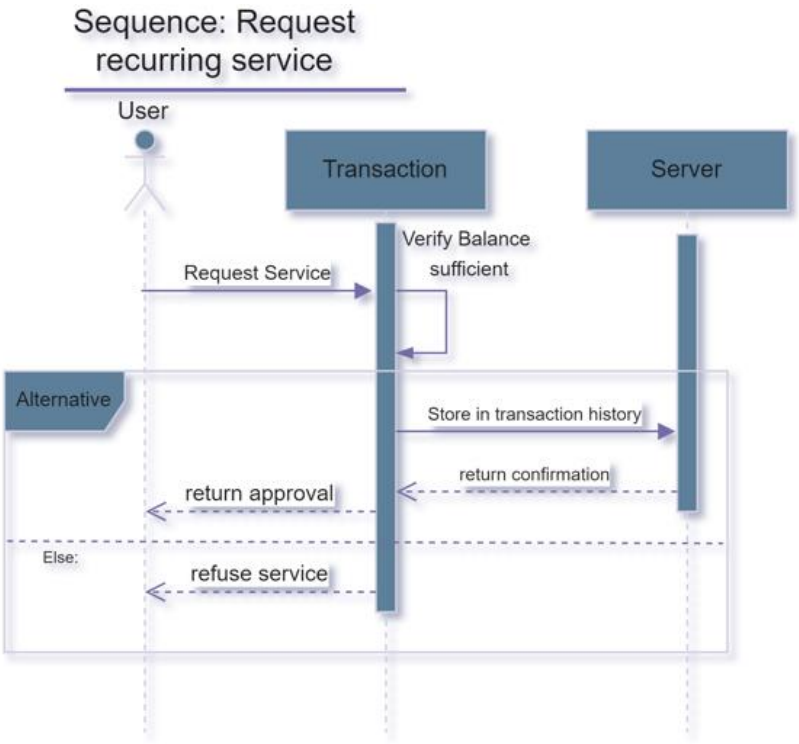


Figure 4.3 Request recurring service's sequence diagram

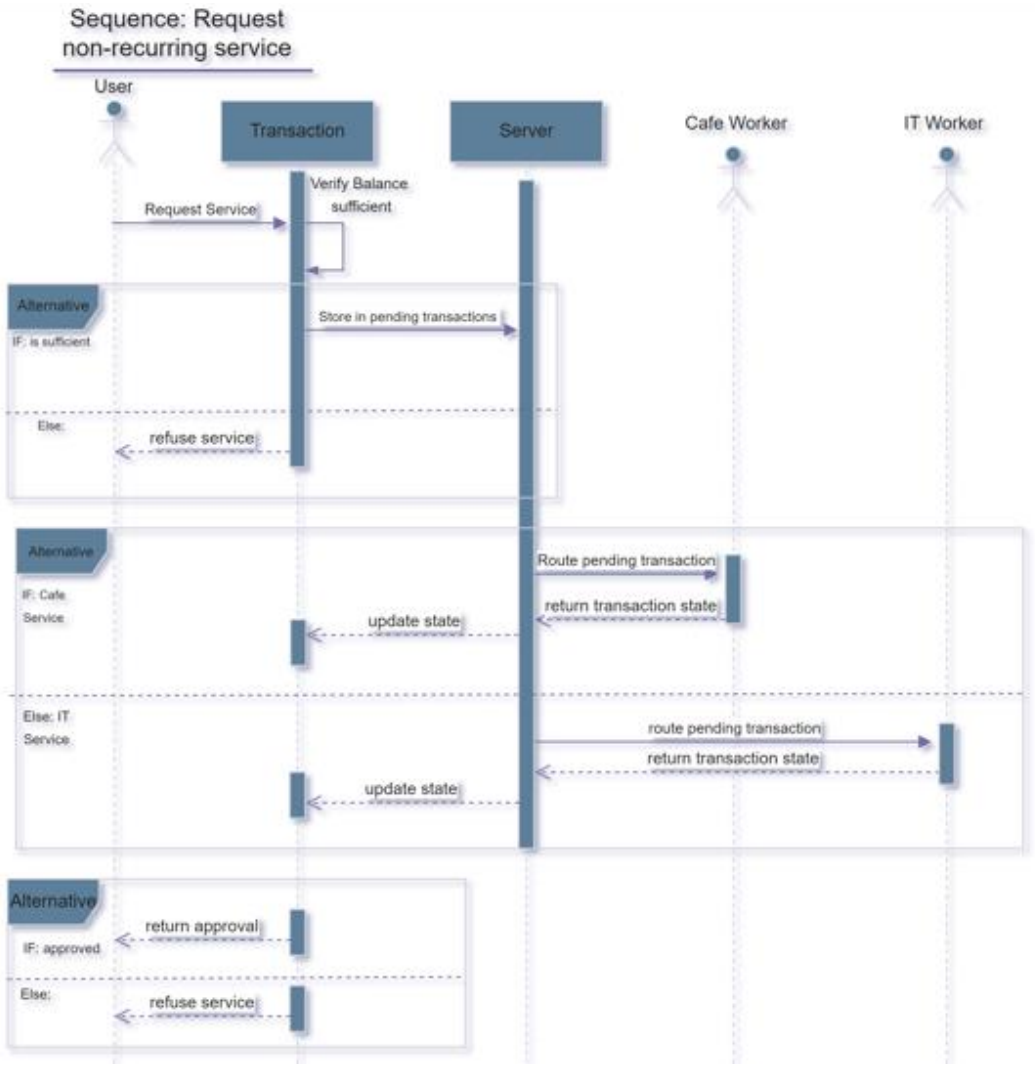


Figure 4.4 Request non-recurring service's sequence diagram

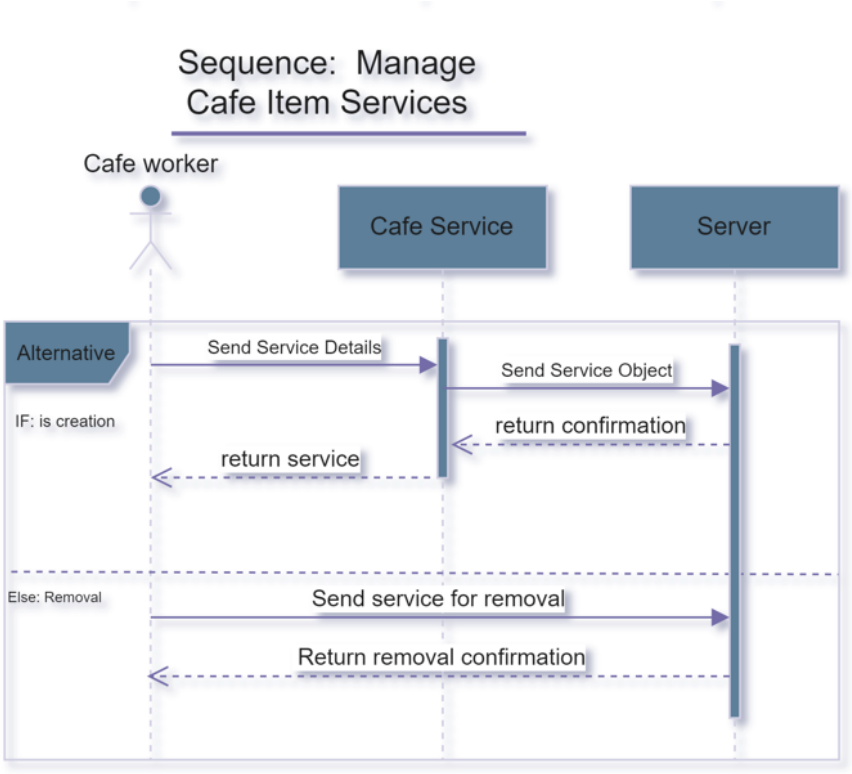


Figure 4.5 Manage Café Item Services sequence diagram

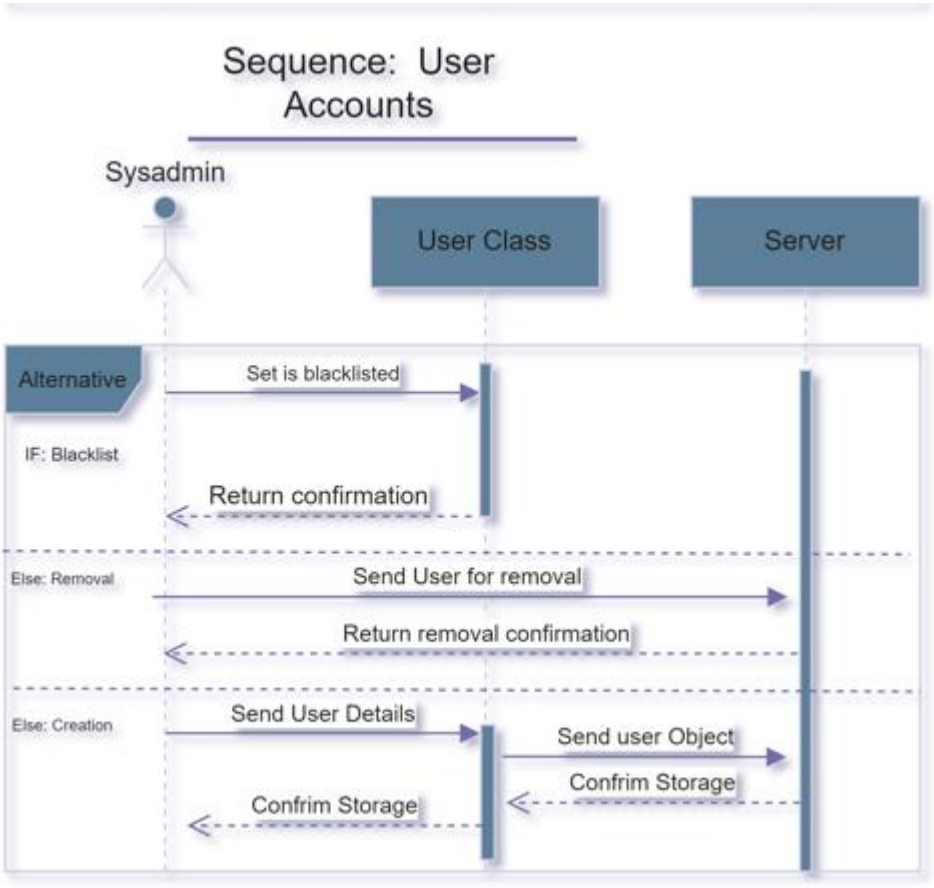


Figure 4.6 User Accounts' sequence diagram

7 Noun Extraction and CRC Cards

First Class

USER CLASS
Responsibility
USER CLASS heightheightTake User requests to **Server**
Take User requests to **Authenticator**
Request charging of credits from **Sysadmin**
Request **IT Service** from **Server**
Request **Pc Service** from **Server**
Request **Ps Service** from **Server**
Request **Cafe Service** from **Server**
Request **Login** from **Authenticator**
Request **Wifi Password** from **Server**
Request cancellation of pending request from **Server**
Check Transaction history
Check Credits
Collaboration
Server
Authenticator

Second Class

Sysadmin CLASS
Responsibility
Create new **User** account and send to **Server**
Send messages to manage **User** account changes to **Server**
Create new **IT** account and send to **Server**
Send messages to manage **IT** account changes to **Server**
Create new **Cafe** account and send to **Server**
Send messages to manage **Cafe** account changes to **Server**
Create new **Sysadmin** account and send to **Server**
Send messages to manage **Sysadmin** account changes to **Server**
Send message to blacklist **User** account
Send change signal to **User** to change **User** credits
Request **Login** from **Authenticator**
Collaboration
Server
User
Authenticator

Third Class

ItWorker CLASS
Responsibility
Send **create_new_Pc_service** signal to **Server**
Send **remove_new_Pc_service** signal to **Server**
Send **create_new_Ps_service** signal to **Server**
Send **remove_new_Ps_service** signal to **Server**
Send **change_wifi_password** signal to **Wifi Password**
Approve pending **It Transactions** from **Server**
Reject pending **It Transaction** from **Server**
Send **create_new_It_service** signal to **Server**
Send **remove_new_It_service** signal to **Server**
Request **Login** from **Authenticator**
Collaboration
Server
Wifi Password
Authenticator
It Service
Pc Service
Ps Service
Transaction

Fourth Class

CafeWorker CLASS
Responsibility
Send **create_new_Cafe_Item_service** signal to **Server** Send **remove_new_Cafe_Item_service** signal to **Server** Approve pending **Cafe Transaction** from **Server** Reject pending **Cafe Transaction** from **Server** Request **Login** from **Authenticator** Change is Available for **Cafe items**
Collaboration
Server **Authenticator** **Cafe Item Service**

Fifth Class

Transaction CLASS
Responsibility
Create new **Transaction** for **User** Route pending **Transaction** to **Server** Encapsulate Service
Collaboration
Server **User** **Cafe Item Sevice** **It Service** **Pc Service** **Ps Service**

Sixth Class

Machine CLASS
Responsibility
Create new machine for **PC service** upon receiving a signal from **Server** Create new machine for **PS service** upon receiving a signal from **Server**
Collaboration
PC service **PS service**

Seventh Class

Cafe Item Service CLASS
Responsibility
Provide Service base to **Transaction**
Collaboration
Transaction

Eighth Class

IT Support Service CLASS
Responsibility
Provide Service base to **Transaction**
Collaboration
Transaction

Ninth Class

PC service CLASS
Responsibility
Provide Service base to **Transaction**
Collaboration
Transaction

Tenth Class

PS Service CLASS
Responsibility
Provide Service base to **Transaction**
Collaboration
Transaction

Eleventh Class

Server CLASS
Responsibility
Route **User** account to **Authenticator**
Route **IT worker** account to **Authenticator**
Route **Sysadmin** account to **Authenticator**
Route **Cafeworker** account to **Authenticator**
Save newly created **User** accounts sent by **Sysadmin**
Save newly created **IT** accounts sent by **Sysadmin**
Save newly created **Cafe** accounts sent by **Sysadmin**
Save newly created **Sysadmin** accounts sent by **Sysadmin**
Route pending **Cafe Item Transaction** to **Cafeworker**
Route pending **PC Service Transaction** to **IT worker**
Route pending **PS Service Transaction** to **IT worker**
Route pending **It Support Transaction** to **IT worker**
Route **PC Service** to **User**
Route **Cafe Item Service** to **User**
Route **PS Service** to **User**
Route **IT Support Service** to **User**
Collaboration
Itworker
Cafeworker
User
Sysadmin
PC Service
PS Service
Cafe Service
IT Support Service

Twelfth Class

Wif password CLASS
Responsibility
Show wifi password to **User**
Show wifi password to **IT worker**
Change the wifi password
Collaboration
Itworker
User

Thirteenth Class

Authenticator CLASS
Responsibility
Get authenticated accounts from **Server**
Authenticate **User**
Authenticate **Itworker**
Authenticate **Sysadmin**
Authenticate **Cafeworker**
Collaboration
Itworker
User
Cafeworker
Sysadmin
Server

1 Noun Extraction

User Registration:
Users request account creation with national ID, email, and password at the front desk. Admins process requests for account creation.
Prepaid Credits:
Users prepay at the front desk to add money, converted to credits for usage. Admins update accounts with relevant credits.
Credit Usage:
Credits are deducted when users access PCs or PlayStations based on predefined rates. Credits are also deducted on requests for WiFi password, cafe service, or tech support. Credits are reimbursed on cancellation of a pending request.
Support Requests:
Users request technical support, WiFi password, and order snacks/drinks. Support requests are pending until approved by designated cafe or IT workers. Snack service requests and IT support requests are pending until approved. All service requests deduct credits. Cafe and tech support, as well as WiFi requests, result in a one-time credit deduction. PC and PlayStation usage leads to recurring credit deduction per minute. If credits become less than or equal to 0 during usage, PC or PlayStation services end. If credits are less than 0, users can't make service requests.
Pause Time:
Users can pause time, locking PCs while credits continue to be deducted.
Account Management (Admin):
Admins create/delete user accounts, manage details, and view transaction history.
Blacklisting (Admin):
Admins blacklist users to restrict access.
System Control (Admin):
Admins control PC/PS availability.
Credit Management (Admin):
Admins add/remove credits from user accounts.
System Structure:
Frontend: User-friendly mobile app with Android Studio. Backend: Java simulation on RAM using OOP principles (No Database).
Storage:
User Data: Stored in a pseudo-database class, structure to be determined. Transactions: Stored for each user.
Authentication:
Authenticator Class takes username and password and compares them against server.

Newly Extracted Nouns

- **User Registration**
  - Account Creation
  - National ID
  - Email
  - Password
- **Prepaid Credits**
  - Money
  - Credits
  - Usage
- **Credit Usage**
  - Credits
  - Users
  - PCs
  - PlayStations
  - WiFi Password
  - Cafe Service
  - Tech Support
  - Support Requests
  - Snacks/Drinks
  - Service Requests
  - Cafe
  - WiFi Requests
  - Credit Deduction
  - PC and PlayStation Usage
- **Support Requests**
  - Users
  - Technical Support
  - WiFi Password
  - Snacks/Drinks
  - Support Requests
  - Snack Service Requests
  - IT Support Requests
  - Service Requests
  - Cafe
  - Tech Support
  - WiFi Requests
  - Credit Deduction
  - PC and PlayStation Usage
- **Pause Time**
  - Users
  - Time
  - PCs
  - Credits
- **Account Management (Admin)**
  - Create/Delete User Accounts

  - Manage Details
  - Transaction History
- **Blacklisting (Admin)**
  - Blacklist Users
  - Access
- **System Control (Admin)**
  - Control PC/PS Availability
- **Credit Management (Admin)**
  - Add/Remove Credits from User Accounts
- **System Architecture**
  - Frontend
  - User-Friendly Mobile App
  - Android Studio
  - Backend
  - Java Simulation on RAM
  - OOP Principles
  - Database
- **Storage**
  - User Data
  - Pseudo-Database Class
  - Structure
  - Transactions
- **Authentication**
  - Authenticator Class
  - Username and Password
  - Server

8 OOAD Methodologies

The other OOAD methodology that was used is OMT, short for object modelling techniques, the reason why OMT is used is because it helps the stakeholders have a comprehensive understanding of the the project by using diagrams. In addition to this, there is data flow in the system, and DFDs are a good way to represent the data flows.

In OMT, there are four phase, firstly, the analysis phase, where the system is modelled using class,state and DFD diagrams(shown in figure 5 below). Each corresponding to the object, dynamic and functional models respectively.

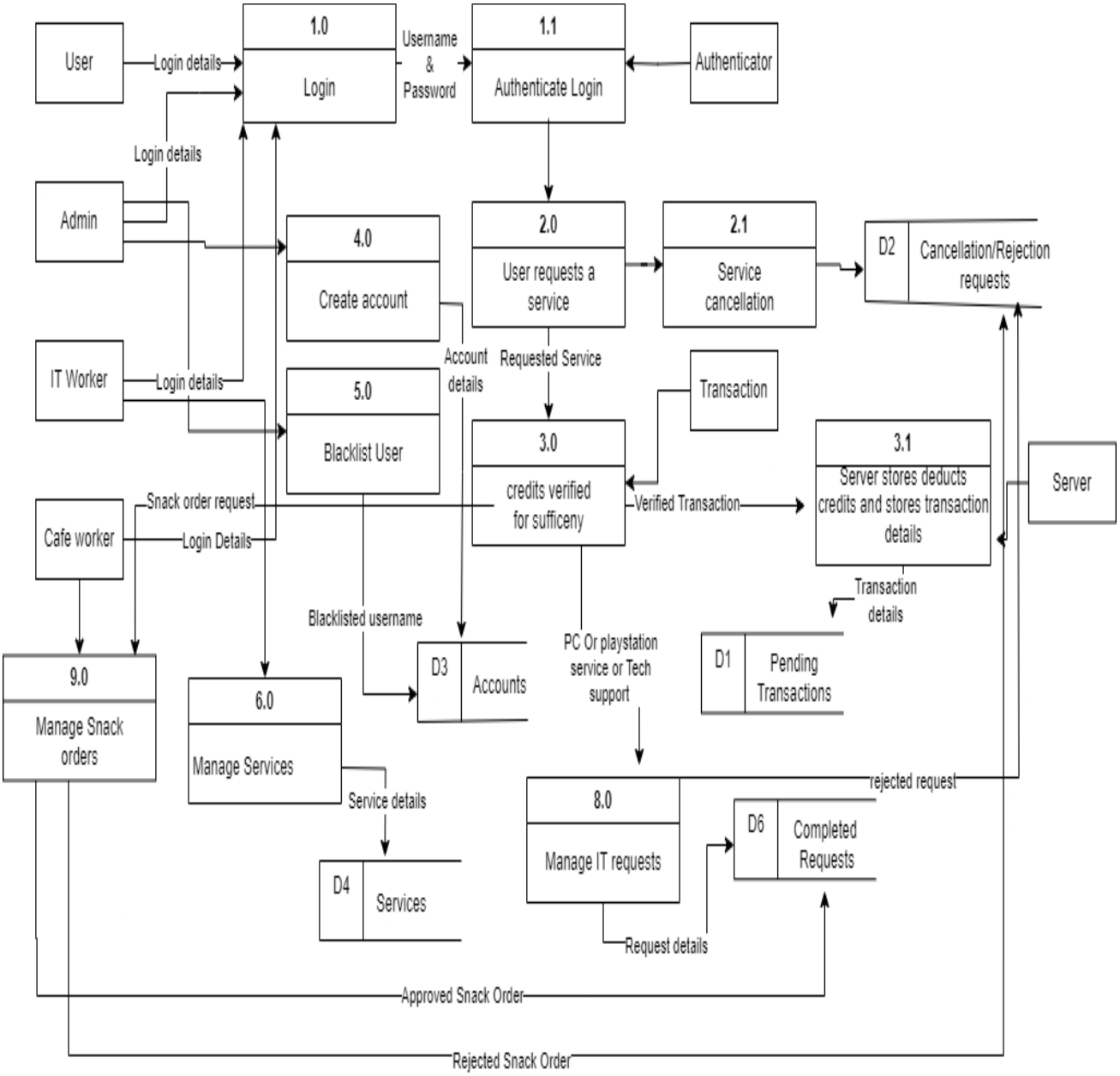


Figure 5 DFD diagram of GGS

The 2nd phase is system design, where the system is broken down into sub systems(see component diagram for more details). Those subsystems allows the stakeholders to see how different systems interact with each. This helps to strengthen the understanding of the systems involved in the project, hence more stakeholders would probably have a better idea of whether the developers are going along with the requirements correctly or not.

The 3rd phase is object design, by using state and class diagrams, the objects and their interactions can be shown, their classes and the relations between their classes and their multiplicity and also shown on the class diagram. While the state diagram shows the dynamic model of the system, this gives stakeholders a better idea of how the system works.

The 4th phase is Implementation, a prototype is shown in the report to give the stakeholder an idea of how the program looks like. For example, the prototype used here is the GUI of the system, this would give stakeholders a better idea of how the app would look like, they could also suggest changes, this would probably reduce the cost and time taken to develop the software as unwanted features are more likely to be eliminated.

The CRC cards help provide the static model of the system, stating the responsibilities of the actors and the collaboration of them in each class in the system.

This helps the stakeholders understand the impact of each actor in the system and whether the clients would want to modify the permissions of each user to suit their needs.

Noun extraction is one way to clarify the actions of actors. E.g. Admins control PS/PC availability.

To summarize, Object Modeling Techniques (OMT) is a methodology in Object-Oriented Analysis and Design (OOAD). OMT navigates through four distinctive phases—analysis, system design, object design, and implementation—leveraging class, state, and Data Flow Diagrams (DFDs) to encapsulate static, dynamic, and functional aspects. Noun extraction is a pivotal process in OOAD that involves identifying and extracting nouns.

This strategic extraction of nouns ensures that the resulting object model accurately mirrors the real-world entities and relationships, forming a solid foundation for system development. Furthermore, Class-Responsibility-Collaboration (CRC) cards, a technique often employed in with OOAD, play an important role in stating the responsibilities and collaborations between different classes, fostering a clear understanding of the system's behavior and relationships. Together, OMT, noun extraction, and CRC are very good ways to provide the stakeholders with a better understanding of the system.

9 State Diagram

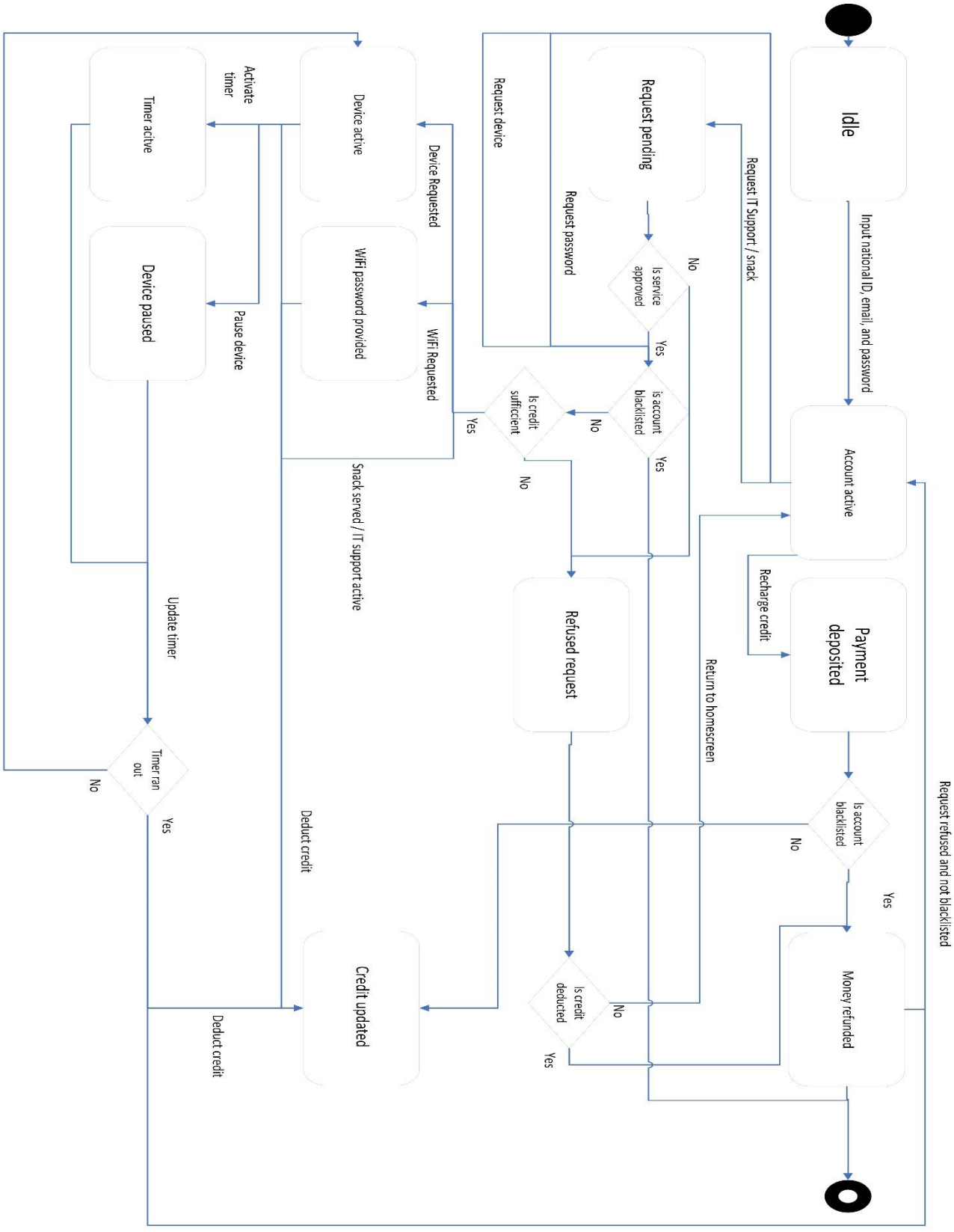


Figure 6 State Diagram of GGS

10 Client-Object Relation Diagram

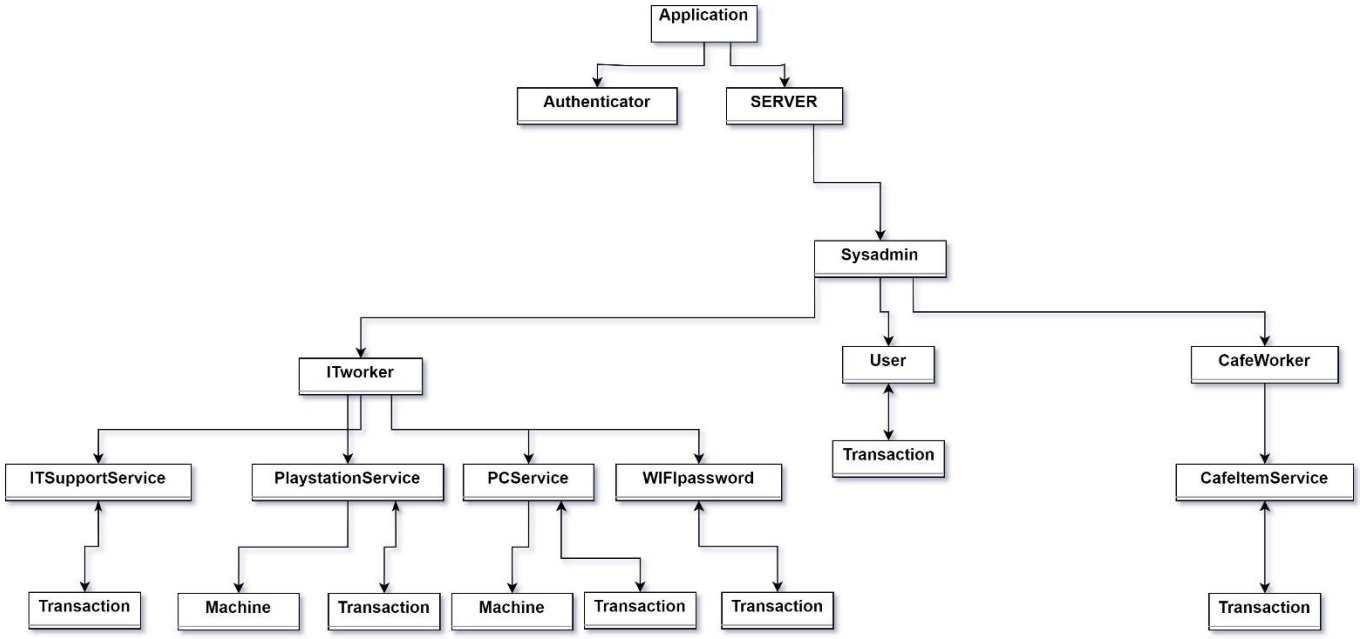
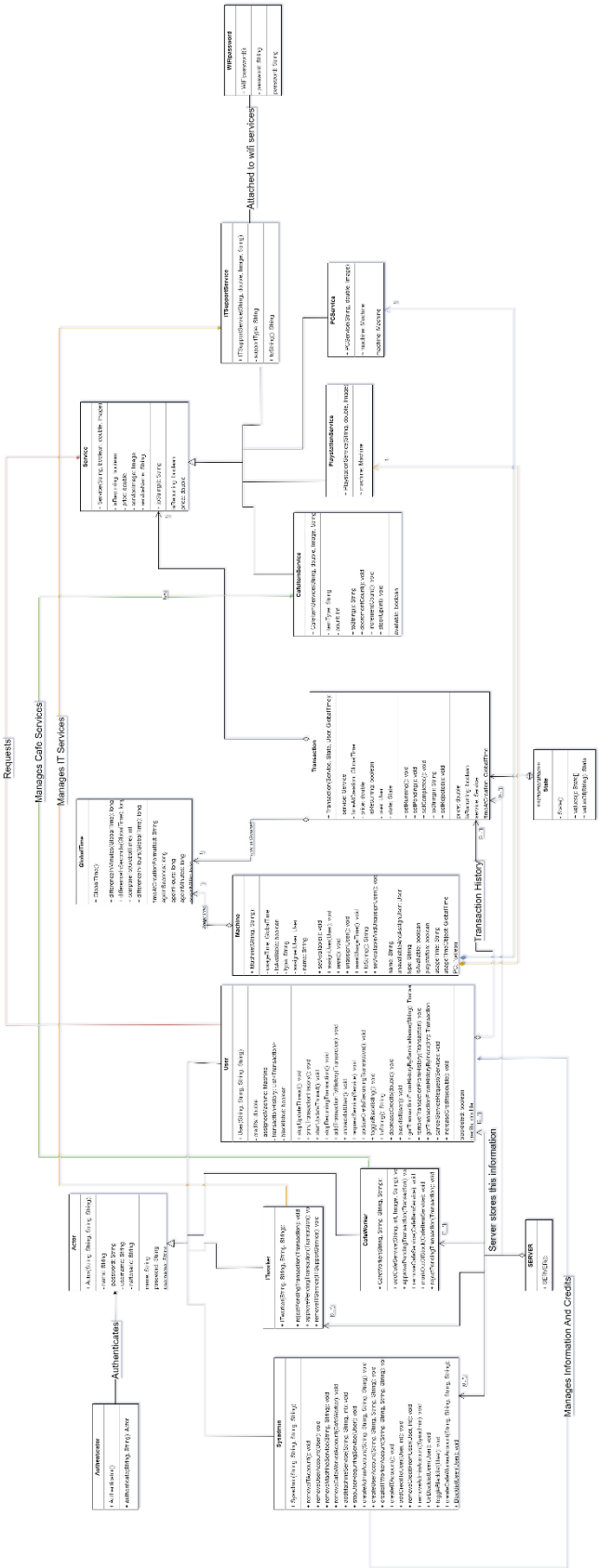


Figure 7 Client-Object Relation diagram of GGS

11 Class diagram

(Figure 8 below shows the class diagram of GGS)



12 Comparative Analysis of the Output of the Adopted Methodologies

Noun Extraction Methodology

Analysis Phase

Noun extraction methodology involves identifying and extracting key nouns from the problem statement or domain to understand the entities and relationships. In the analysis phase, this method focuses on creating a model based on the extracted nouns, which helps in establishing a foundational understanding of the system.

Design Phase

In the design phase, the noun extraction method aids in the creation of class diagrams and object-oriented models. The extracted nouns form the basis for defining classes and their attributes, facilitating the transition from analysis to design. This method provides a clear representation of the static structure of the system.

CRC Methodology

Analysis Phase

CRC methodology involves the identification of classes, their responsibilities, and their collaborations. In the analysis phase, this method encourages stakeholders to define classes and allocate responsibilities to them. Collaboration cards are used to capture how classes interact, which helps the stakeholders understand the dynamic actions in the system.

Design Phase

During the design phase, CRC cards help in identifying the responsibilities of each class and further detailing the collaborations between them. This method helps in building a dynamic model, providing a view of the interactions and responsibilities of classes.

OMT (Object Modeling Technique) Methodology

Analysis Phase

The OMT has three models, the Object model, modelled by a class diagram, Functional modelled using DFDs, and dynamic model is modelled using a state diagram.

Design Phase

In the design phase, OMT extends its modeling capabilities to class diagrams, sequence diagrams, and collaboration diagrams. It provides a detailed view of the system.

Comparative Analysis of Benefits

Analysis Phase

OMT provides the most level of analysis, covering both static and dynamic aspects. CRC shows interactions between classes via collaborators , while noun extraction focuses mostly on static structures.

Noun extraction is simple, so stakeholders with low technical background might grasp a better idea of the system.

In CRC, stakeholders engage with each other during the analysis phase, which helps to improve clarity among the stakeholders

Design Phase

OMT and CRC ensure a smooth transition from analysis to design, with OMT providing a more integrated approach. Noun extraction serves as a solid foundation for the design phase.

OMT and CRC offer more detailed modeling capabilities, e.g. (OMT has 3 types of modelling), this allows for a better representation of the system. Noun extraction, while simpler, may be preferred for smaller projects or when simplicity is a priority.

13 Architectural Model

The GameGuard.system follows the Model-View-Controller (MVC) architectural pattern to systematically structure its components,

In the context of GameGuard.system, which manages Internet Cafe related functionalities,

MVC proves to be a fitting choice for handling various aspects of user interactions and system operations.

Model

Within this context, the server class assumes a pivotal role as a pseudo-database.

While currently acting as a stand-in for a Model, it serves as a prototype layer that can later be swapped to a more traditional database.

This class functions as a data repository, which allows querying and modification of information shared across various classes.

View

The system's user interface is visually designed using fragments and activities within the Android Studio layout editor using XML.

Specifically, views are crafted with Android Jetpack Compose.

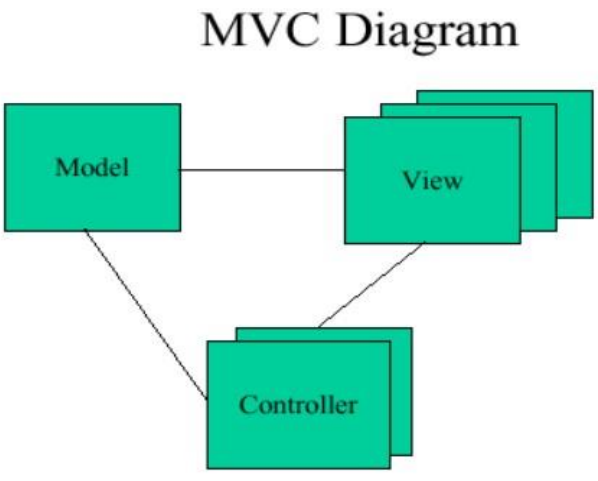


Figure 9 MVC Diagram

Controller

Classes like User, Sysadmin, Itworker, and CafeWorker ...etc. act as controllers, handling user input and implementing the system's logic.

For example, the Sysadmin class manages the creation and modification of user accounts, interacting with the Server to execute user-related tasks.

Itworker and CafeWorker classes act as controllers for IT and cafe-related functions.

The decision to implement the MVC architecture in GameGuard.system is justified by its inherent advantages in promoting modularity and separation of concerns.

By adhering to MVC principles, GameGuard.system ensures that each component has a distinct role, facilitating independent development and maintenance.

This modular design enhances scalability, making it easier to extend and modify the system as gaming requirements evolve during it's development cycle.

Overall, the MVC pattern contributes to the robustness and flexibility of GameGuard.system in managing gaming-related activities.

14 Component Diagram

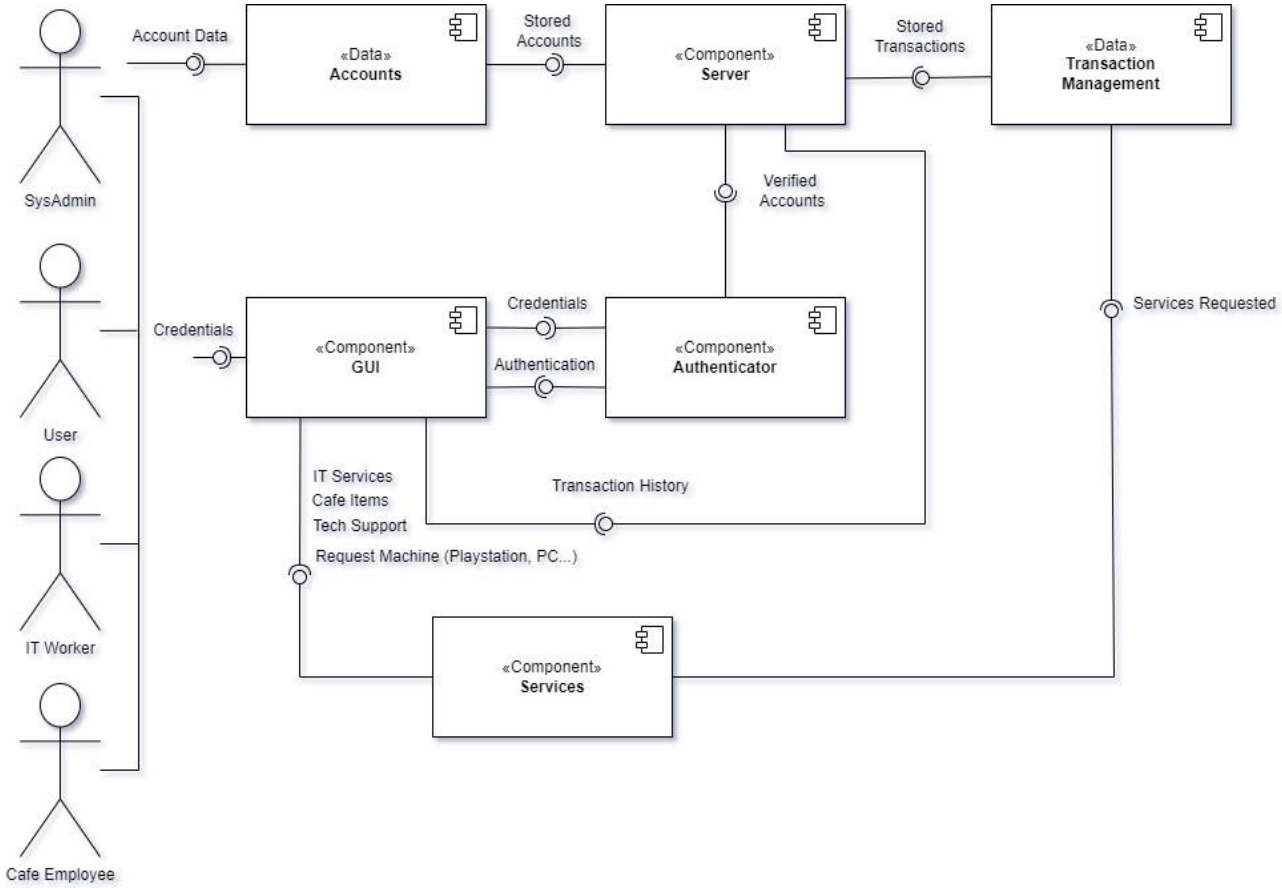


Figure 10 Component Diagram of GGS

15 Timeline and Plan

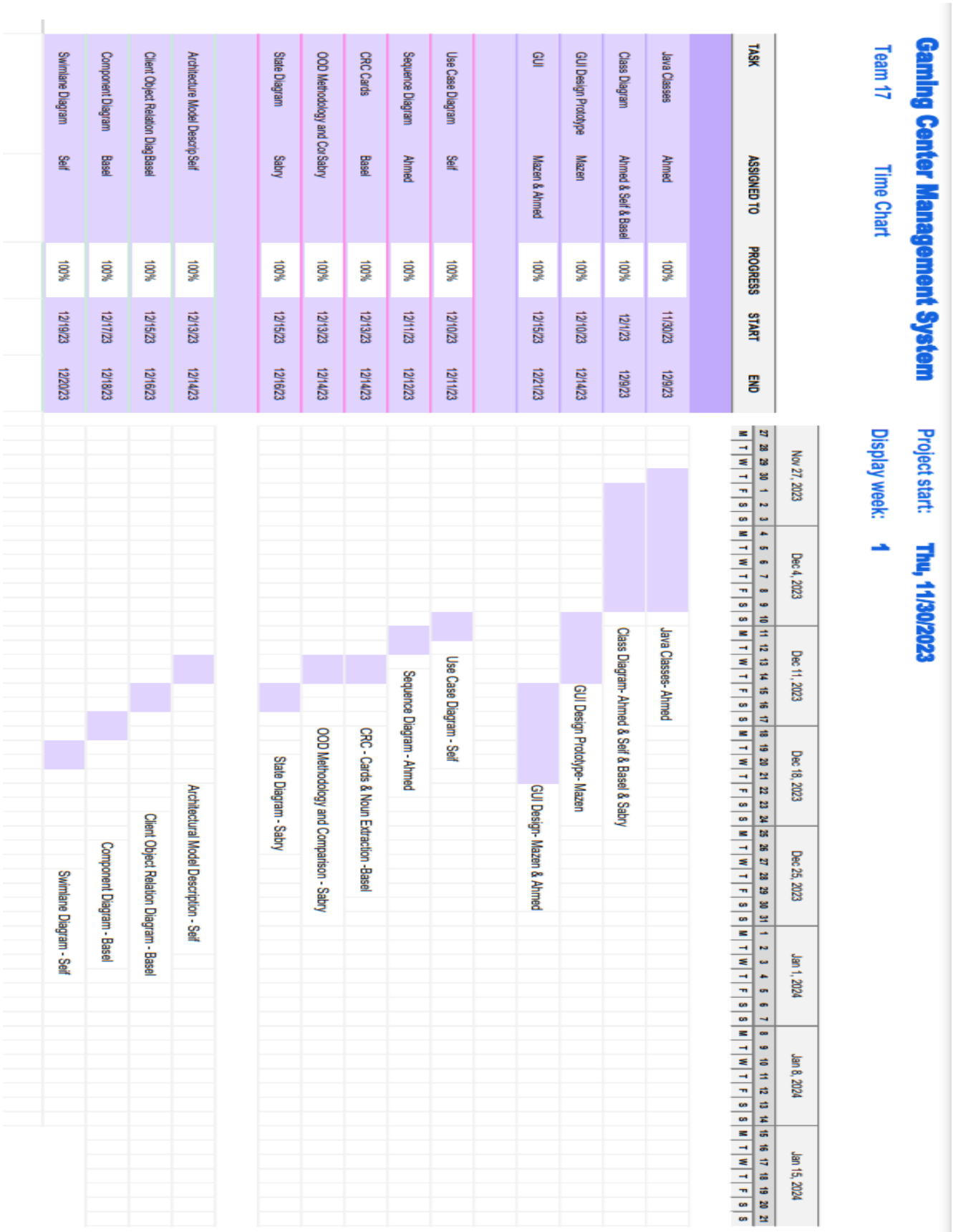
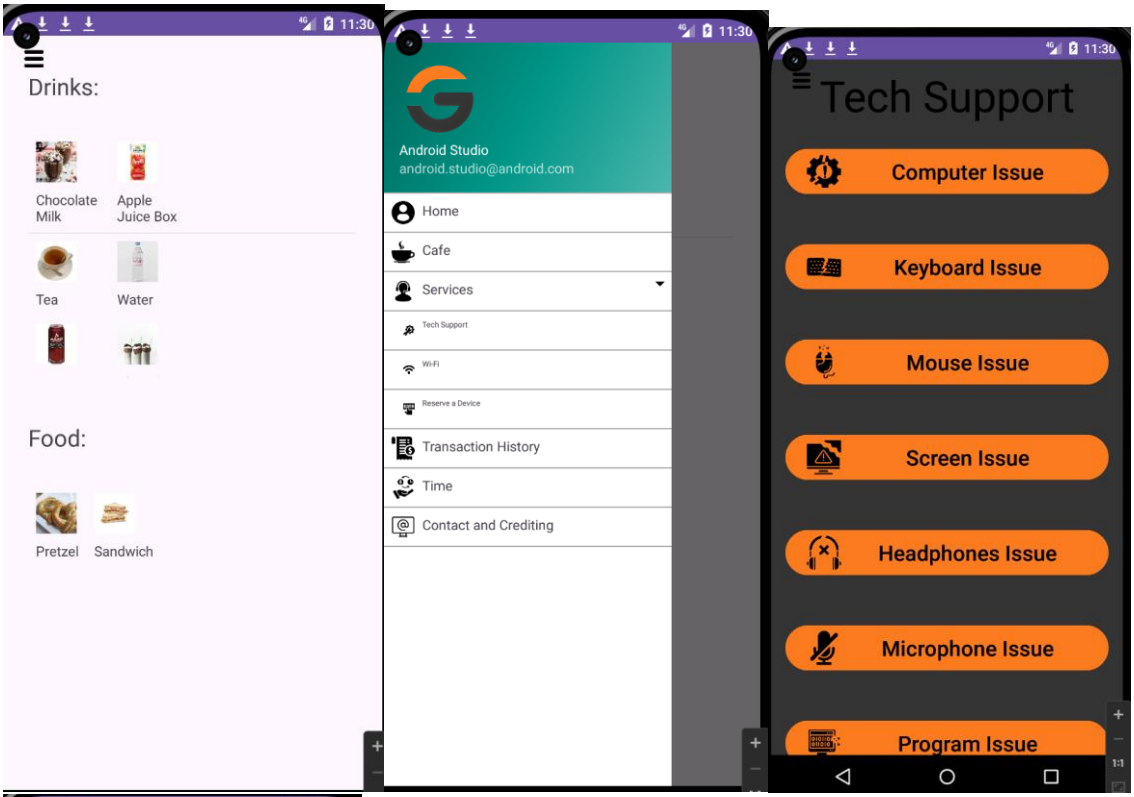
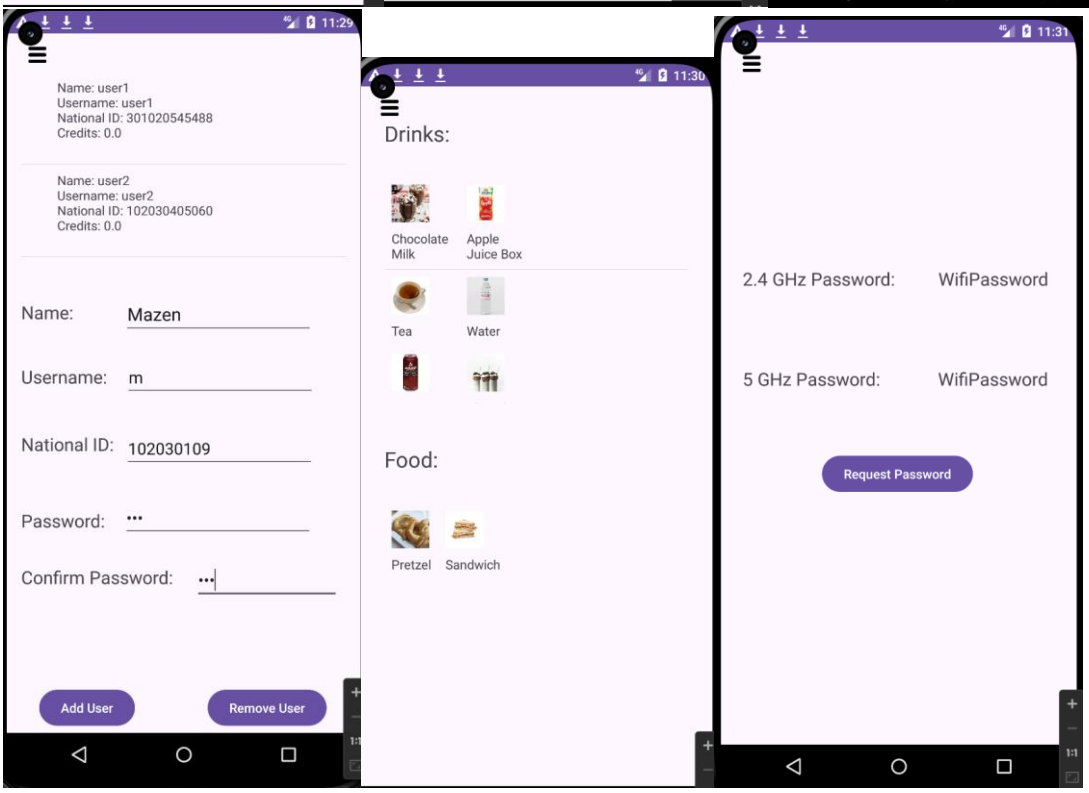


Figure 11 Timeline and Plan

16 Prototype





17 References

Lecture 1 (Advanced Software Engineering)

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture1%20-ObjectOrientedConcepts.pptx?forcedownload=1

Lecture 2 and 3 (UML)

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture2%20-%20UML-Part%201.pptx?forcedownload=1

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture3%20-%20UML-Part%202.pptx?forcedownload=1

Lecture 4 and 5 (OOAD, Noun extraction and CRC)

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture4-ObjectOrientedAnalysisAndDesign-Partt1.pptx?forcedownload=1

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture5-ObjectOrientedAnalysisAndDesign-Partt2.pptx?forcedownload=1

Lecture 8 (Architectural models)

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture8%20-%20ArchitectureModels.pptx?forcedownload=1

Lecture 9 (Software Prototyping)

https://lms.eng.asu.edu.eg/pluginfile.php/538635/mod_folder/content/0/CSE232_Lecture9%20-%20SoftwarePrototyping.pptx?forcedownload=1

18 Appendices