DAWN
Software Architecture Document

Version 1.0

By:
Dawn Group
2019-04

Group Member:
Zihan Xu
Yi Kuang
Chenyu Yang
Yuting Lan
Jianzhen Cao

Document Language:
English

Revision History

Date	Version	Description	Author
2019-4-16	1.0	Finish the 1st edition of Software Architecture Document	Zihan Xu, Yi Kuang, Chenyu Yang, Yuting Lan, Jianzhen Cao

SOFTWARE ARCHITECTURE DOCUMENT

1 Introduction

1.1 Purpose for This Document.

This is a software architecture document. This document describes the system structure of the software. It is mainly expressed in various models in the design model. However, not all models in the design model are included in this document. This document generally covers use cases that have a significant impact on the architecture and introduces related models around them. No need to mention that this document should not only give the model diagram, but also provide a description of the reasons for making the appropriate design decisions.

1.2 Domain

This document is written for our game DAWN.
To our best knowledge, this document will not affect any other products.

1.3 Definition

All the glossaries are provided in the glossary document(词汇表_en.docx).

1.4 Reference

<<Object-Oriented Software Engineering Practice Guide-2>> Shanghai Jiao Tong University Press, 2016
<<Object-Oriented Software Engineering - Using UML, Patterns, and Java>> (3rd edition), Tsinghua University Press, 2011

1.5 Overview

This document includes four parts: Introduction, Current System Architecture, System

Architecture Design Objectives, and Recommended Software System Architecture. The current system section analyzes the current offline second-hand trading market and points out its shortcomings. The system architecture design goals are combined with software requirements to list the goals of the system design. The proposed software system architecture gives an explanation of the architecture and subsystems of the system, and displays the object design, hardware and software deployment, data management, software control, and boundary conditions of the system in a combination of textual representation and model diagram. The various parts of this document are closely related, complement each other and contrast, and present the software architecture of the system.

2 Existing Software Architecture

Unfortunately, we don't have an existing software architecture to refer to.

3 Software Architecture Design objectives

The design goals of the system architecture are as follows:

1) Availability: If the system is unavailable when players enter the game, it will cause negative influence. Therefore, the system needs to ensure high availability.
2) Security: Important personal information is retained in the database, so the security of the system must be guaranteed.
3) High performance: The system responds in real time during operation and has a large flow rate, so it requires high performance.
4) Scalability: The system does not require relatively high-standard hardware when the initial scale is small. However, consideration should be given to the expansion of the system as the number of users increases.

4 Proposed Software Architecture

4.1 Overview

1) Client/Server

The client is App on users' phone, including interaction with users and store of a little information in the game process. The server controls the entire game process and other use cases, such as registration, Login, joining room and so on.

The reason for choosing this mode is that this mode is mature, and the method of separating the client and the server conforms to the environment of App development, and also facilitates the concentration and management of data.

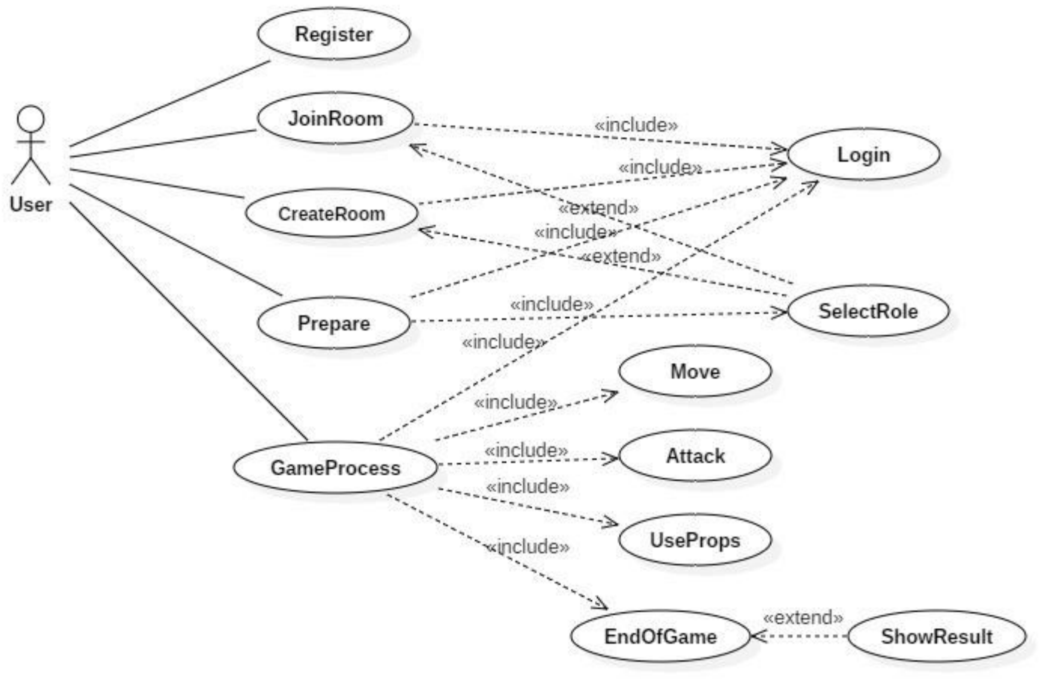
2) Model, View and Controller, MVC

The system organizes subsystems in three levels. The model includes a dataset to store users' account information. The view provides an interface (just UI). And the controller is to control and implement system functions, including UserManagement, RoomManagement, and RoleManagement & MapManagement during a game.

Most of this system is developed independently. A third-party library is used to support the implementation of interactive communication functions and animation display.

4.2 Use Case View

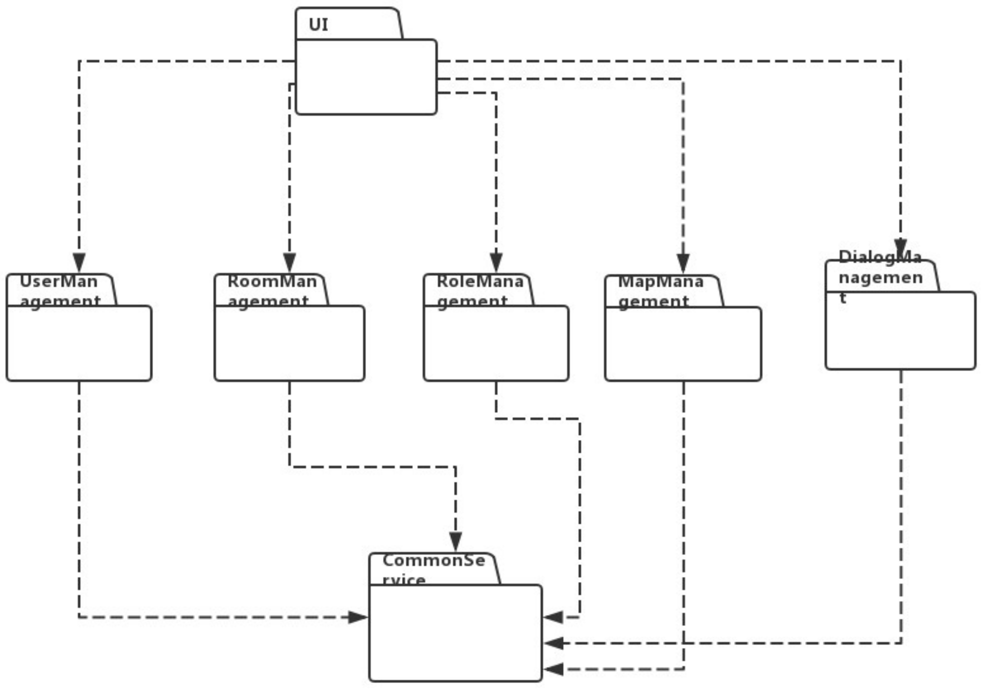
Here is the use case view:



4.3 Logical View

(1) System Architecture

We have six subsystems: User Interface Subsystem, User Management Subsystem, Room Management Subsystem, Role Management Subsystem, Map Management Subsystem, Common Service Subsystem



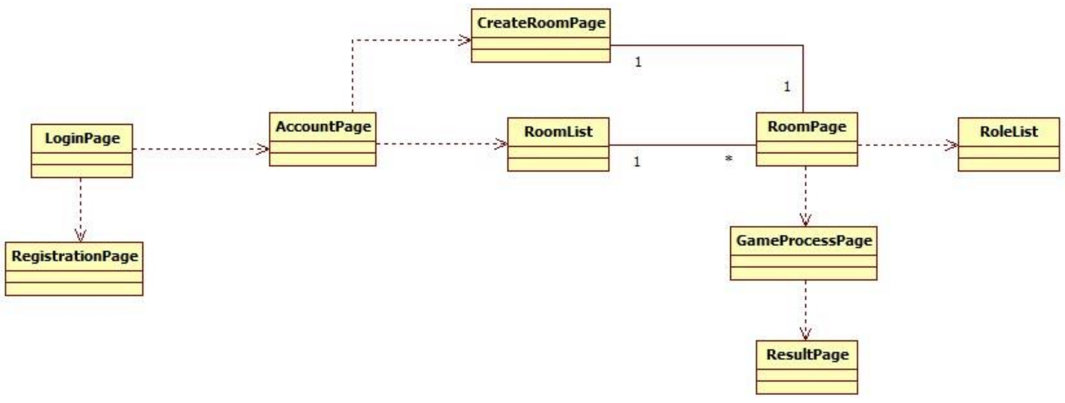
(2) Subsystem

1) User Interface Subsystem

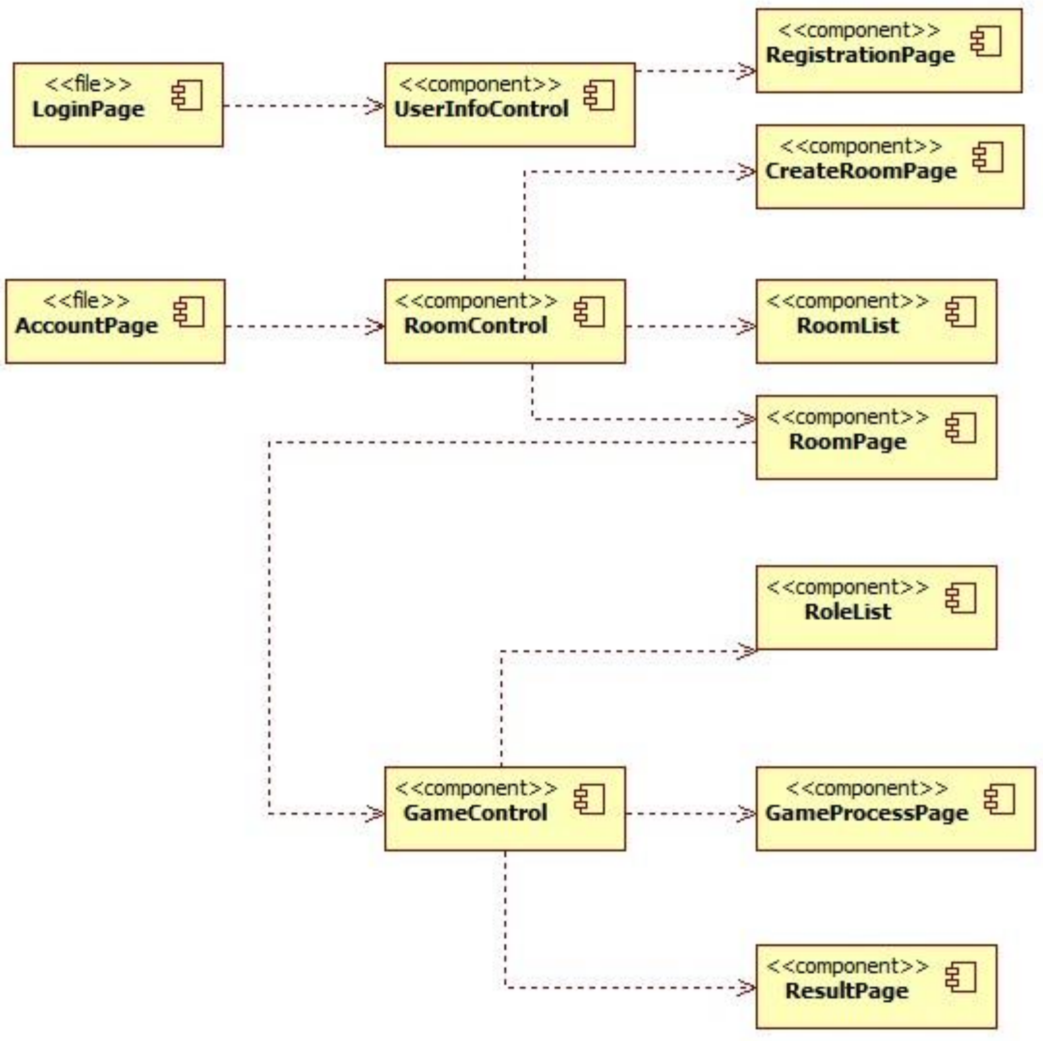
Function: Interact with user.

Service: The UI layer provides kinds of input boxes, buttons and text boxes to show information to user and interact with user.

Class Diagram:



Component Diagram:



<<file>> LoginPage
<<component>> UserInfoControl
<<component>> RegistrationPage
<<component>> CreateRoomPage
<<file>> AccountPage
<<component>> RoomControl
<<component>> RoomList
<<component>> RoomPage
<<component>> RoleList
<<component>> GameControl
<<component>> GameProcessPage
<<component>> ResultPage

2) User Management Subsystem

Function: Provide management service for user management, including registering, login and so on.

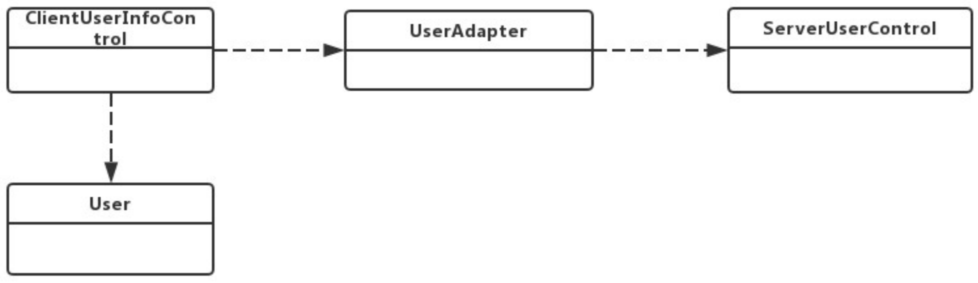
Service:

public Boolean summitRegister(list personalInfomation)

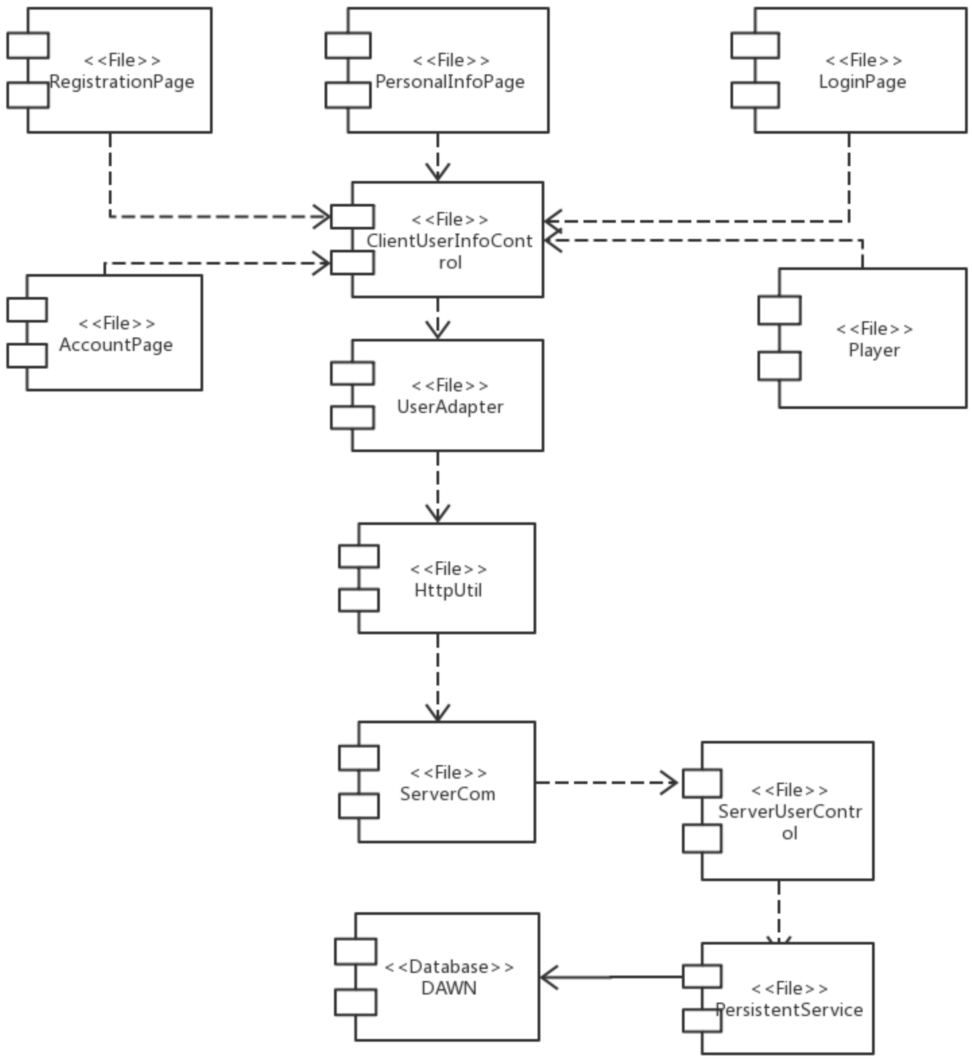
public Boolean submitLogin(String userID, String password)

public list showUserInformation(String UserID)

Class Diagram:



Component Diagram:


<<File>> RegistrationPage
<<File>> PersonalInfoPage
<<File>> LoginPage
<<File>> ClientUserInfoControl
<<File>> AccountPage
<<File>> Player
<<File>> UserAdapter
<<File>> HttpUtil
<<File>> ServerCom
<<File>> ServerUserControl
<<Database>> DAWN
<<File>> PersistentService

3) Room Management Subsystem

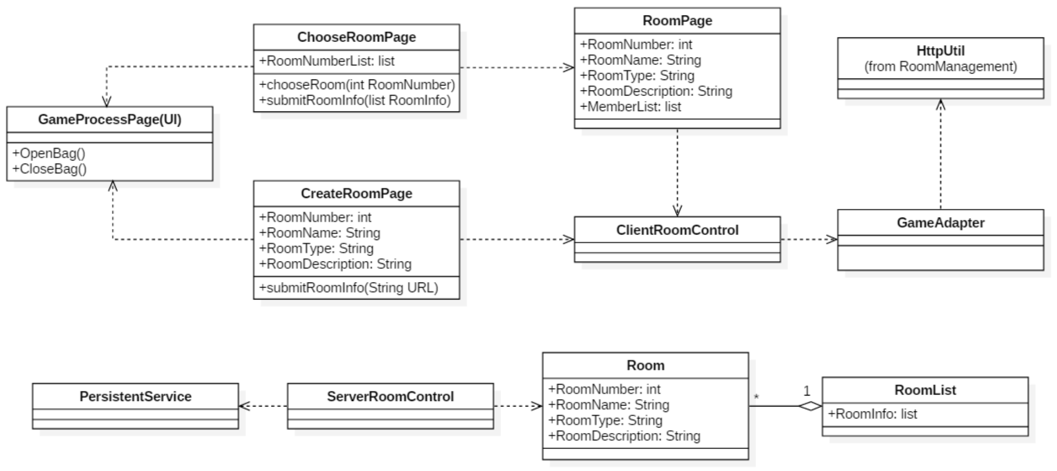
Function: Manage the creation of a room and choose the room to enter.

Service:

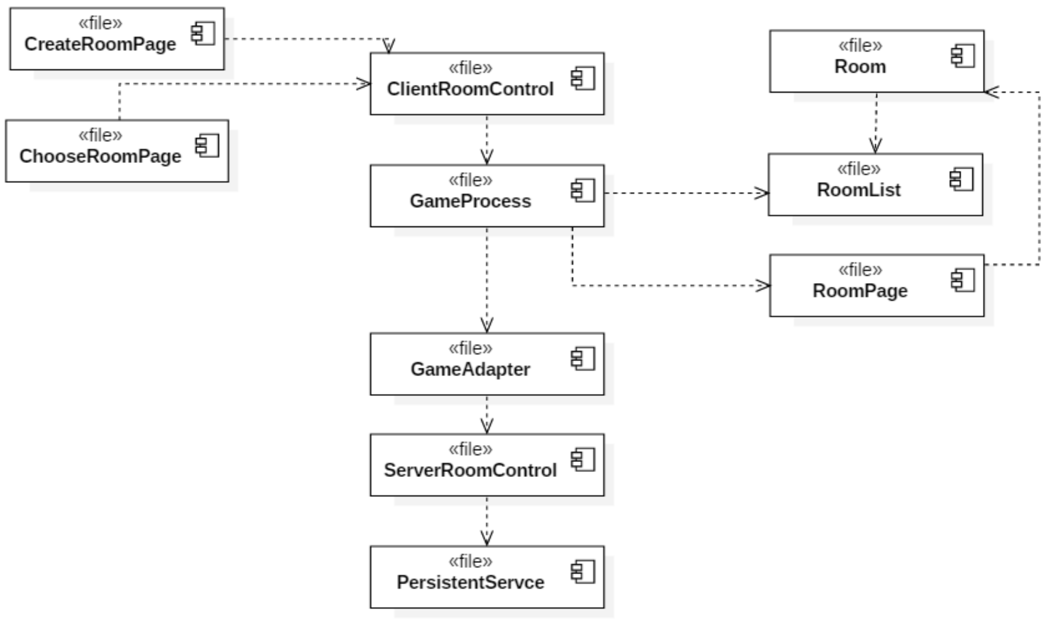
public Boolean chooseRoom(Room chosenRoom)

public Room createRoom(list roomInfomation)
public list requestRoomList(Boolean showAllRoom)
public Boolean joinRoom(Room chosenRoom)

Class Diagram:



Component Diagram:



4) Role Management Subsystem

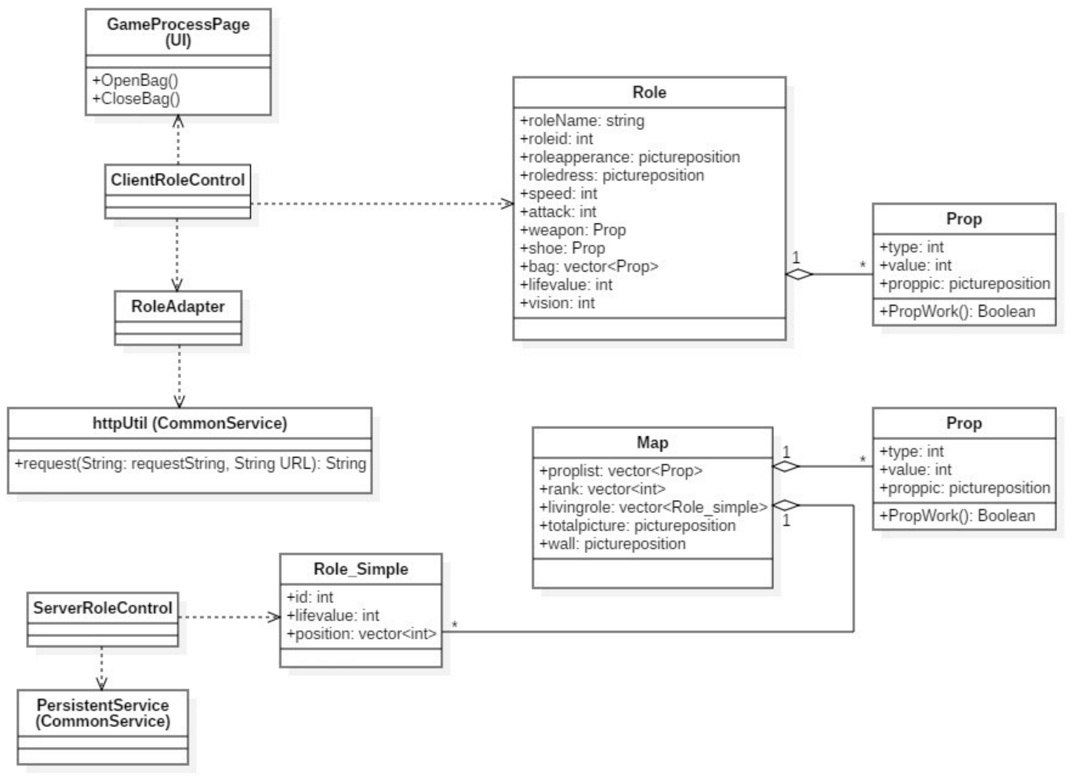
Function: Provide management service for role management, including creating and changing a role.

Service:

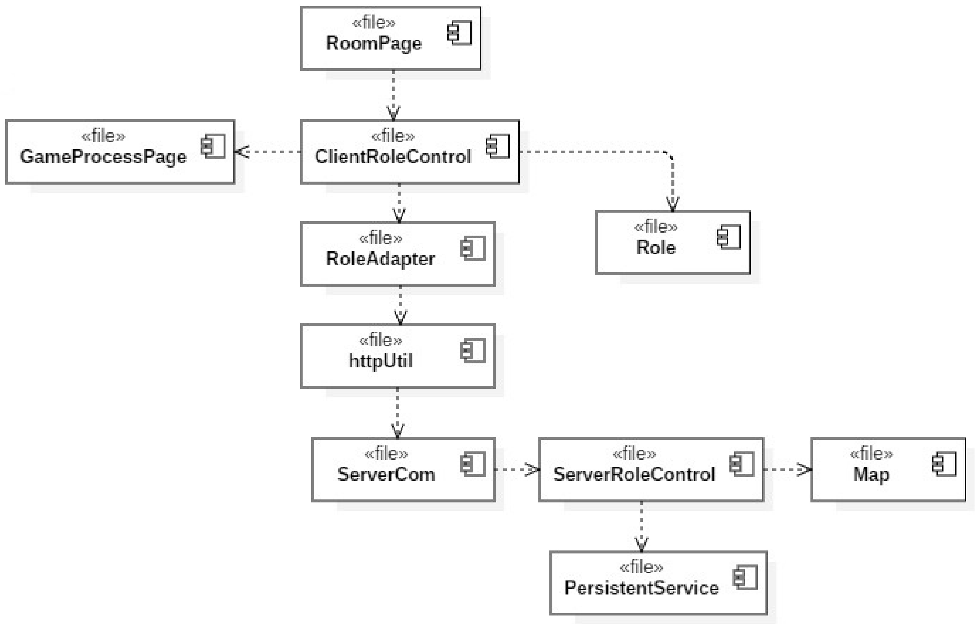
public Role RoleCreate(Player p)
public Boolean makeattack(int direction, int[] position)
public Boolean addprop(Prop p)

```
public Boolean useprop(Prop p)
public Boolean SubmitLifevalue(int ID, int value)
```

Class Diagram:



Component Diagram:



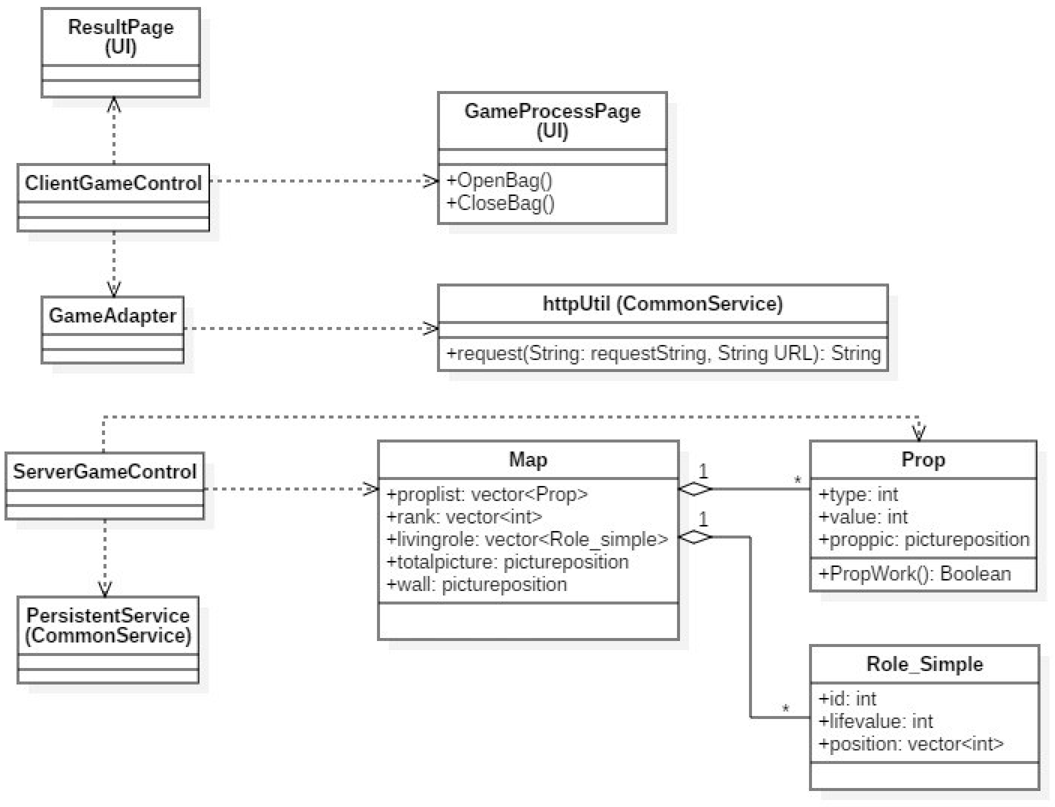
5) Map Management Subsystem

Function: Provide management service for maps, including creating and movements in the map.

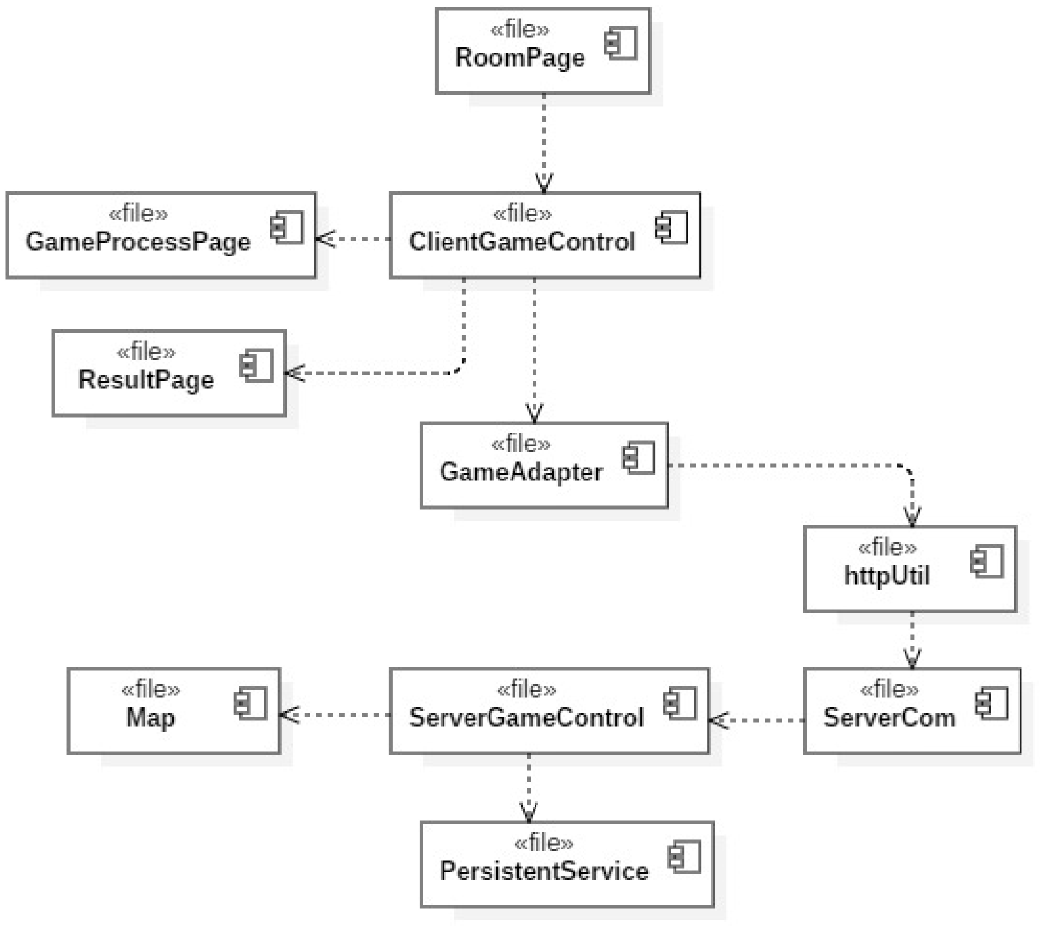
Service:

```
public Map MapCreate(Room r)
public void move(int direction, int[] position, int speed)
public Boolean pickup(Prop[] plist, int[] position)
public Boolean submitdamage(int targetID, int value)
public Map getmapdata()
public void InfoUpdate(Map m)
public void RankRecord()
public Boolean SendResult()
public void ShowResult()
```

Class Diagram:



Component Diagram:



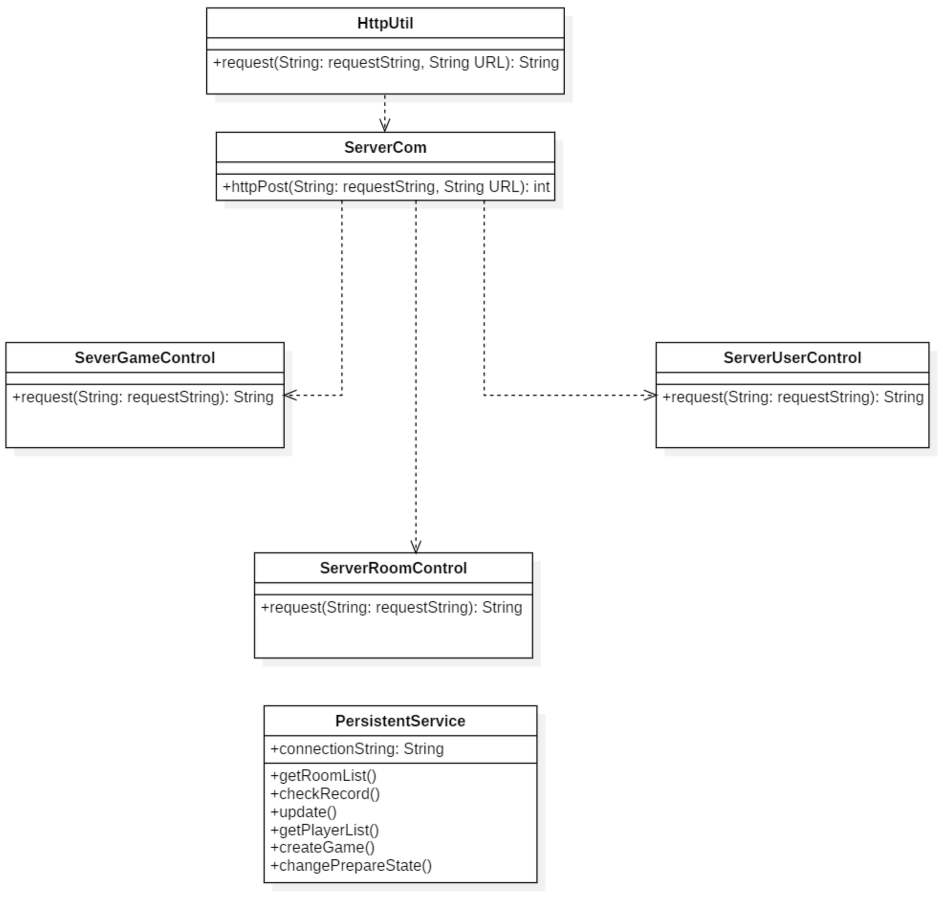
6) Common Service Subsystem

Function: Provide data access management, client and server communication functions, mainly room, game, user information.

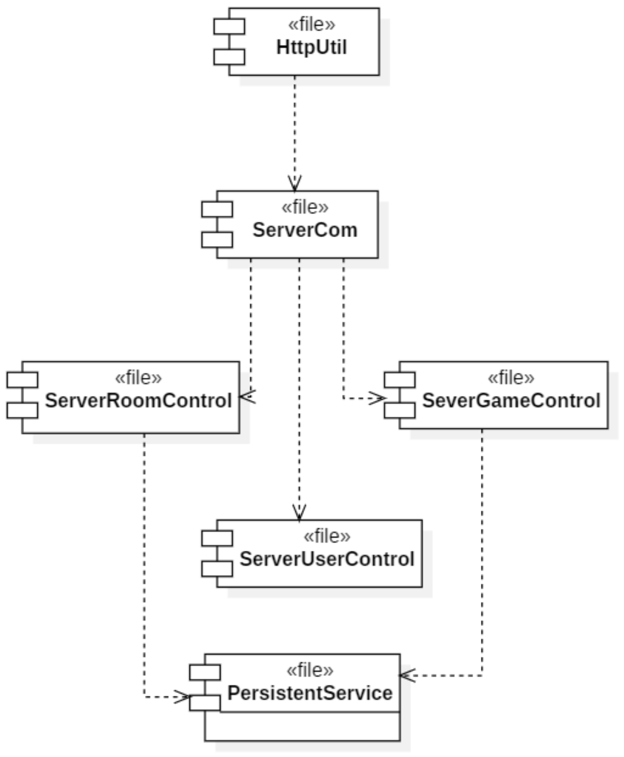
Service:

```
public String request (String requestString, String url)
public Boolean ChangeRoleState(int roomID, int roleID)
public Boolean ChangePrepareState(int roomID)
public Boolean UpdateUserInfo(int AccountID,String contents)
public Boolean ChangeGameState(int roomID,String contents)
```

Class Diagram:

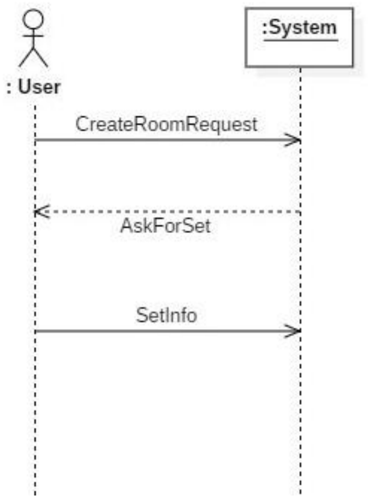

HttpUtil
+request(String: requestString, String URL): String
ServerCom
+httpPost(String: requestString, String URL): int
SeverGameControl
+request(String: requestString): String
ServerUserControl
+request(String: requestString): String
ServerRoomControl
+request(String: requestString): String
PersistentService
+connectionString: String
+getRoomList()
+checkRecord()
+update()
+getPlayerList()
+createGame()
+changePrepareState()

Component Diagram:

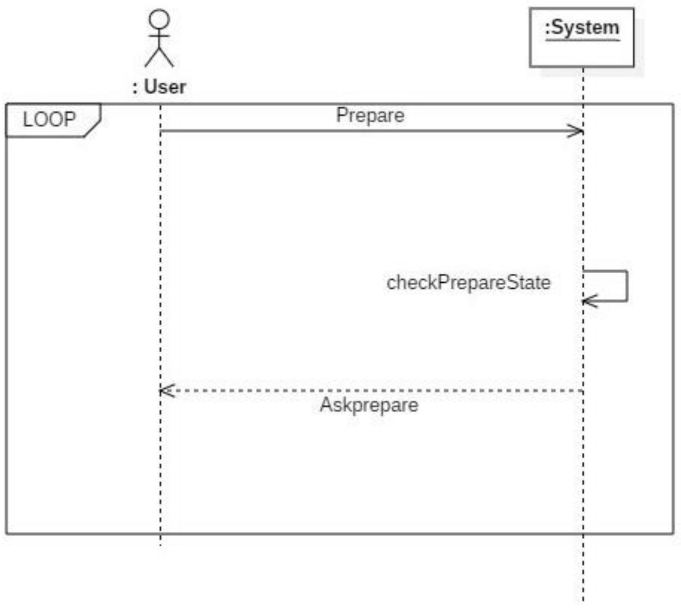


（3） Use Case Implementation

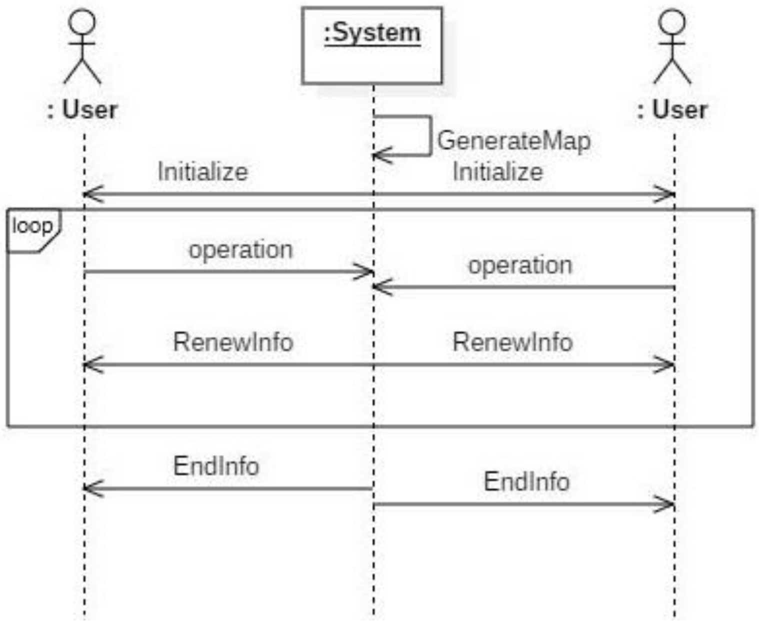
1) CreateRoom



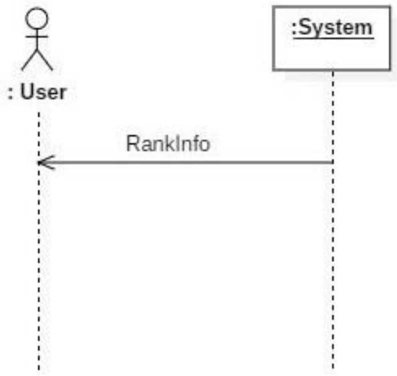
2) Prepare



3) GameProcess

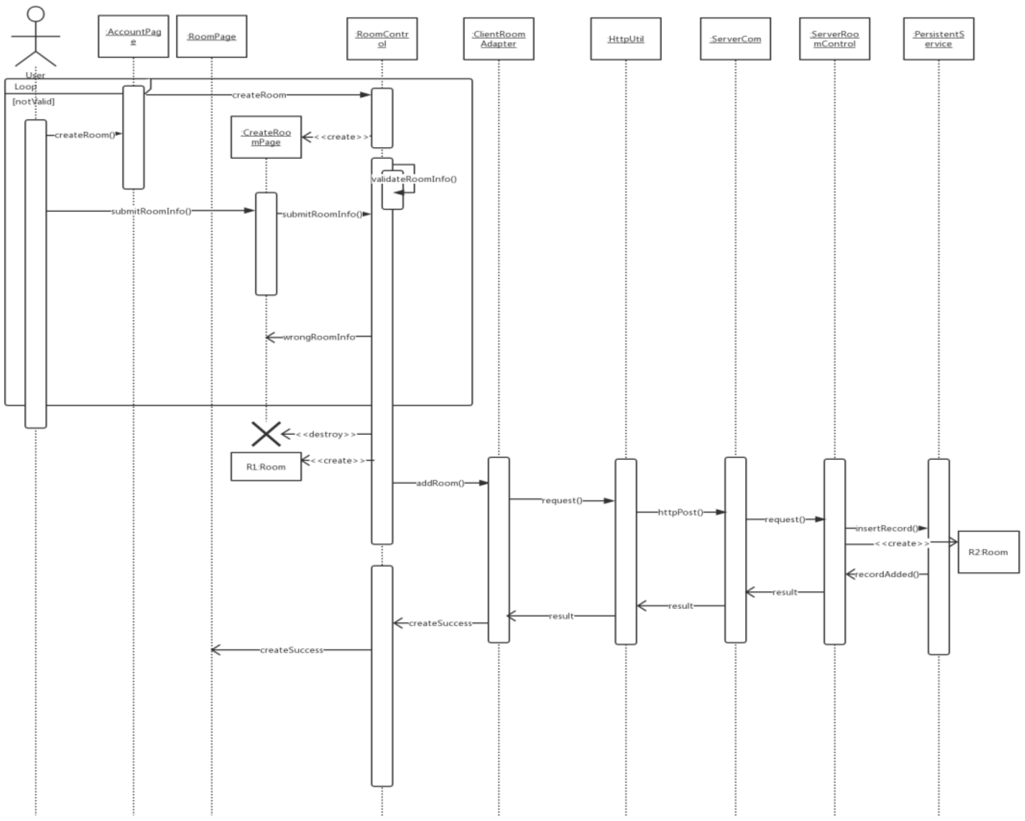


4) ShowResults

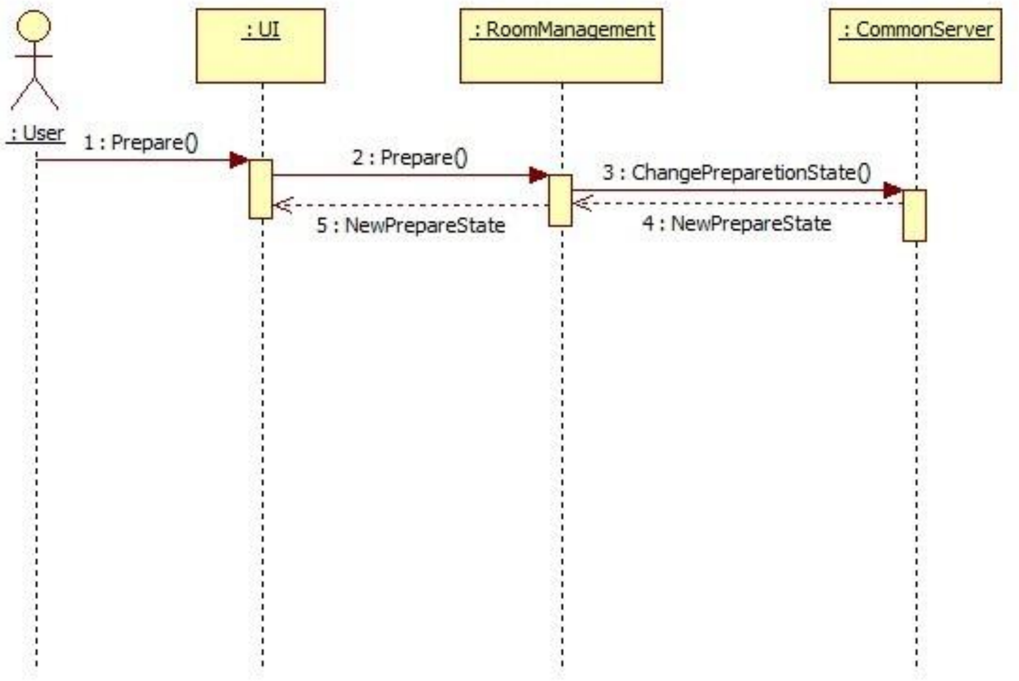


（4） Subsystem Collaboration

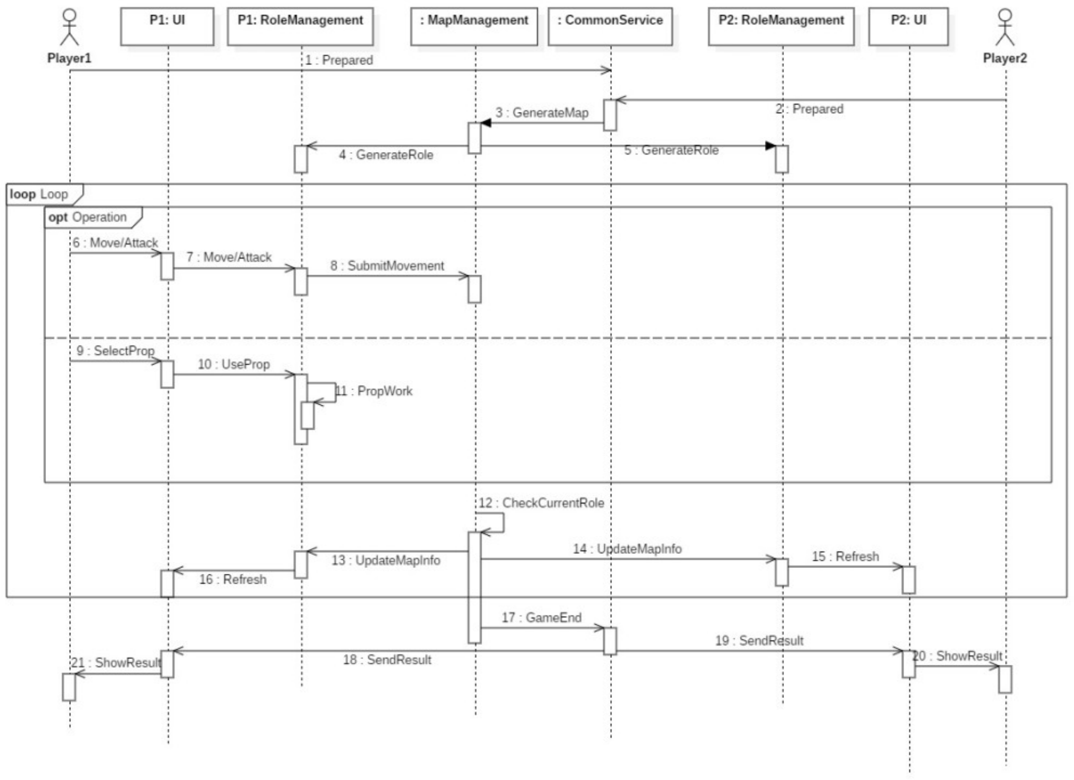
1) CreateRoom



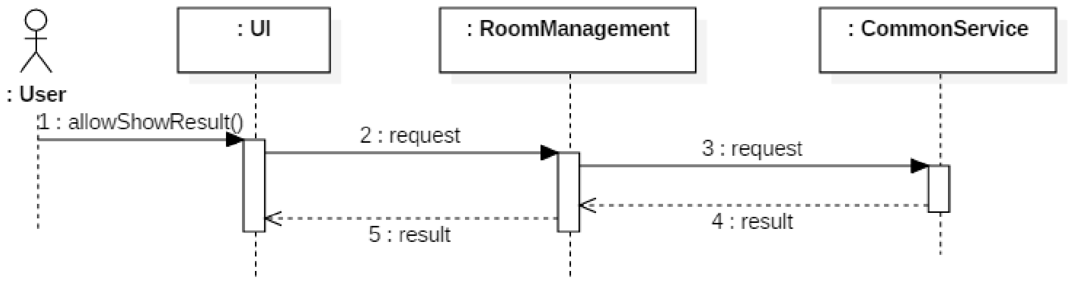
2) Prepare



3) GameProcess



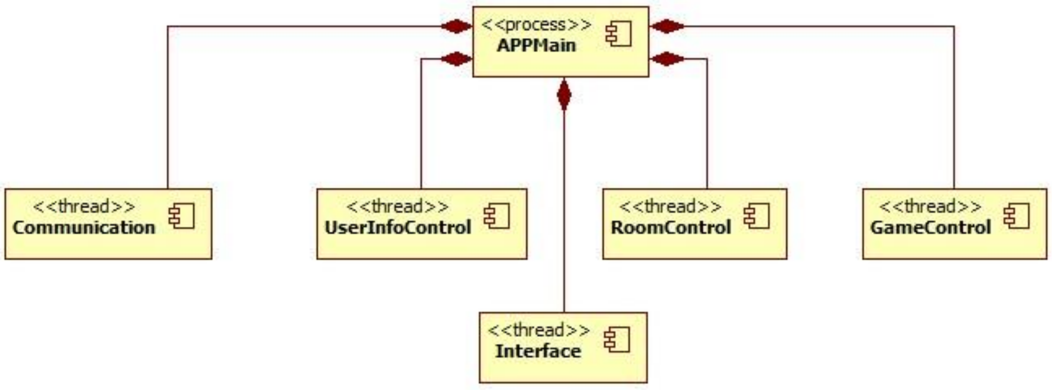
4) ShowResults



4.4 System Process View

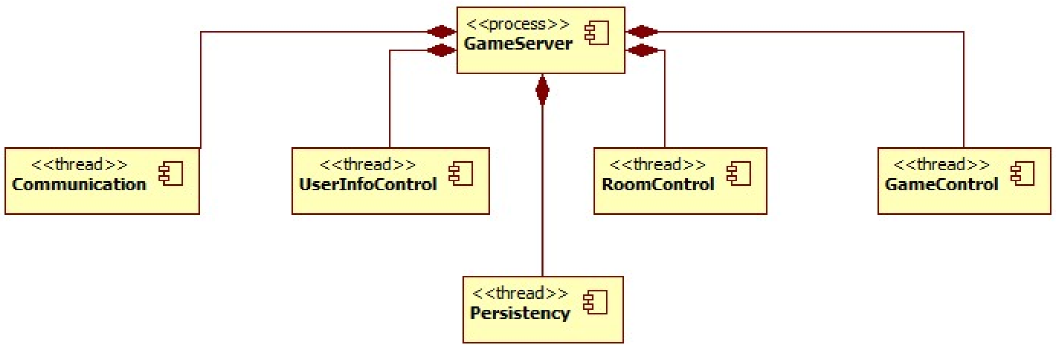
1) Client process diagram

On the client side, to provide the user with the best experience, UI thread only deal with the interaction between boundary objects and user. Each controller owns an exclusive thread, the same with communication module. Such design can ensure that the front end, function implementation and communication will not be block.



2) Server process diagram

In the design of server process, for the communication part, in response to the request of each customer, multithread management will be performed according to the concurrency of request. At the same time, each controller has their own thread. For persistent service, we will multi-threading mechanism to ensure its responsiveness.



4.5 System Implementation View

1) System development environment

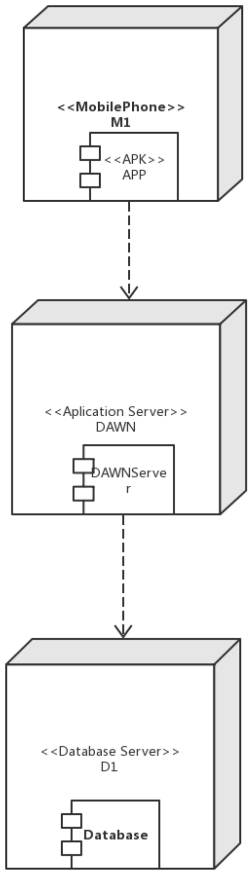
Development environment: Eclipse, AndroidStudio.

Development language: Java.

2) System development model

In this system, the corresponding software file definition is performed according to the component diagram. Therefore, the system development model is consistent with the component diagram in the software design model.

4.6 Physical View



Hardware configuration requirements

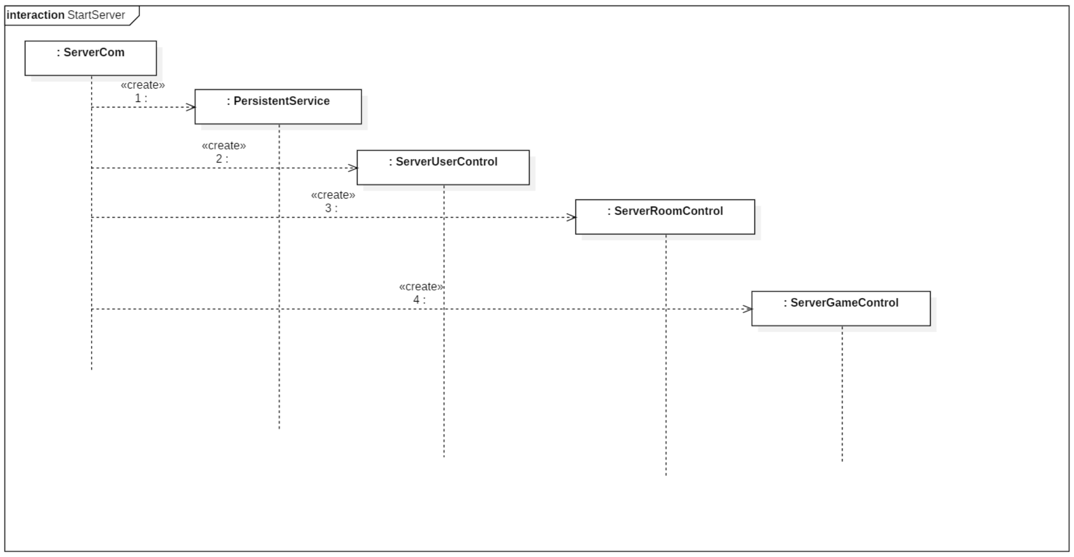
(1) user machine: a phone with at least 1.5GHz dual core processor and 2GB memory. To have good user experience, this is the minimum configuration to run most of the android game smoothly.

(2) Application Server: One or more Server with dual core i series processor, 4GB memory and 20GB storage. Considering the limited number of user, a laptop with similar or better configuration is enough.

(3) Network: 50Mbps internetwork. There are plenty of data exchange during a game, which means larger bandwidth is needed.

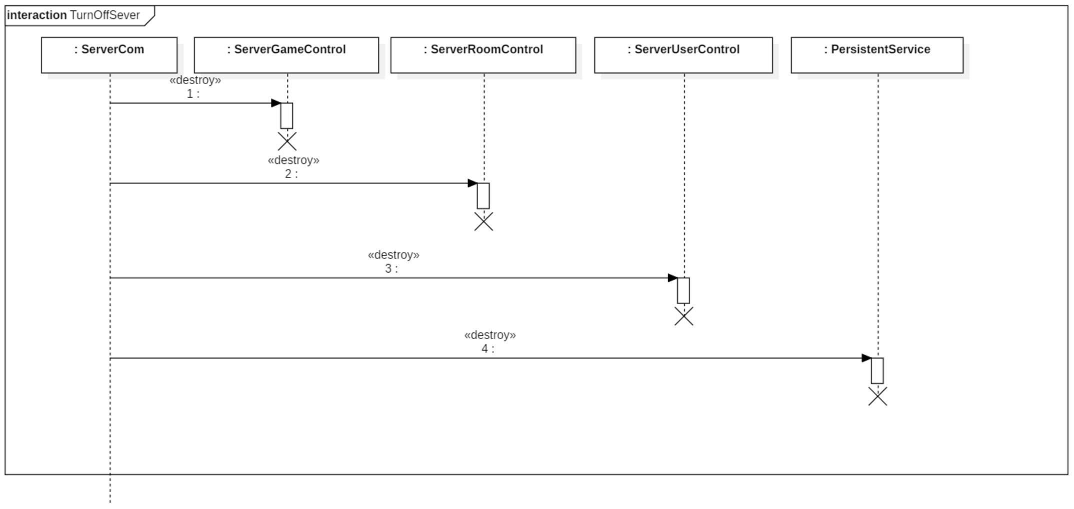
4.7 Boundary Condition Design

(1) Start Up Server



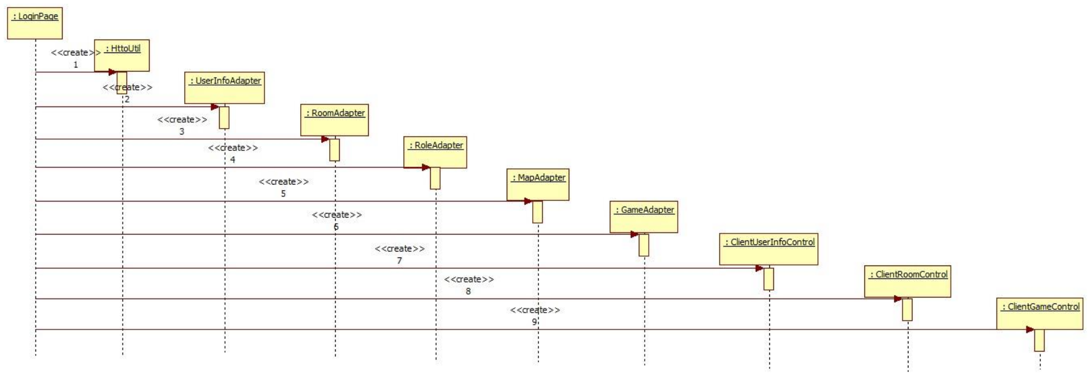
a) The user clicks on the app icon on the server
b) Start the ServerCom object.
c) Start the PersistentService PersistentService object
d) Start the SeverUserControl, SeverRoomControl , ServerGameControl object in turn, and pass the reference to the ServerCom object and the PersistentService object to these objects.

(2) Shut Down Server



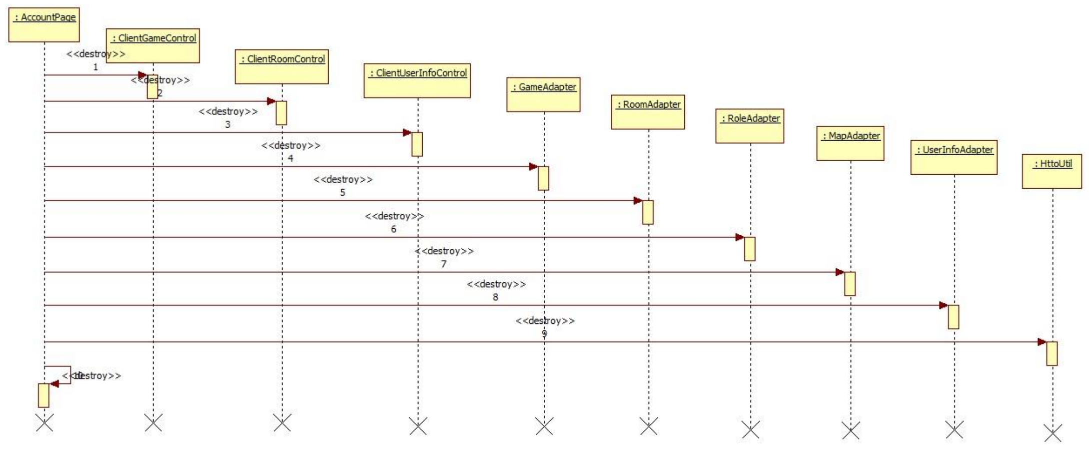
a) User clicks the app's close button
b) The system delete SeverGameControl, SeverRoomControl , ServerUserControl object in turn
c) The system disconnects from the database and deletes the PersistentService object.
d) The system deletes the ServerCom object.

(3) Start Up App



a) The user clicks the APP icon.
b) The system creates LoginPage object.
c) LoginPage object create HttpUtil object.
d) LoginPage object create UserInfoAdapter, MapAdapter, RoleAdapter, RoomAdapter and GameAdapter object and post HiipUtil object to them.
e) LoginPage object create ClientUserInfoControl, ClientRoomControl and ClientGameControl object and post the reference of UserInfoAdapter, MapAdapter, RoleAdapter, RoomAdapter and GameAdapter object to them individually.

(4) Shut Down App



a) Shut down APP on phone.
b) Shut down all the ClientControl objects in turn.
c) Shut down all the Adapter objects in turn.
d) Shut down HttpUtil object.
e) Shut down AccountPage.

(5) Abnormality
a) When system goes wrong, we first need to determine where the abnormality comes from.
b) If the problem came from server, we will have to logout all users and restart the server; if the problem came from app, we will only need to restart the app.
c) The database itself maintains its correctness by transactional features.

4.8 Data Management System

Considering that the data in the game is temporary, when a game comes to an end, data about this game in no longer needed. only the user information needs to be stored persistently.

This system has a large amount of user information and needs to be updated frequently. So we use a database to store user information and room lists.

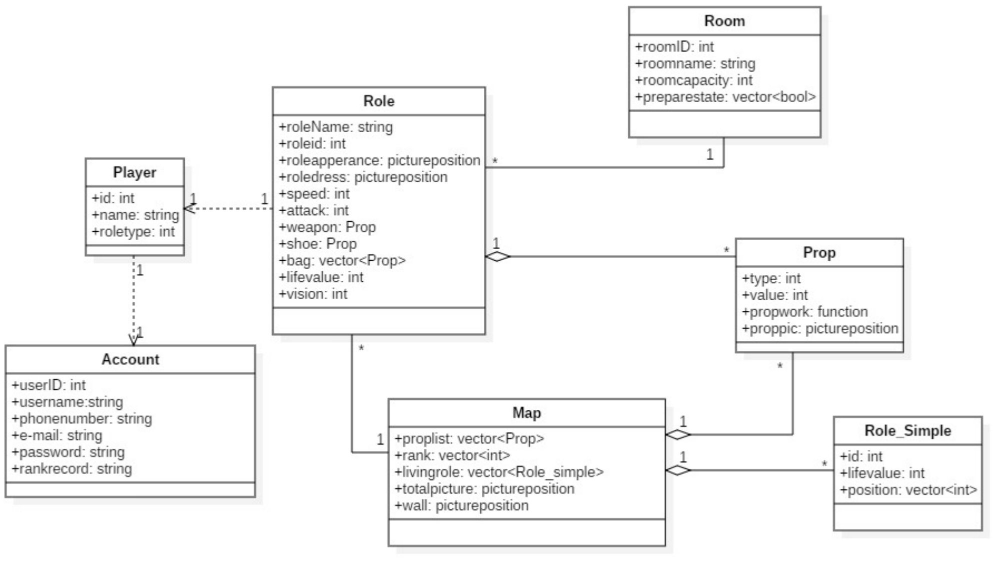
1) Account List

No.	Field	Description	Type	Empty	Key	Unit	Remark
1	userID	user's ID	int	N	Y		
2	password	user's password	String	N	N		
3	name	user's name	String	N	N		
4	PhoneNumber	user's phone	String	N	N		
5	e-mail	user's e-mail	String	Y	N		
6	RankRecord	user's rankrecord	String	N	N		

2) Room List

No.	Field	Description	Type	Empty	Key	Unit	Remark
1	roomID	room's ID	int	N	Y		
2	roomName	room's name	String	N	N		
3	roomCapacity	the maximum of players in the room	int	N	N		

The entity class diagram:



4.9 Other Design

(1) Access Control and Security Design

Use tables to list the permissions of different operators on different objects;

Describe the way user authentication is performed;

If necessary, give the data encryption/decryption method and give the security authentication of the interface call;

And other security issues.

(2) Anti-plug Design

Check each player's relevant data especially overly outstanding players.

(3) Reliability Design

If there are specific reliability requirements, give a specific design approach to reliability.