DAWN

Software Design Model

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 Design Model	Zihan Xu, Yi Kuang, Chenyu Yang, Yuting Lan, Jianzhen Cao

SOFTWARE DESIGN PATTEN

1 Introduction

1.1 Purpose

The purpose of this software design model document is to organize various design models of our game. Based on the previous requirements specification and system analysis, the structure of the system is displayed in detail, laying the foundation for the subsequent software implementation. We will construct logical views, implementation views, process views and deployment views. We will be able to develop the whole project based on it.

1.2 Domain

The Dawn game system. The characteristics, subsystems and models that related to the system meet the content of this document

1.3 Definition

See more details in glossary document.

1.4 Reference

<<Object-Oriented Software Engineering Practice Guide-2>> Shanghai Jiao Tong University Press, 2016

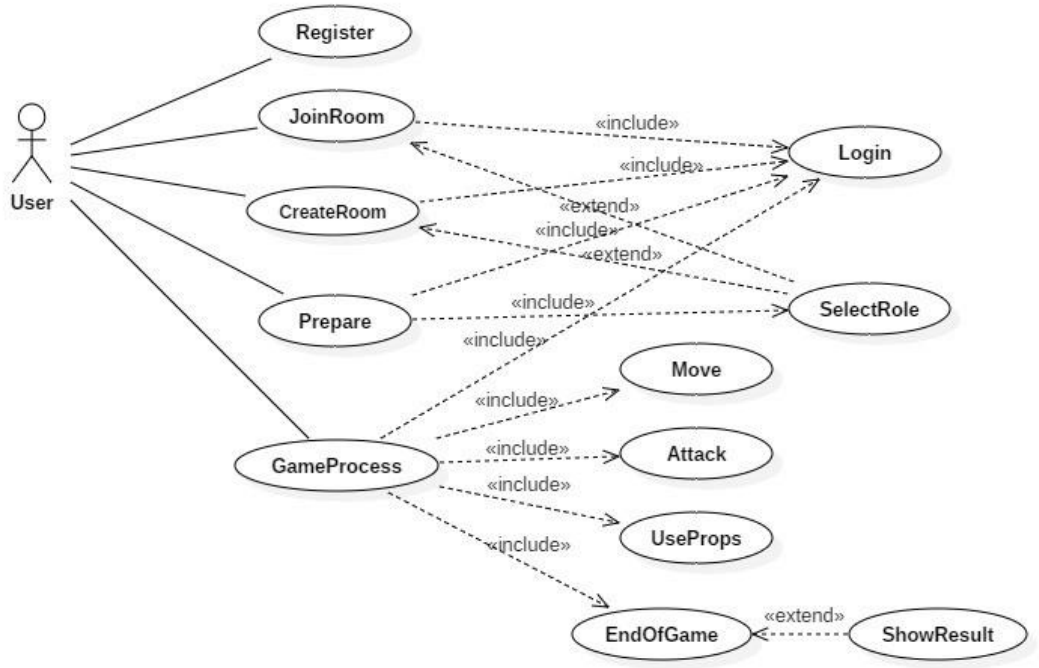
<<Object-Oriented Software Engineering - Using UML, Patterns, and Java>> (3rd edition), Tsinghua University Press, 2011

1.5 Overview

This document includes introduction, Use-case Diagram, Logic Diagram, Implementation Diagram, Process View and Deployment View. The Use-case Diagram displays functions provided by our app. Logical View talks about classes of each subsystem and Implementation Diagram displays how the functions are realized with some components. Process View lists each thread of the process. Deployment View shows an arrangement in physical level.

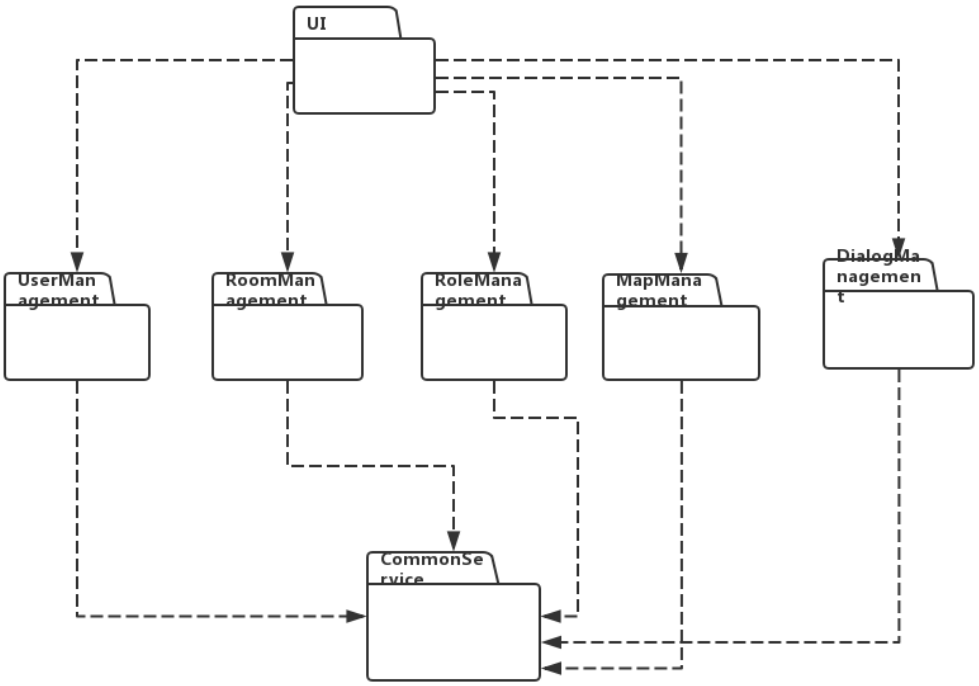
2. Use Case View

The complete use case model:

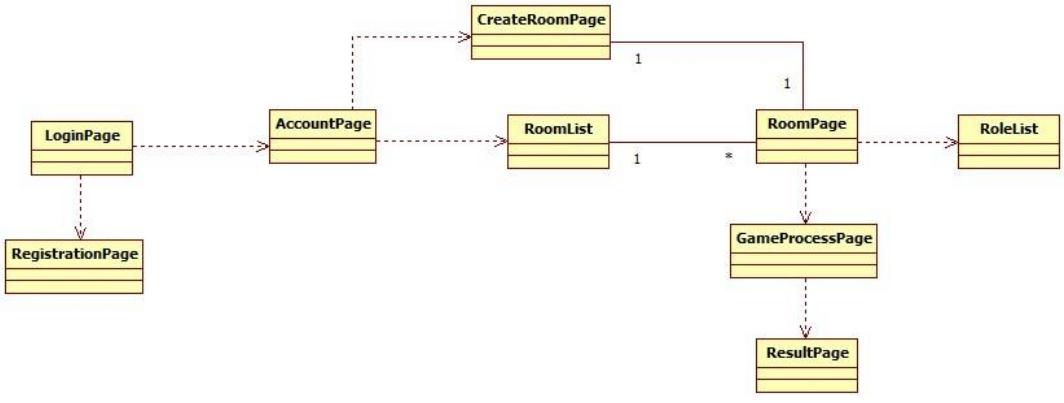


3. Logical View

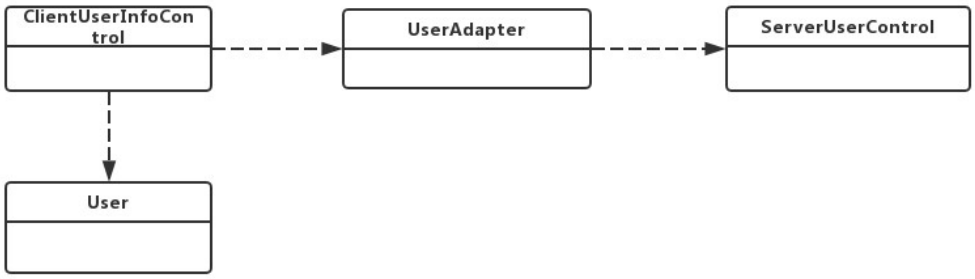
3.1 System structure



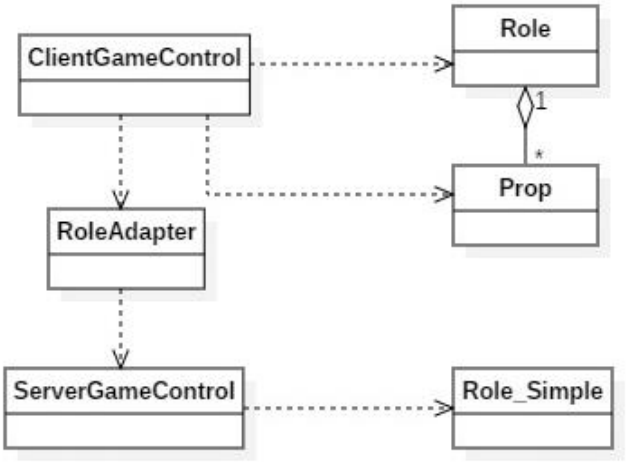
(1) User-Interface Subsystem



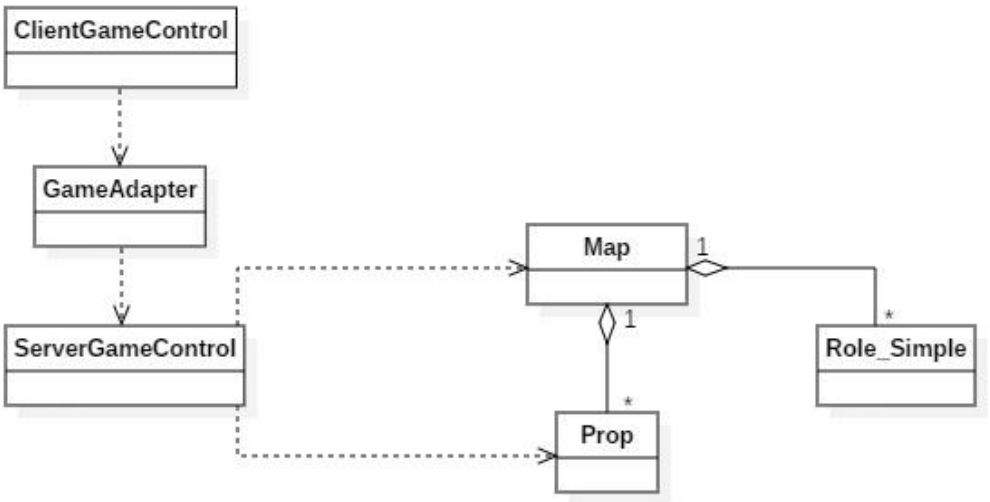
(2) User-management Subsystem



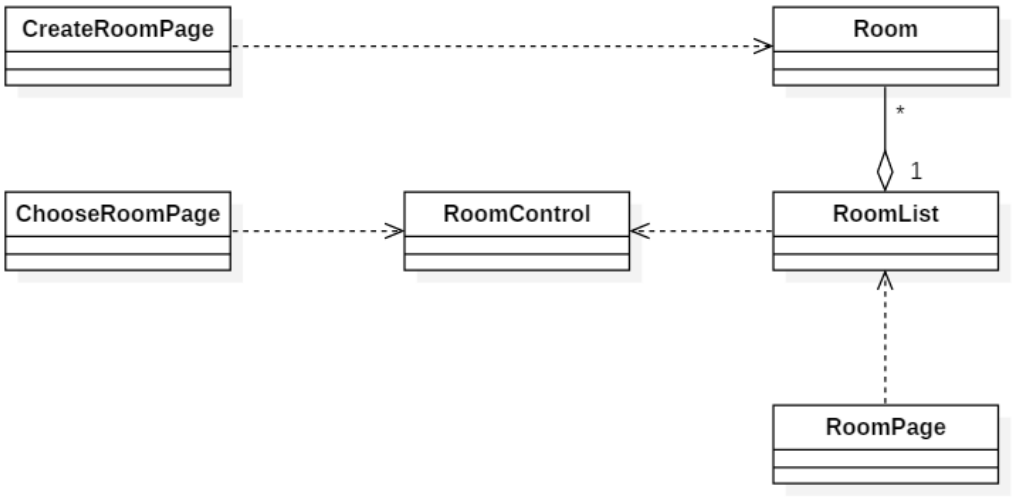
(3) Role-management Subsystem



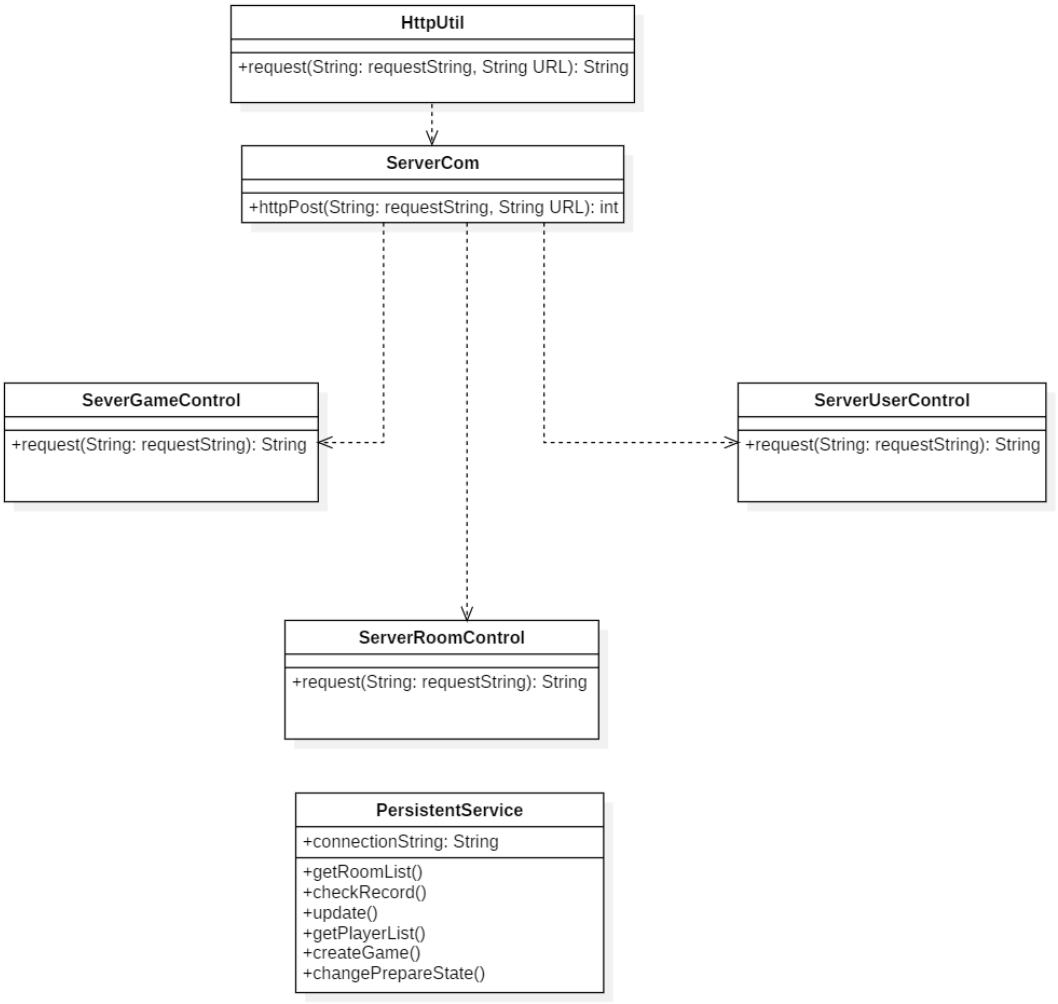
(4) Map-management Subsystem



(5) Room-management Subsystem

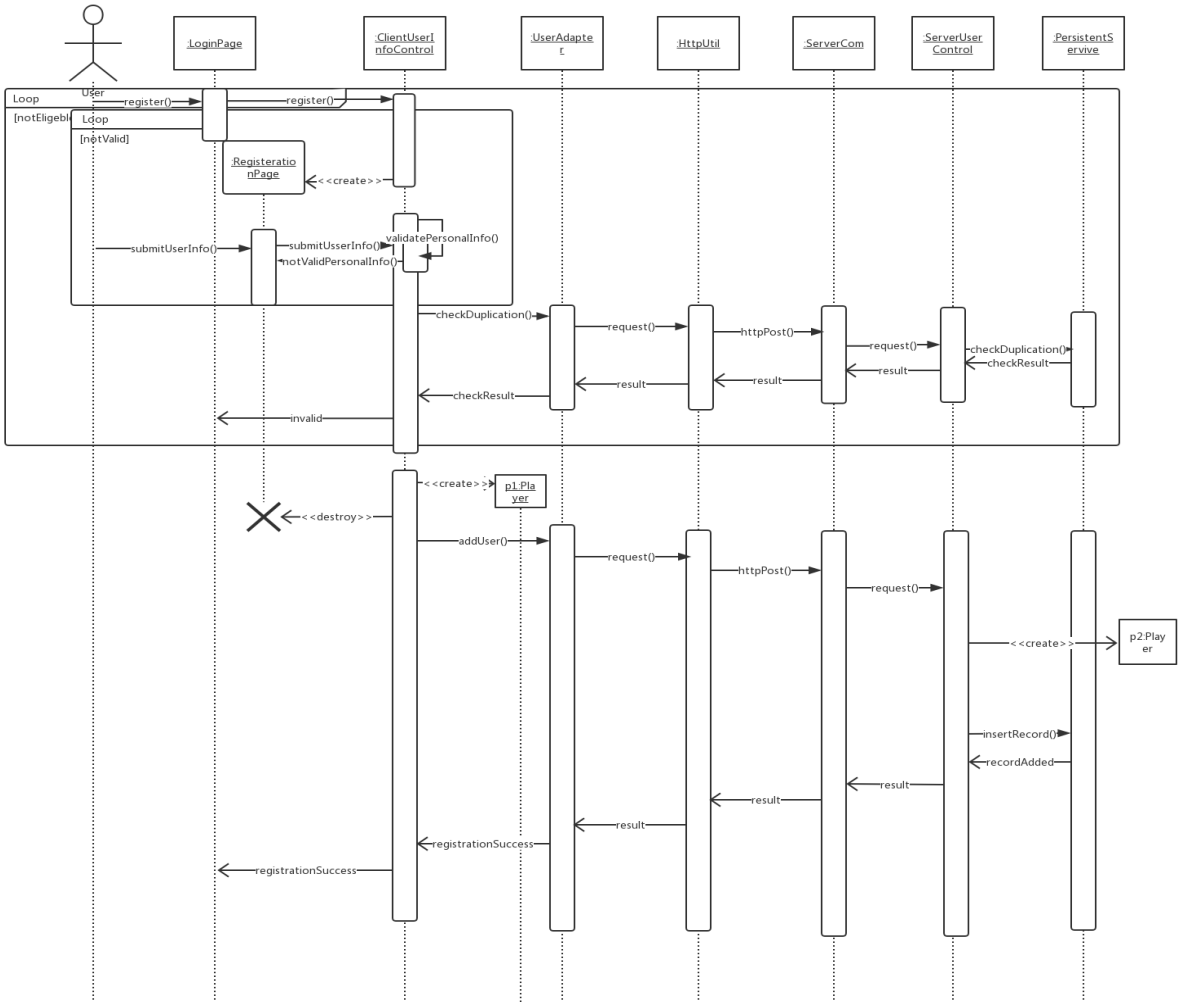


(6) Common-service Subsystem

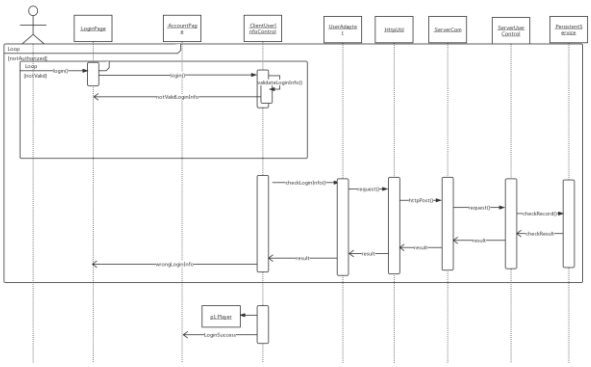


3.2 Use Case View

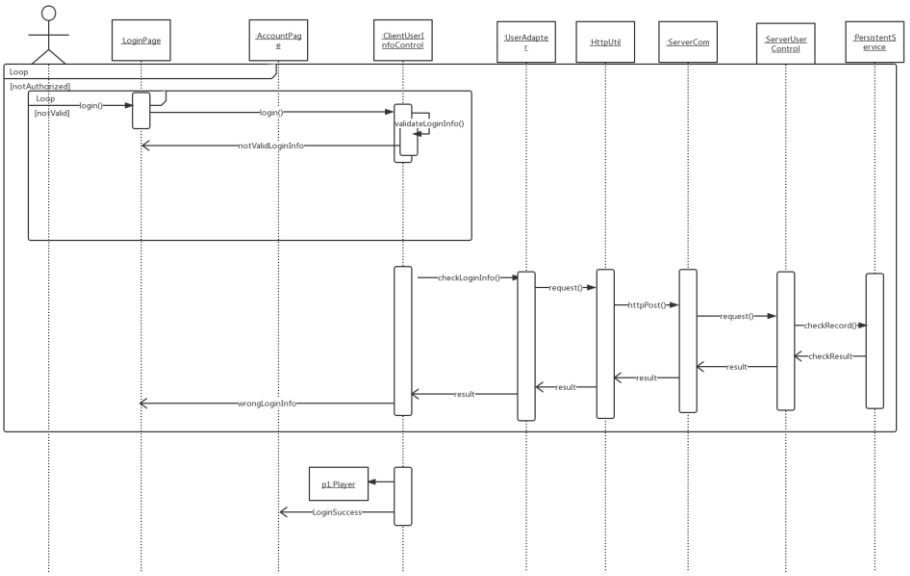
(1) <Register> Implementation



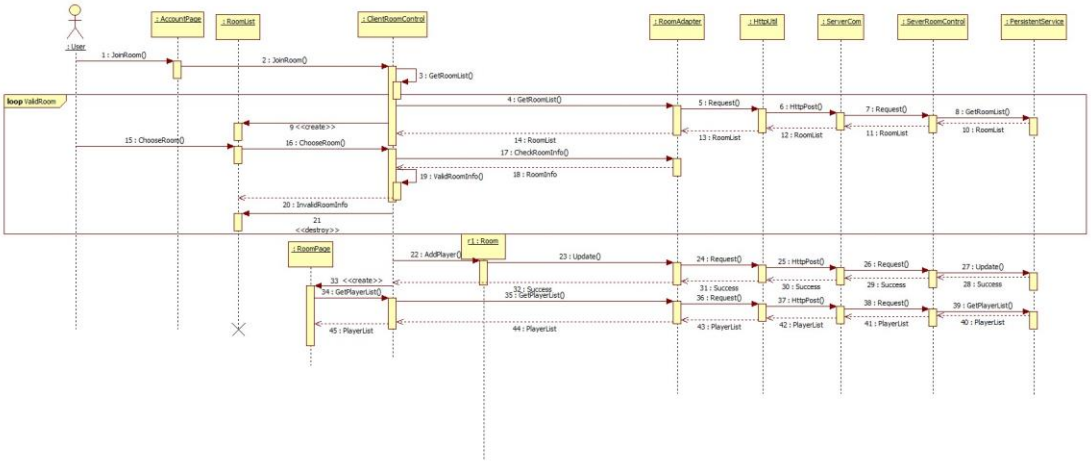
(2) <Login> Implementation



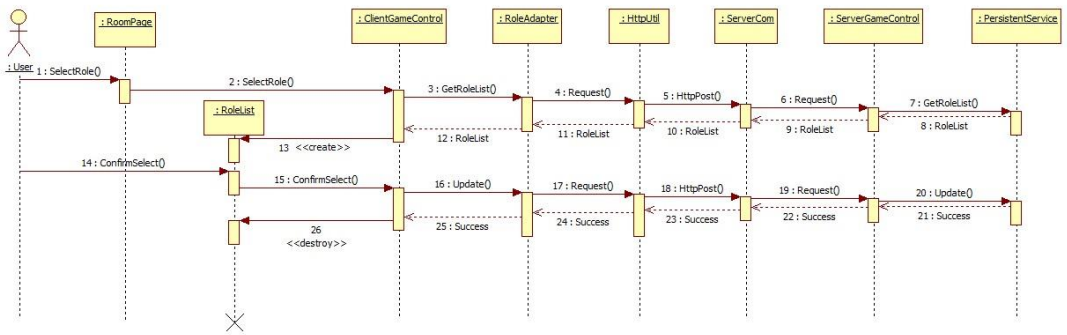
(3) <CreateRoom> Implementation



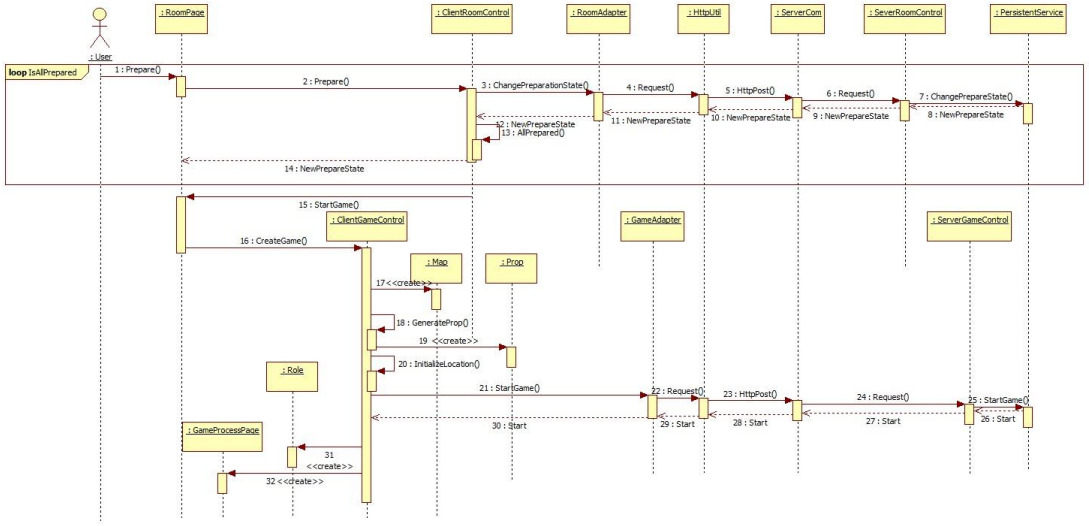
(4) <JoinRoom> Implementation



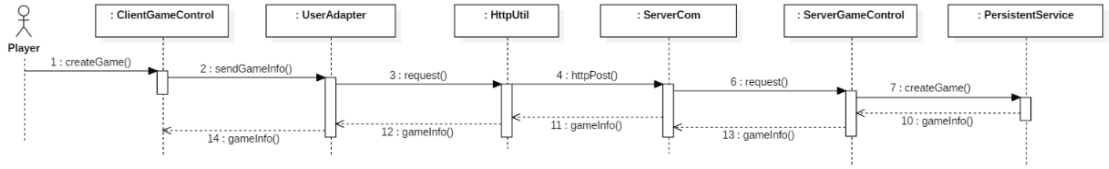
(5) <SelectRole> Implementation



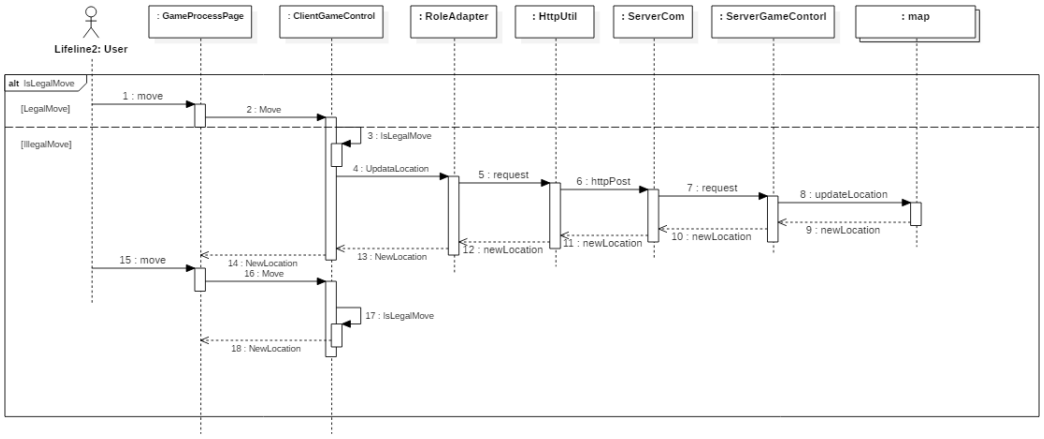
(6) <Prepare> Implementation



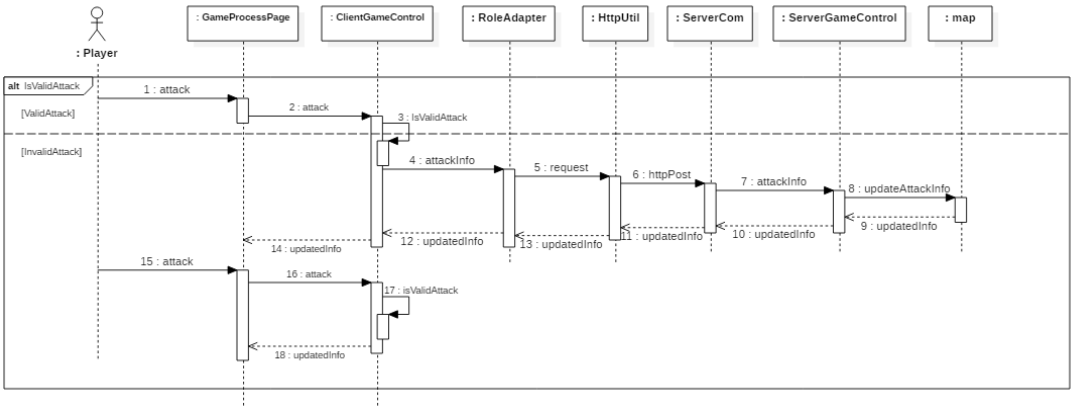
(7) <GameProcess> Implementation



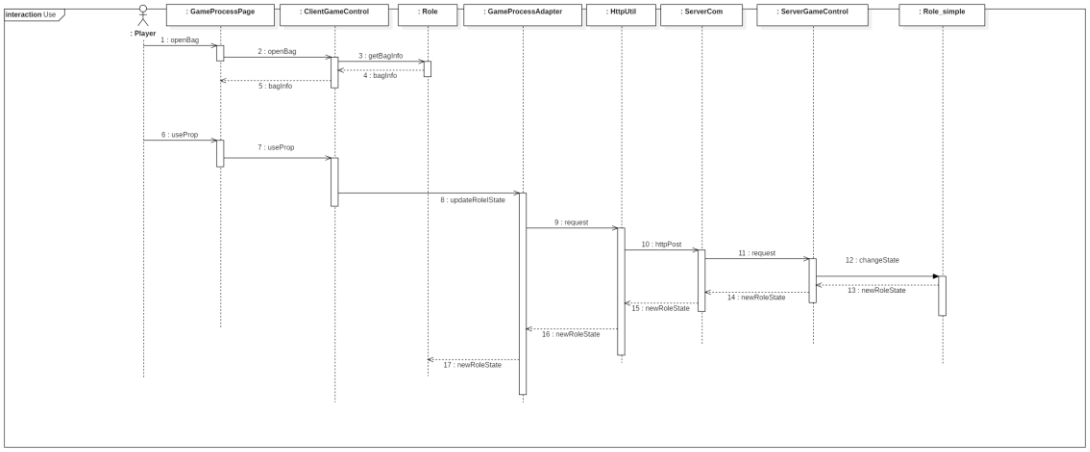
(8) <Move> Implementation



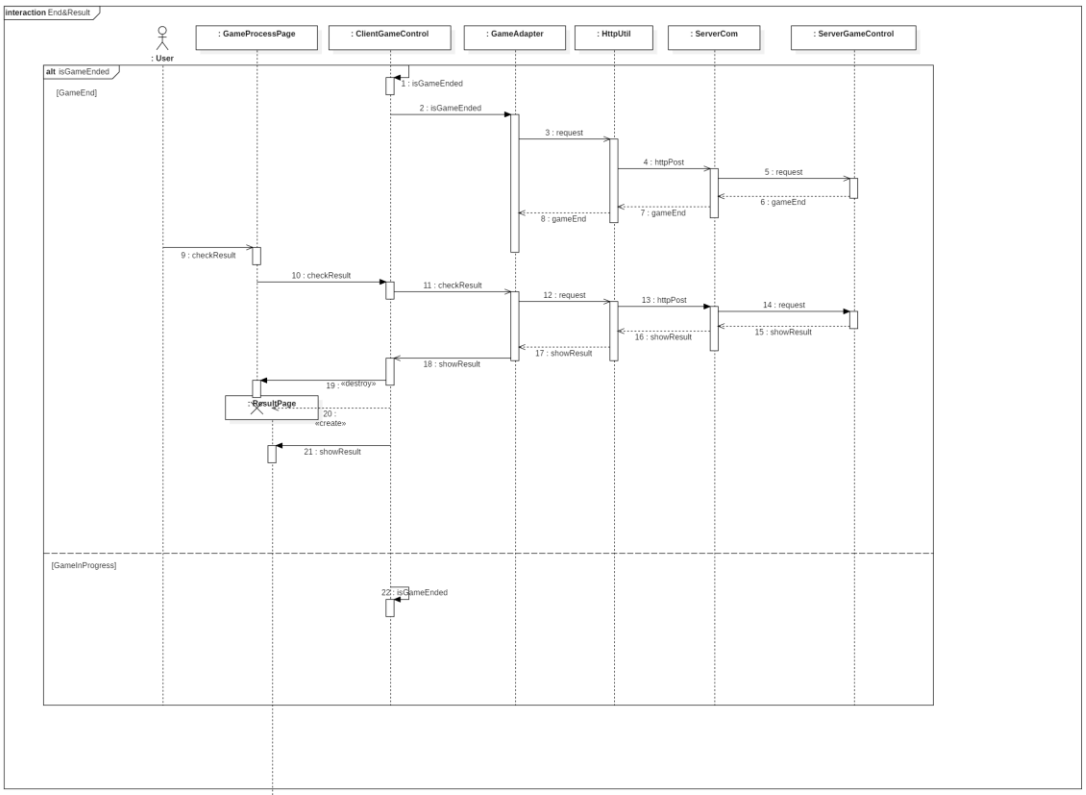
(9) <Attack> Implementation



(10) <Use> Implementation

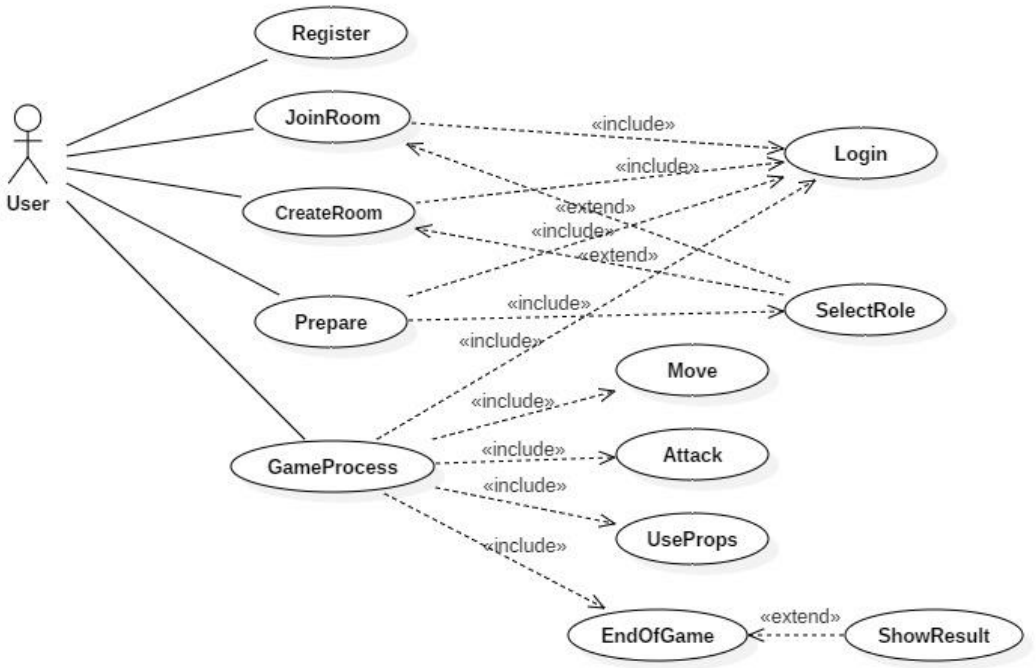


(11) <End&Result> Implementation

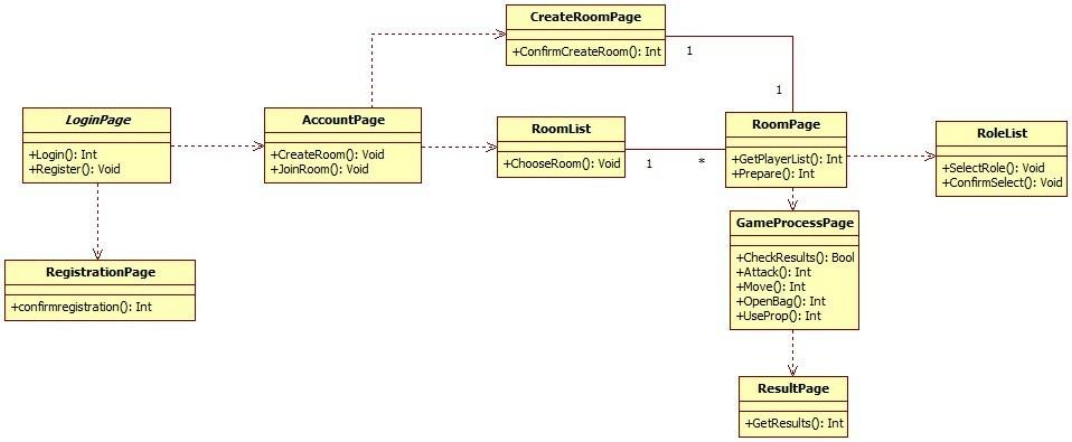


3.3 Design View

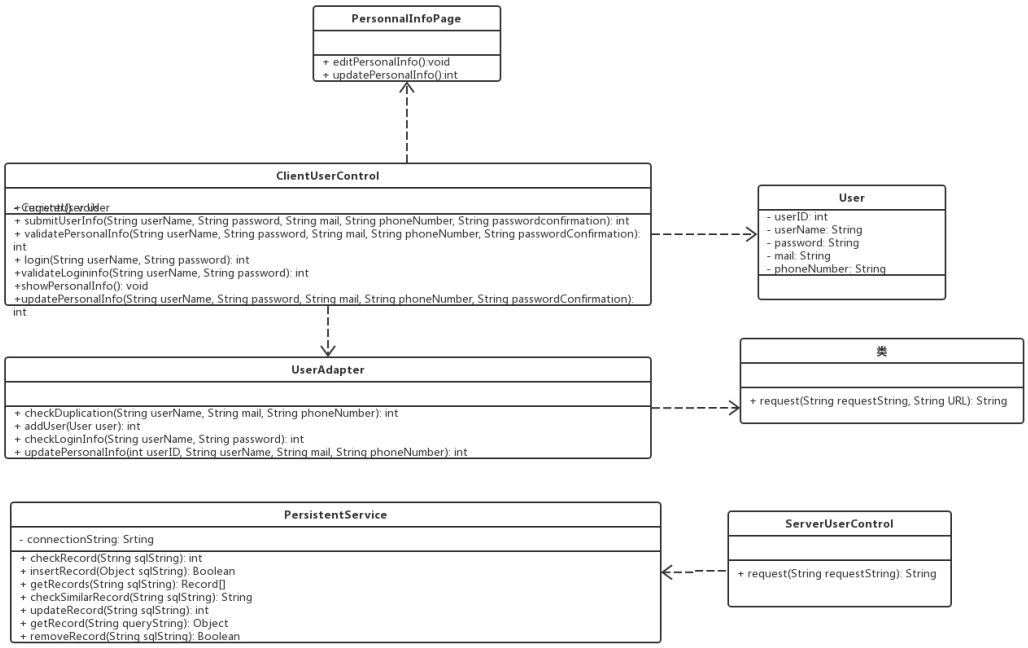
Since there are lots of classes, we decided to create a Package Diagram and then draw Class Diagrams on the basis of subsystems.



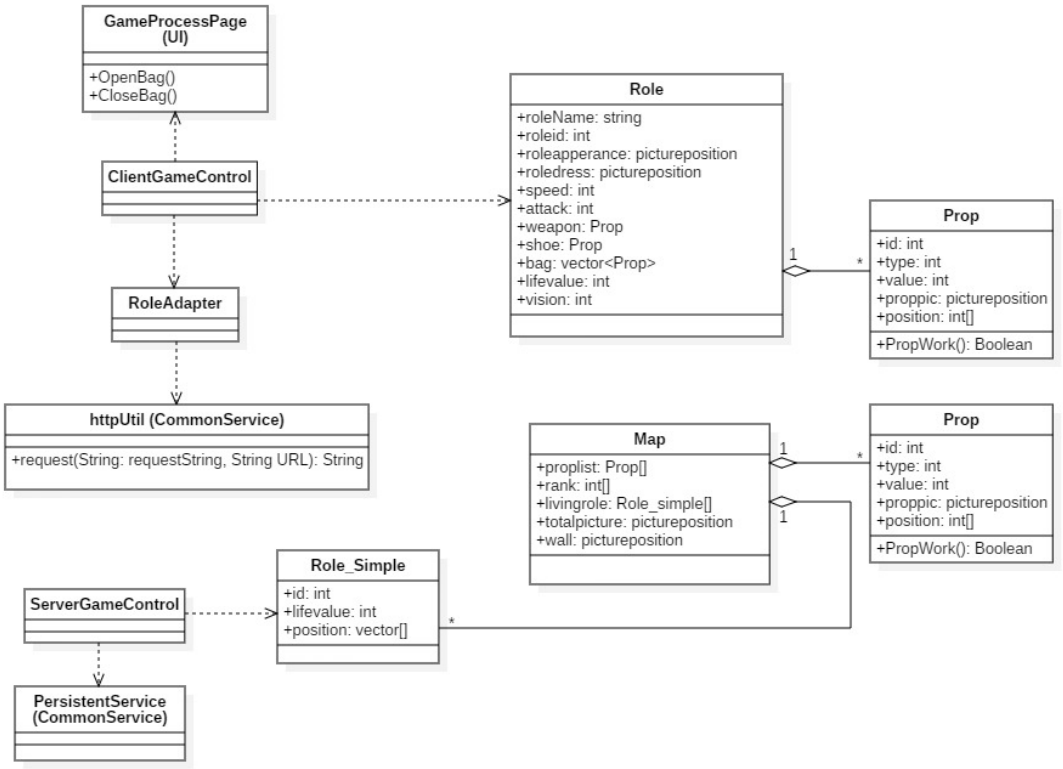
(1) User-Interface Subsystem



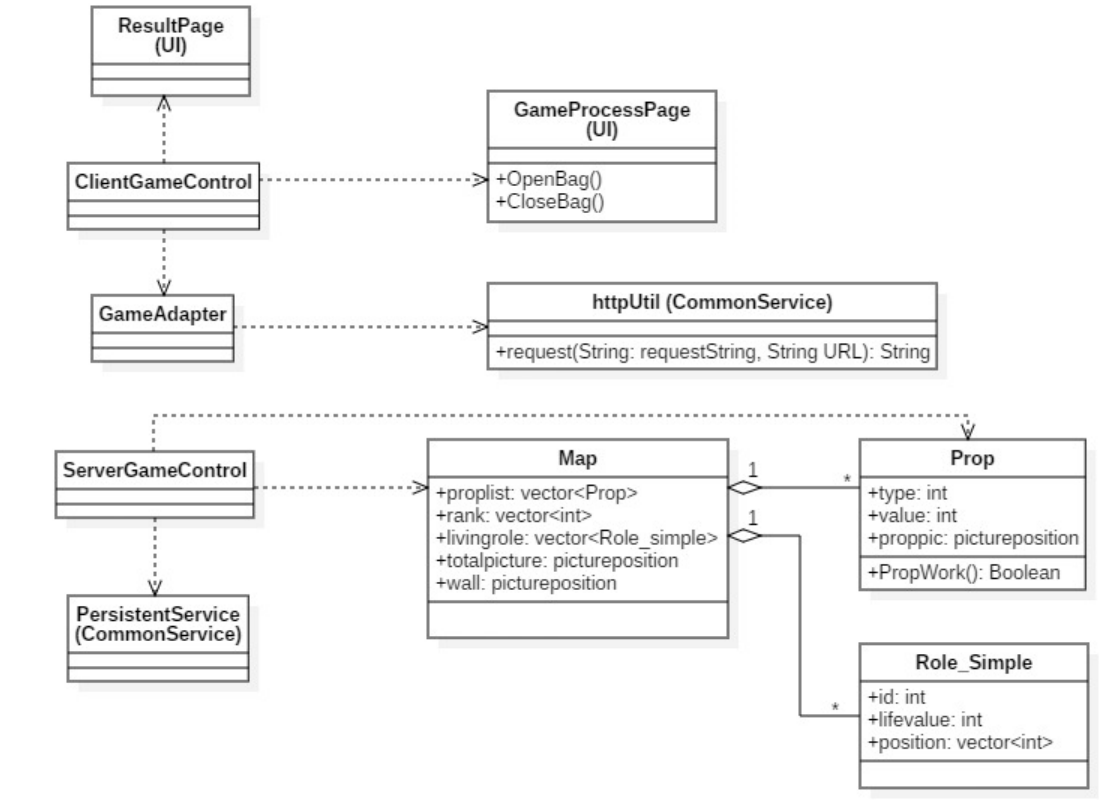
(2) User-management Subsystem



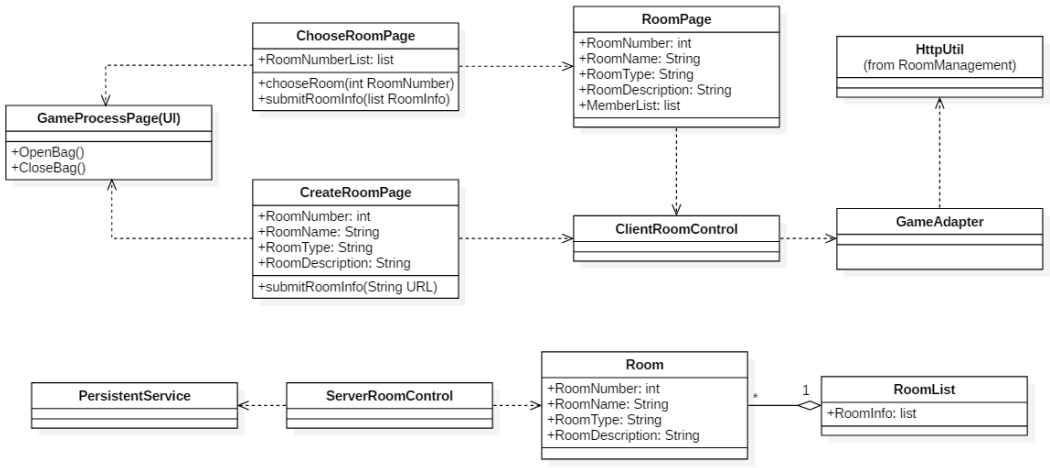
(3) Role-management Subsystem



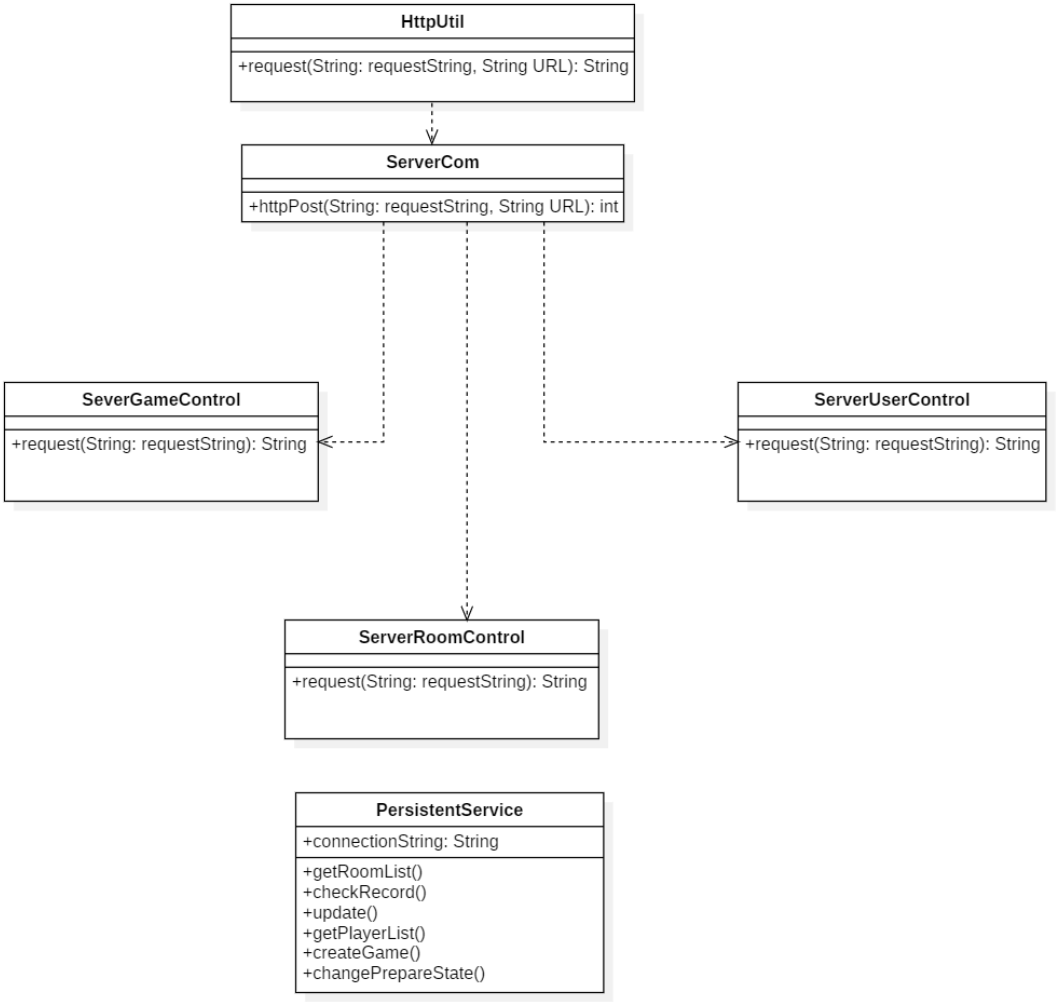
(4) Map-management Subsystem



(5) Room Management Subsystem



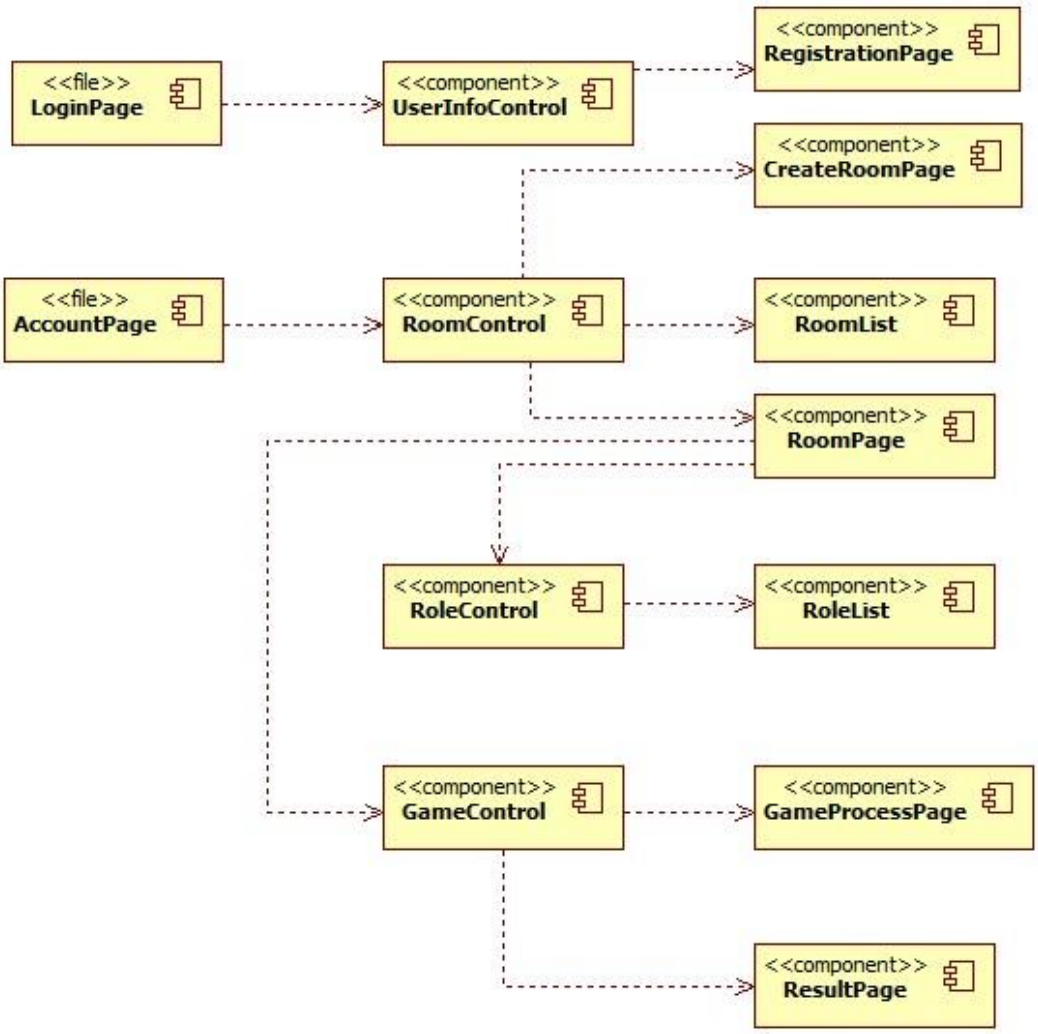
(6) Common-service Subsystem



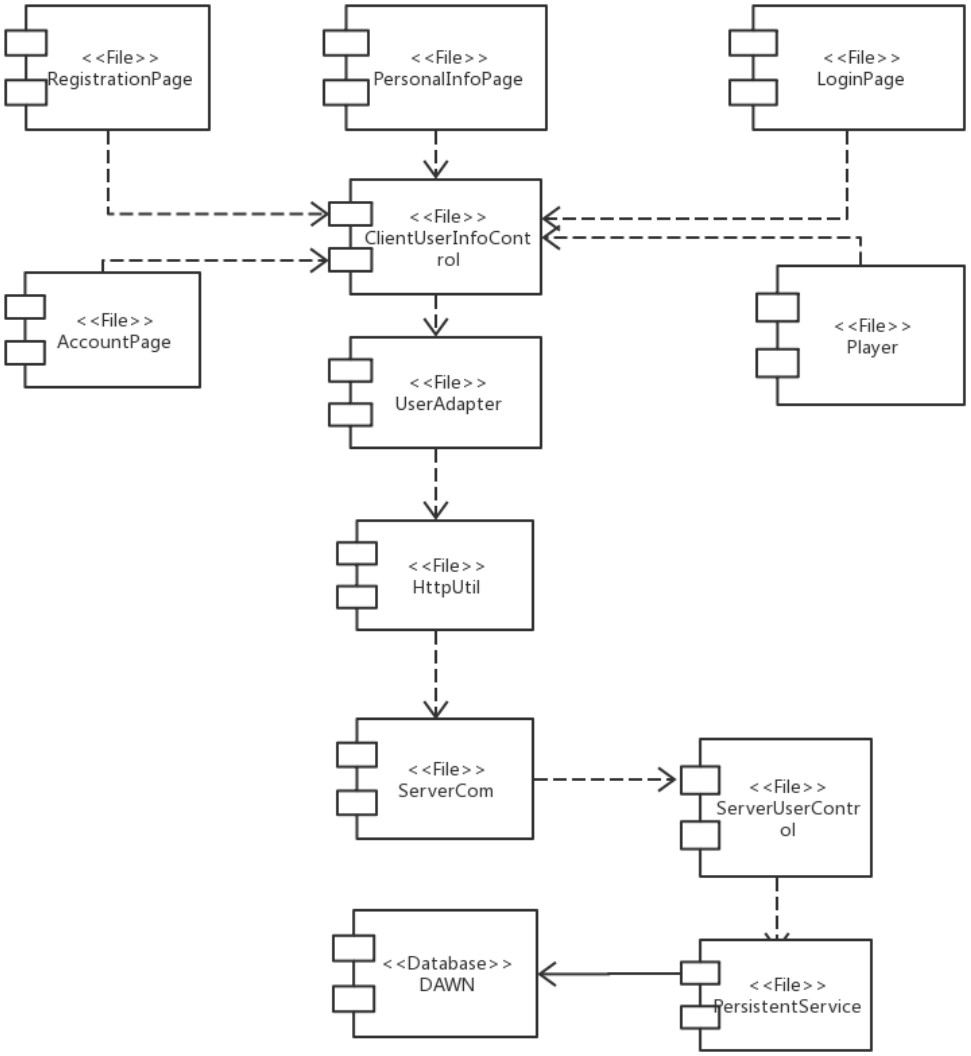
4. Implementation View

（1） Development Model Composition

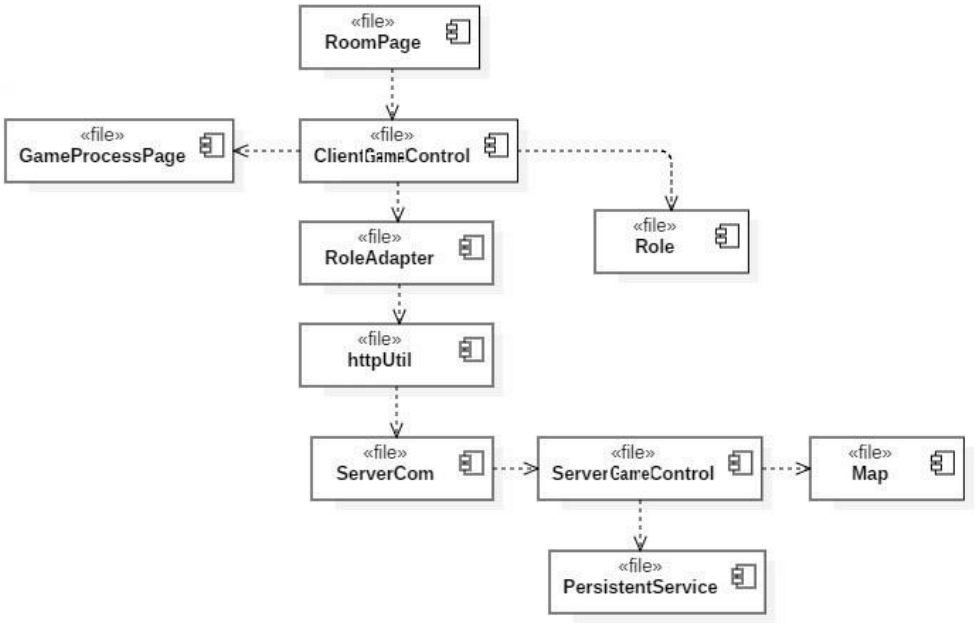
a) User-Interface Subsystem



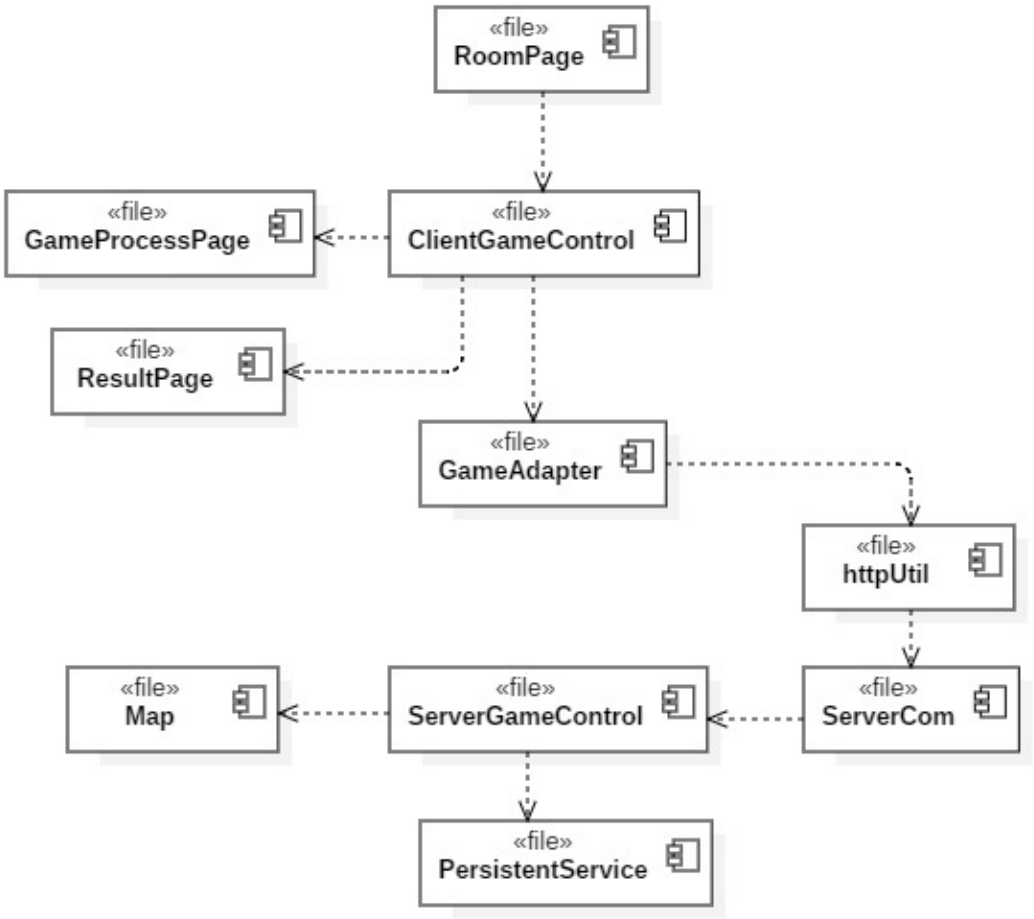
b) User-management Subsystem



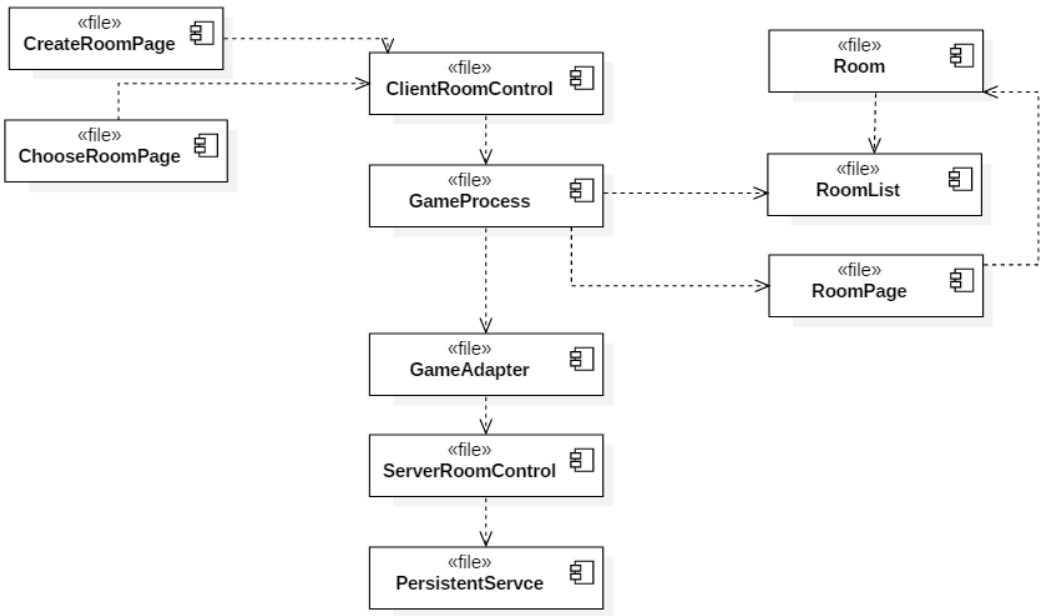
c) Role-management Subsystem



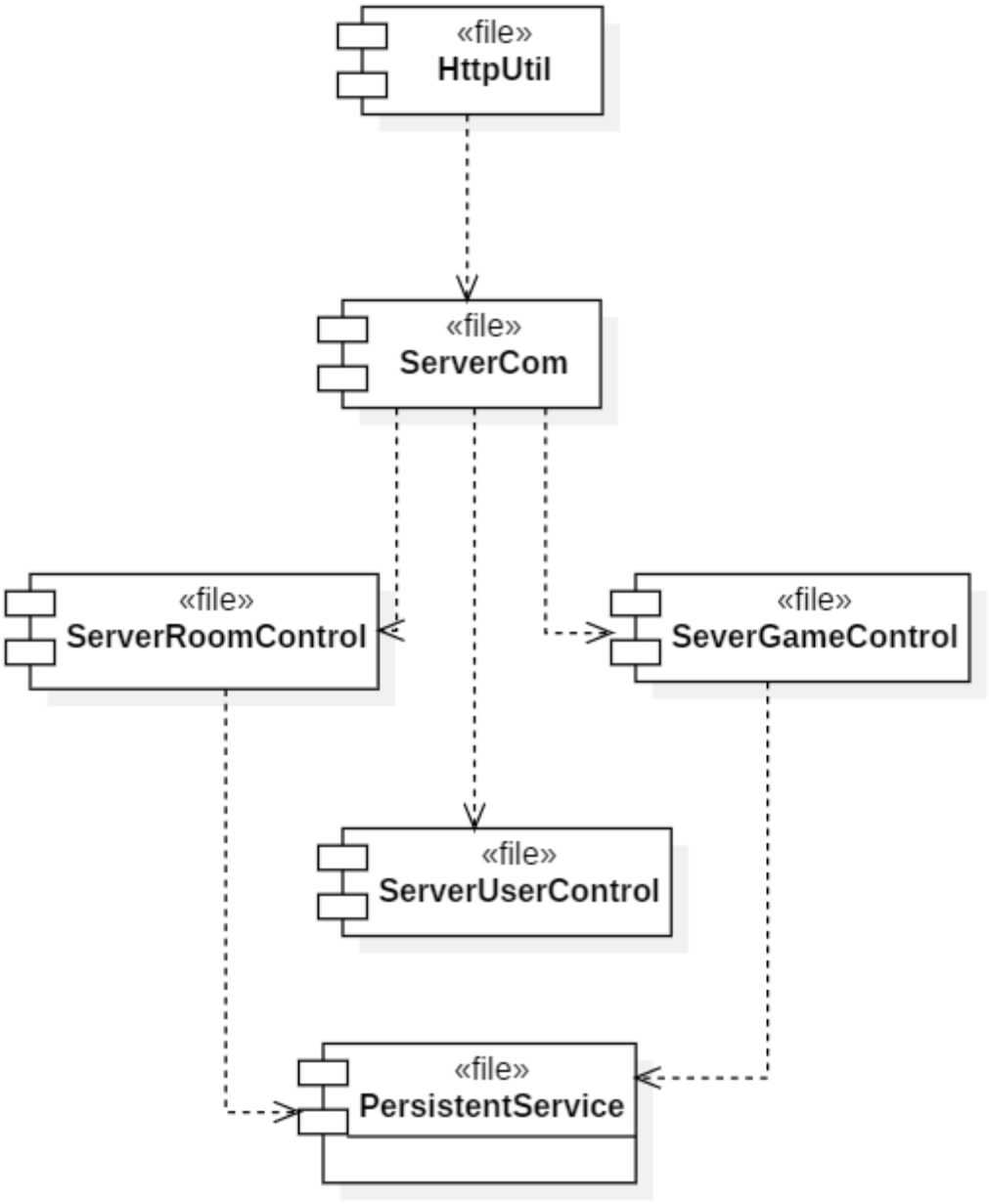
d) Map-management Subsystem



e) Room-management Subsystem



f) Common-service Subsystem

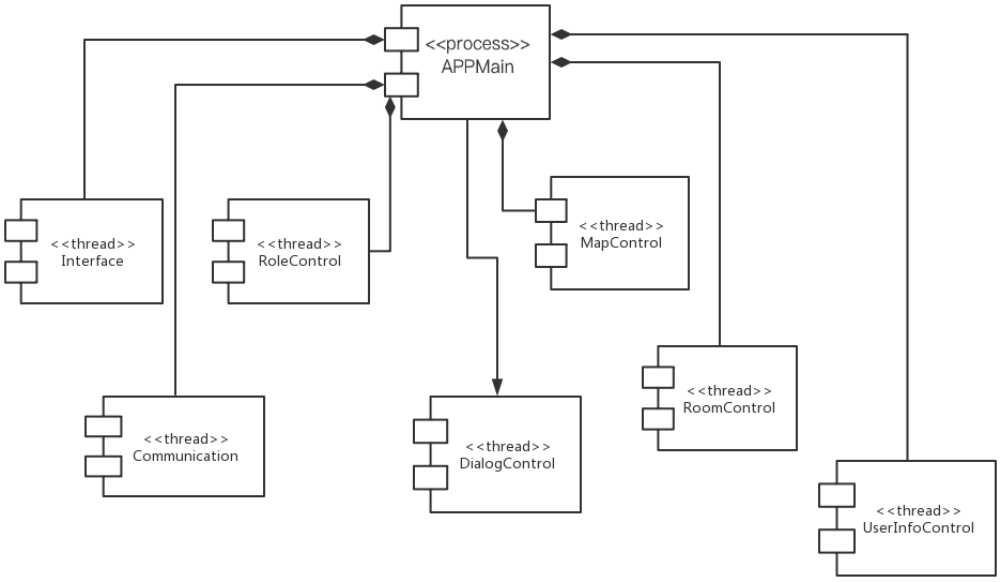


（2） Post-compilation

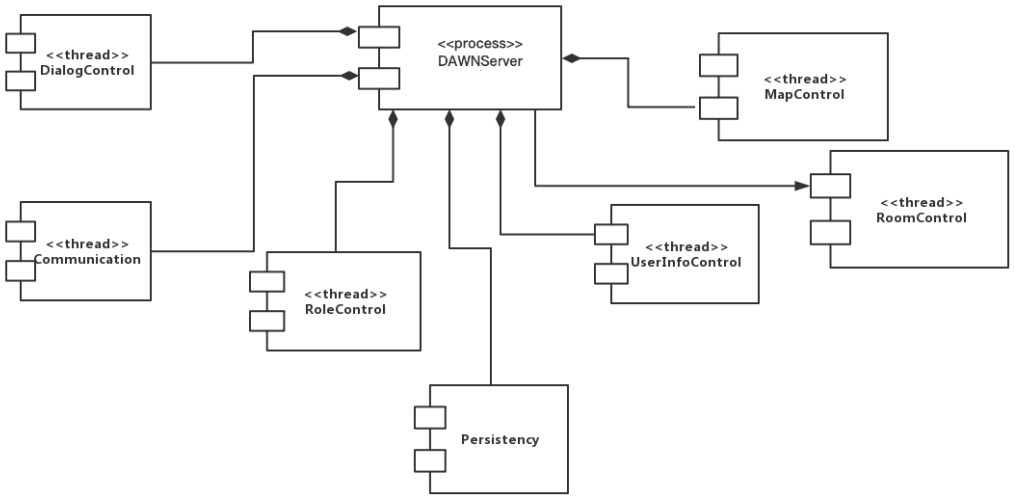
Since each one of the .java file will be compiled to generate a corresponding .class file, the compiled component map should be identical to the development model composition.

5. Process View

(1) Client Process View



(2) Server Process View



6. Deployment View