< Journey Assistant >
Software Design Model
Version 2.0

Group Member

Yiwen Song
Zhihui Xie
Weizhe Wang
Huangfei Jiang
Haoping Chen

Modification History

Date	Version	Description	Author
2019-04-16	1.0	The first version of this document.	Huangfei Jiang & Haoping Chen
2019-06-18	2.0	Modify some description.	Zhihui Xie

Contents

1 Introduction

1.1 Purpose

The purpose of this software design model is to collect and draw various models of our 'Journey Assistant' software. We will present the system structure in detail on the basis of former requirement specifications and system analysis, and lay the foundation of subsequent software implementation. This document will focus on our system and use UML to construct the Logic View, Implementation View, Process View, Deployment View, etc. It is designed for the development team to specify the various design models of the 'Journey Assistant' software, and base on which to start developing of this system.

1.2 Scope

This document is applied to our Journey Assistant System and other subsystems or models related with our software. It is a specialized document for our product.

1.3 Definition

There is no extra abbreviation shown in this document. Refer to our 'Glossary' document if any question occurs.

1.4 Bibliography

(1) <Feasibility Study Feedback> (GB8567-88)

(2) <Object Oriented Software Engineering (Version 3)> (Tsinghua University Press)

(3) <Object Oriented Software Engineering Practice Guidelines>

1.5 Summary

This document includes models of the following 5 parts: 'Use Case View', 'Logic View', 'Implementation View', 'Process View' and 'Deployment View'. 'Use Case View' presents all possible use cases of the system. 'Logic View' mainly includes system architecture diagram, design class diagram and implementation of the use case. 'Implementation View' construct component diagrams for each subsystem. 'Process View' uses class diagrams and component diagrams to represent the processes and threads of the system. 'Deployment View' presents software and hardware implementation of the system. Each part of this document is closely related with each other, and present the design model of the system jointly.

2 Use Case View

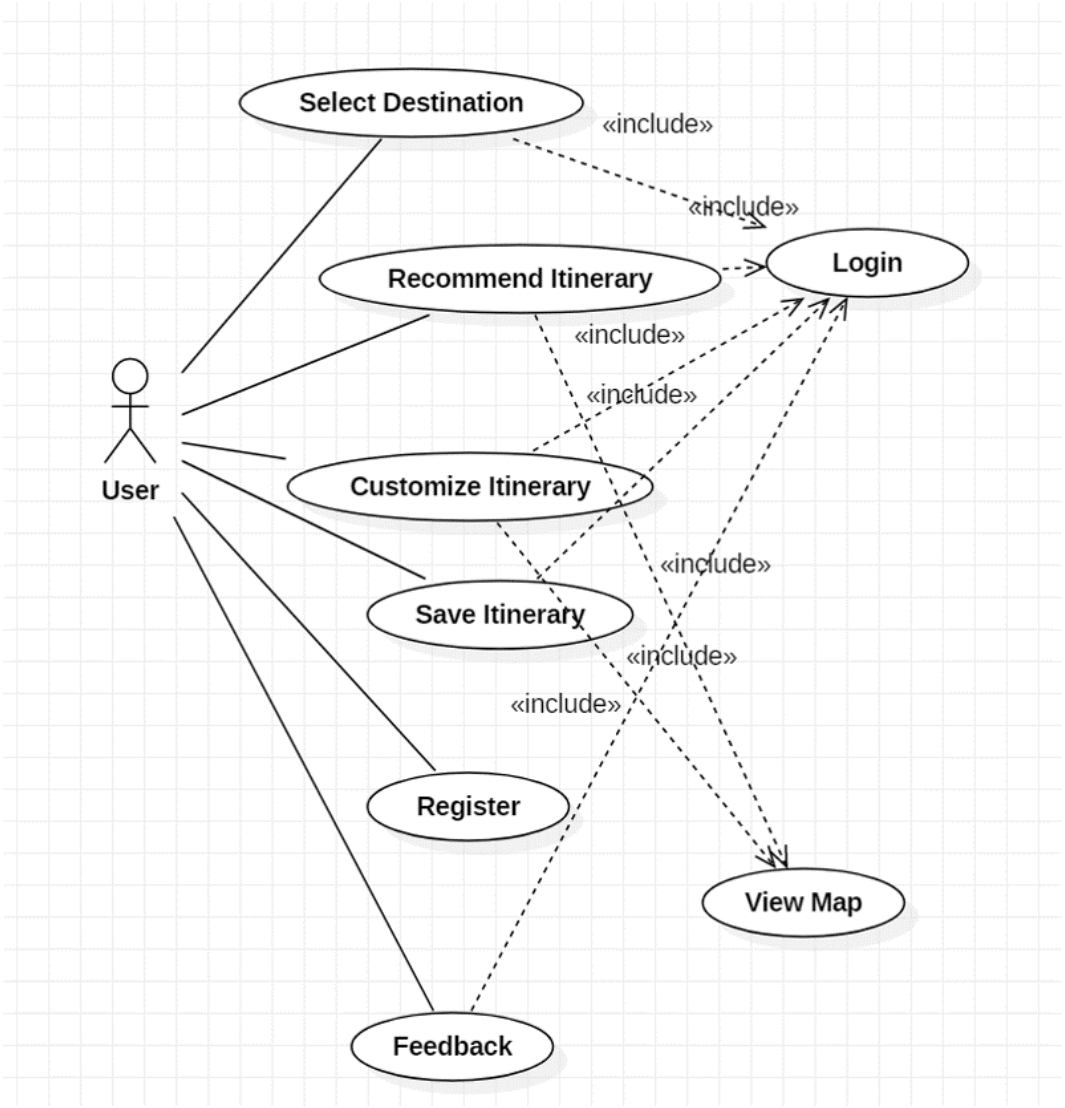

Figure 1: Use Case View

3 Logic View

3.1 System Architecture

The whole system can be decomposed into 4 subsystems: UI subsystem, user management subsystem, journey management subsystem, and reedback management subsystem.

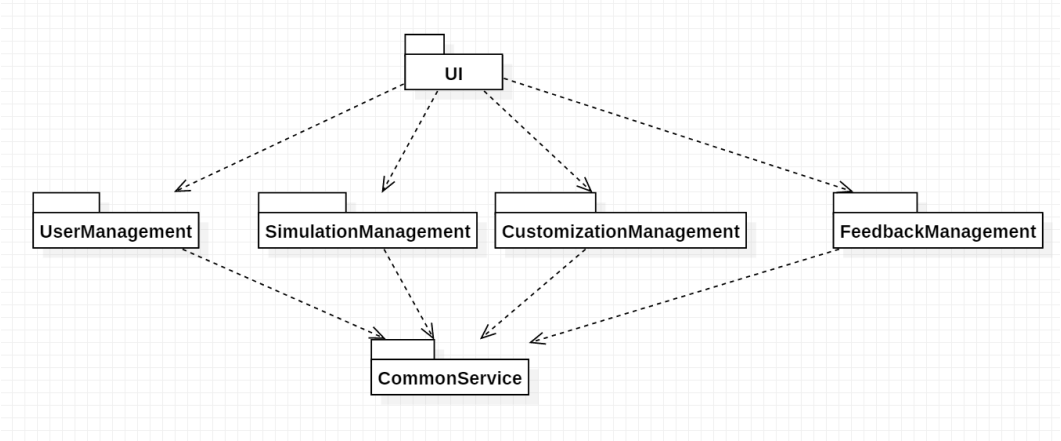

Figure 2: System Architecture

The following class diagrams (Figure 10 - 6) show inner logic for every package.

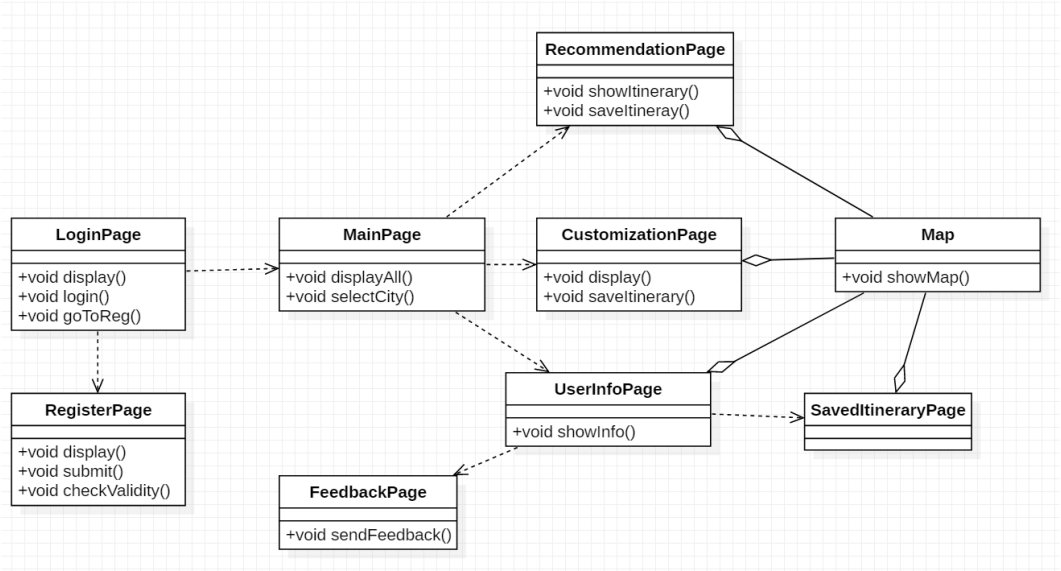

Figure 3: UI Subsystem

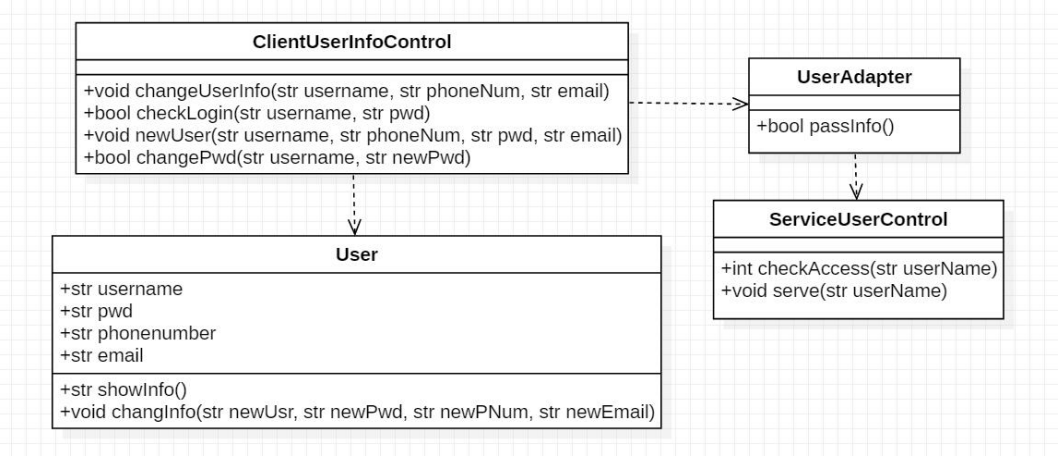

Figure 4: User Management Subsystem

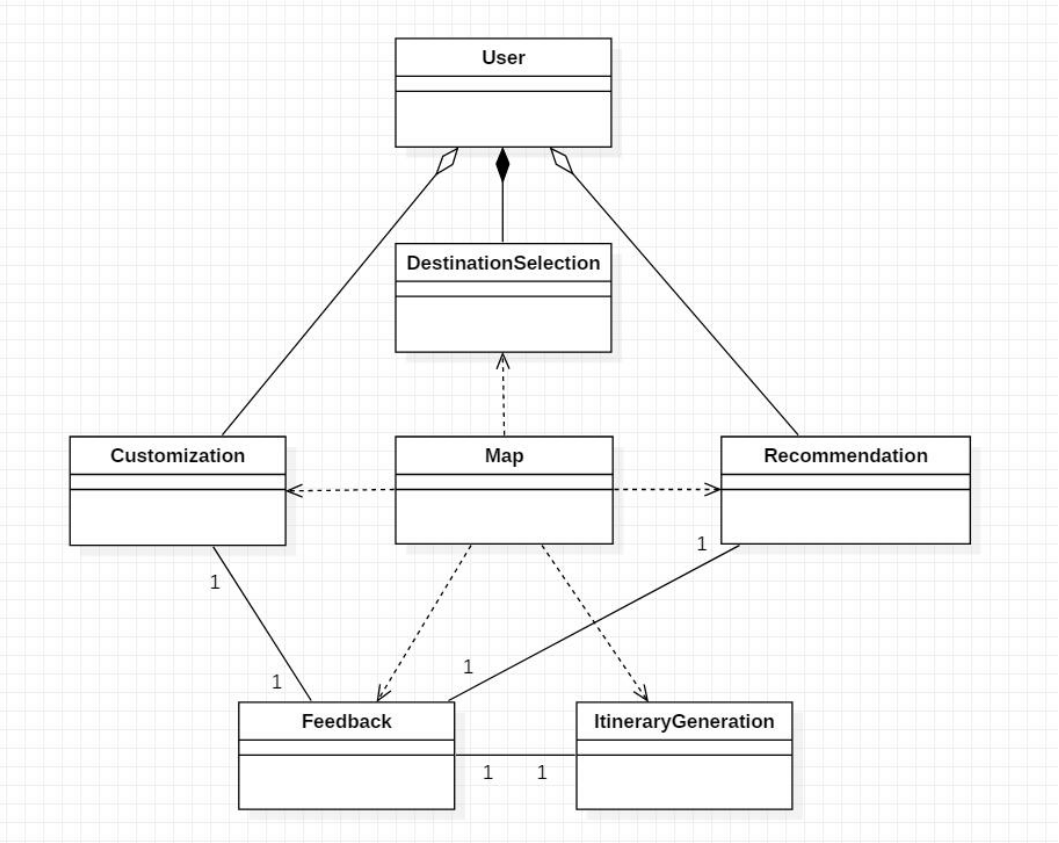

Figure 5: Journey Management Subsystem

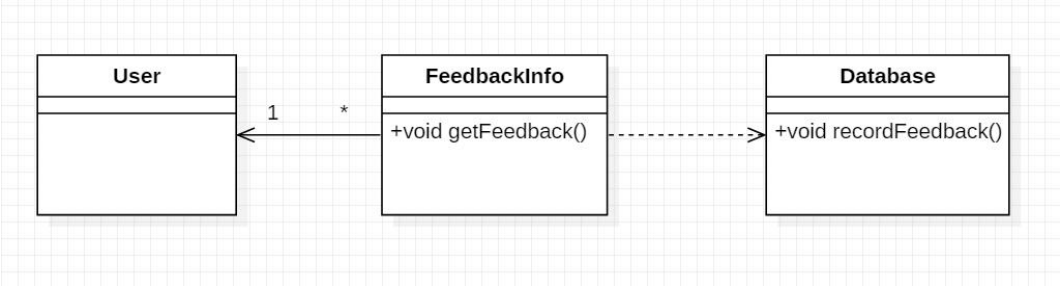

Figure 6: Feedback Management Subsystem

3.2 Use Case Realization

For each type of use case, we use an interaction diagram to display our corresponding inner design.

3.2.1 Log-in Realization

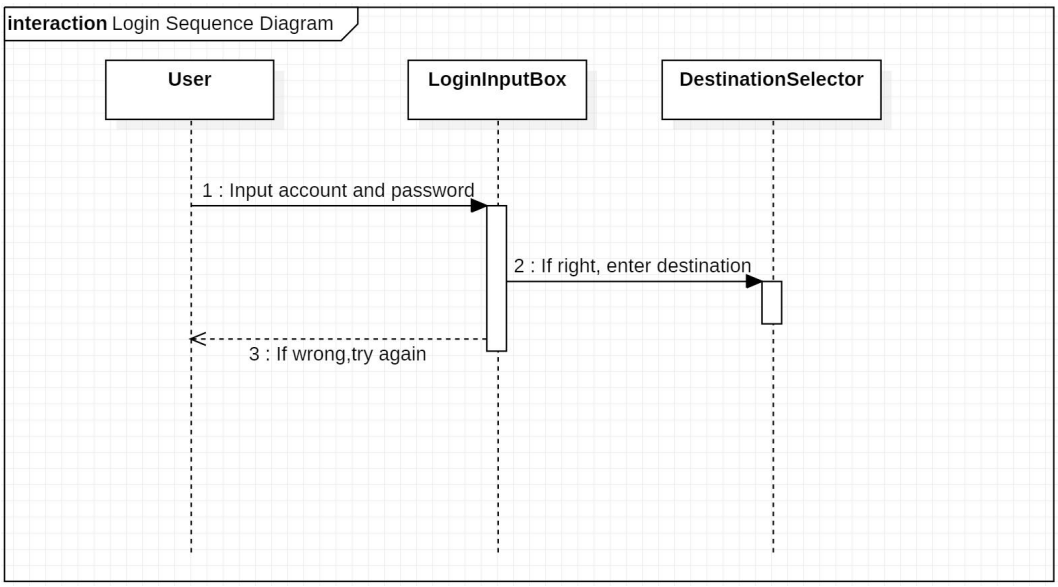

Figure 7: Log-in Sequence Diagram

3.2.2 Recommendation Realization

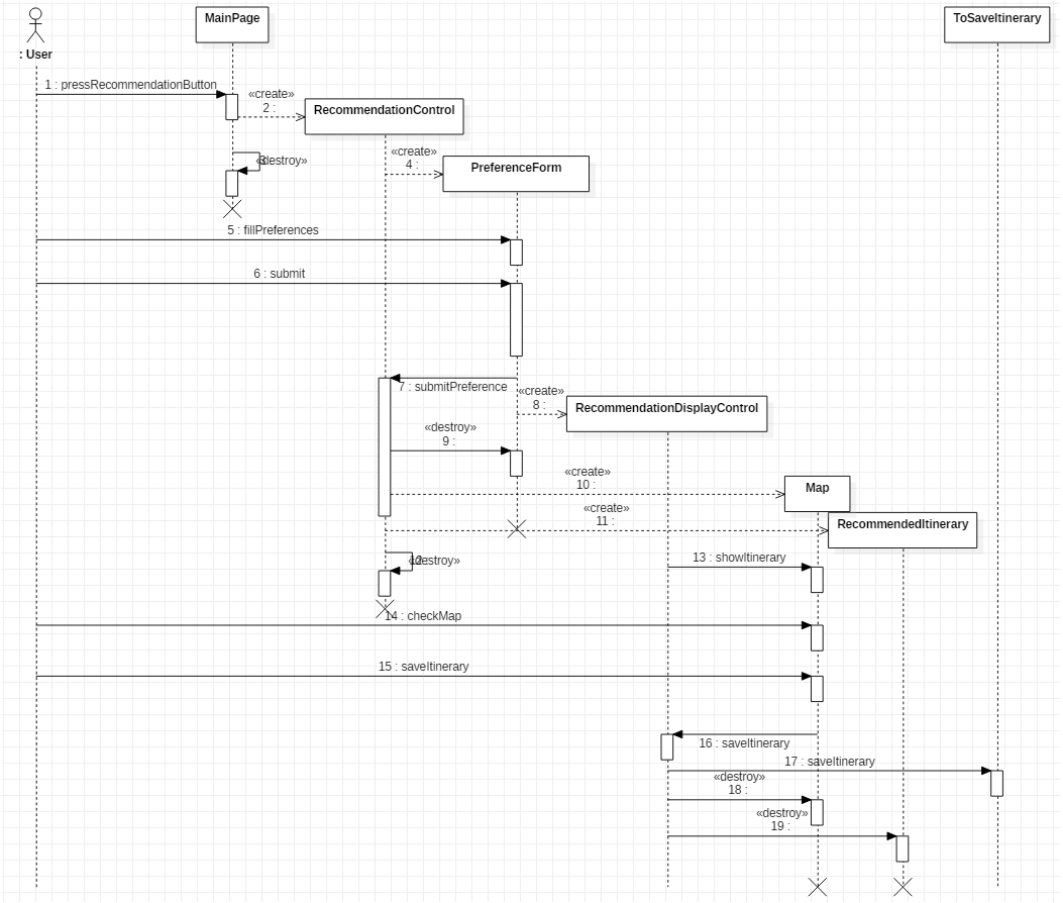

Figure 8: Recommendation Sequence Diagram

3.2.3 Feedback Realization

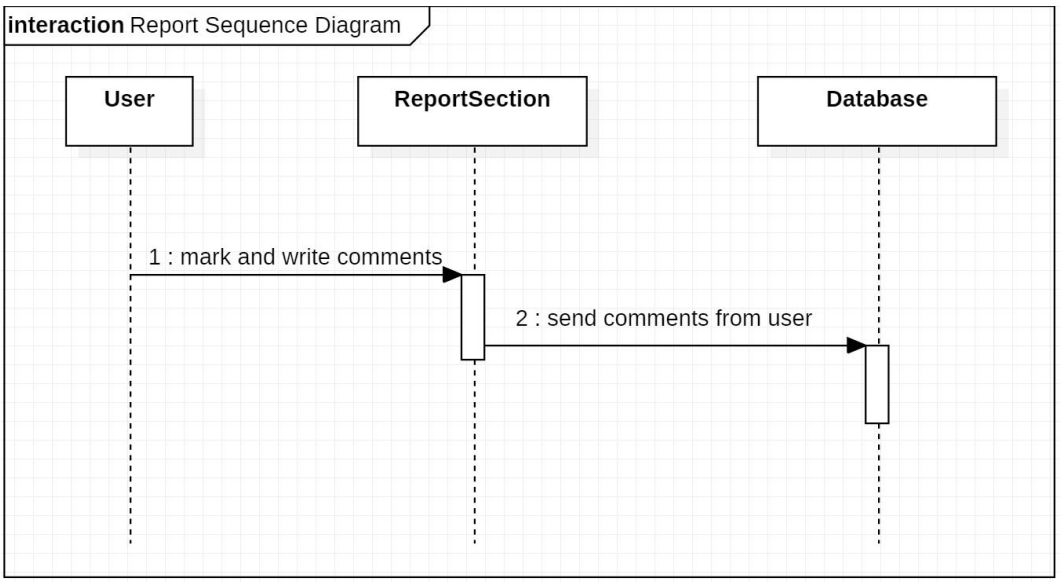

Figure 9: Feedback Sequence Diagram

3.3 Design Class Diagram

We list all the class diagrams for every corresponding package in Figure 10 -14.

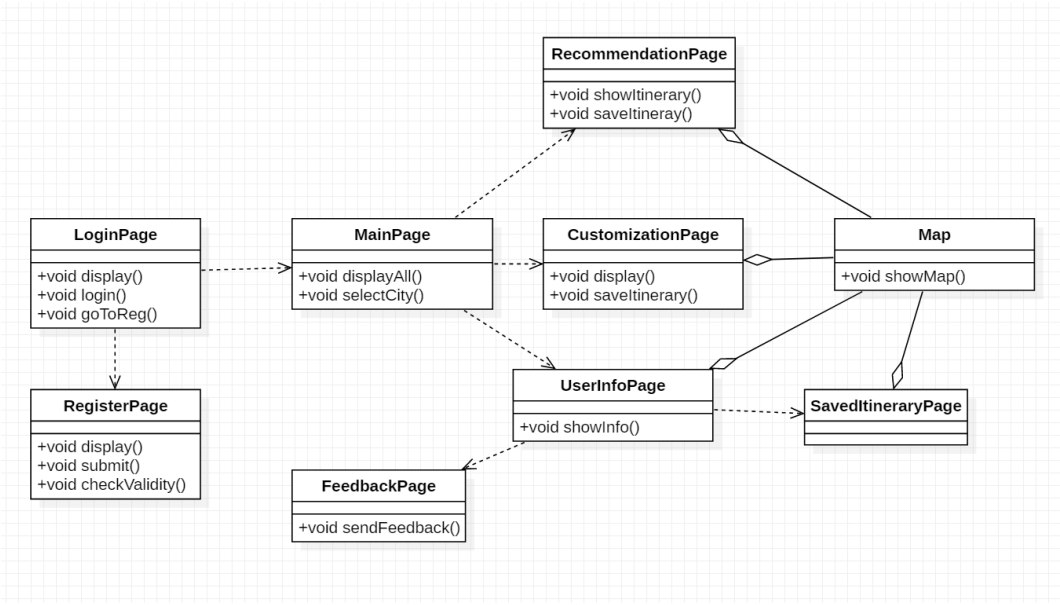

Figure 10: UI Subsystem

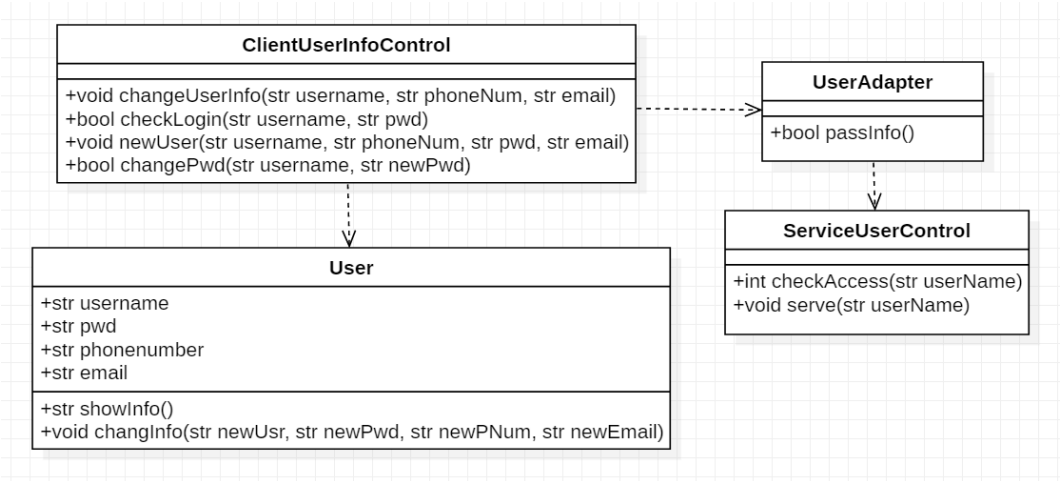

Figure 11: UserManagement Subsystem

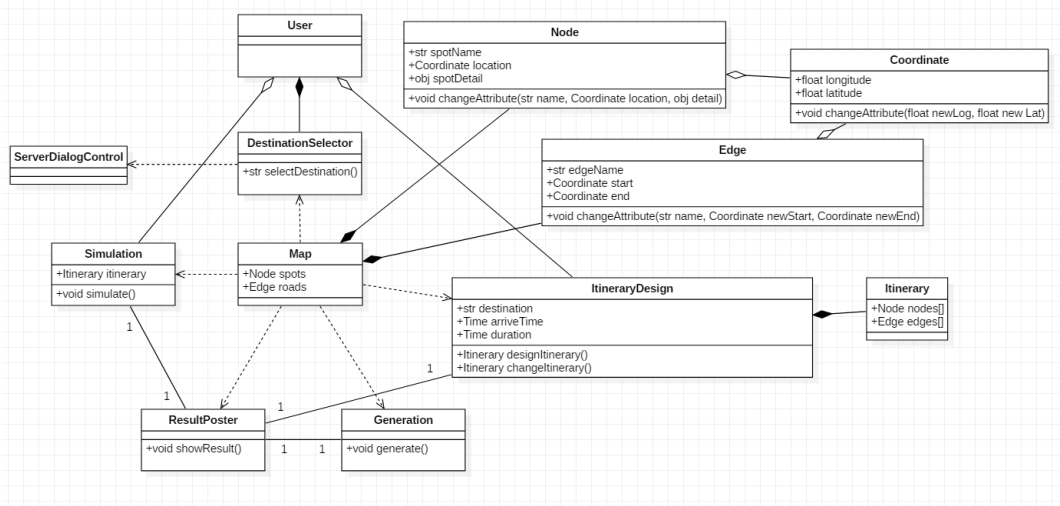

Figure 12: JourneyManagement Subsystem

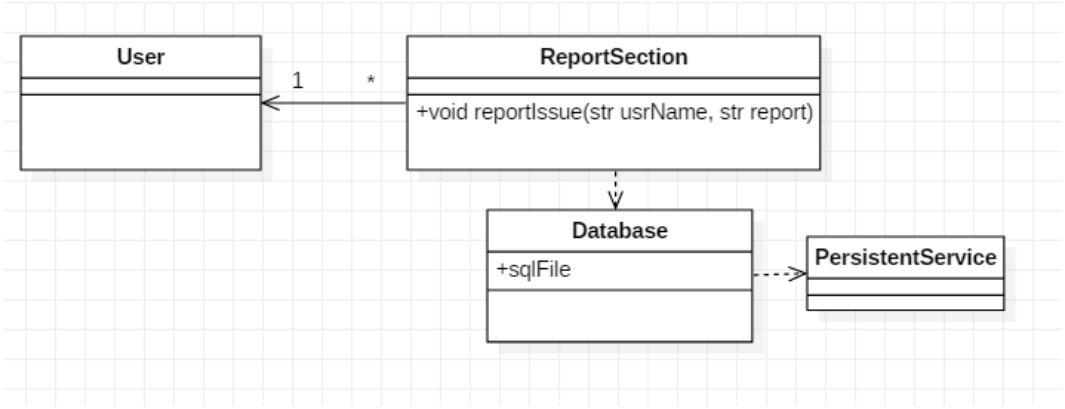

Figure 13: FeedbackManagement Subsystem

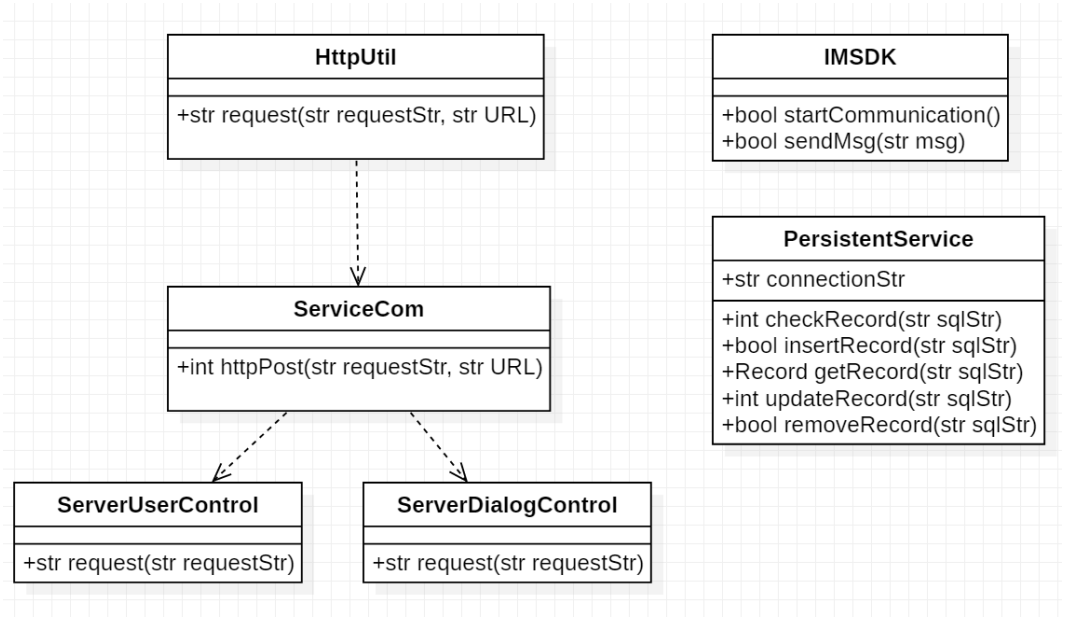

Figure 14: CommonService Subsystem

3.4 Other Diagram

None.

4 Implementation View

4.1 Organization of our Development Model

4.1.1 User Management Subsystem

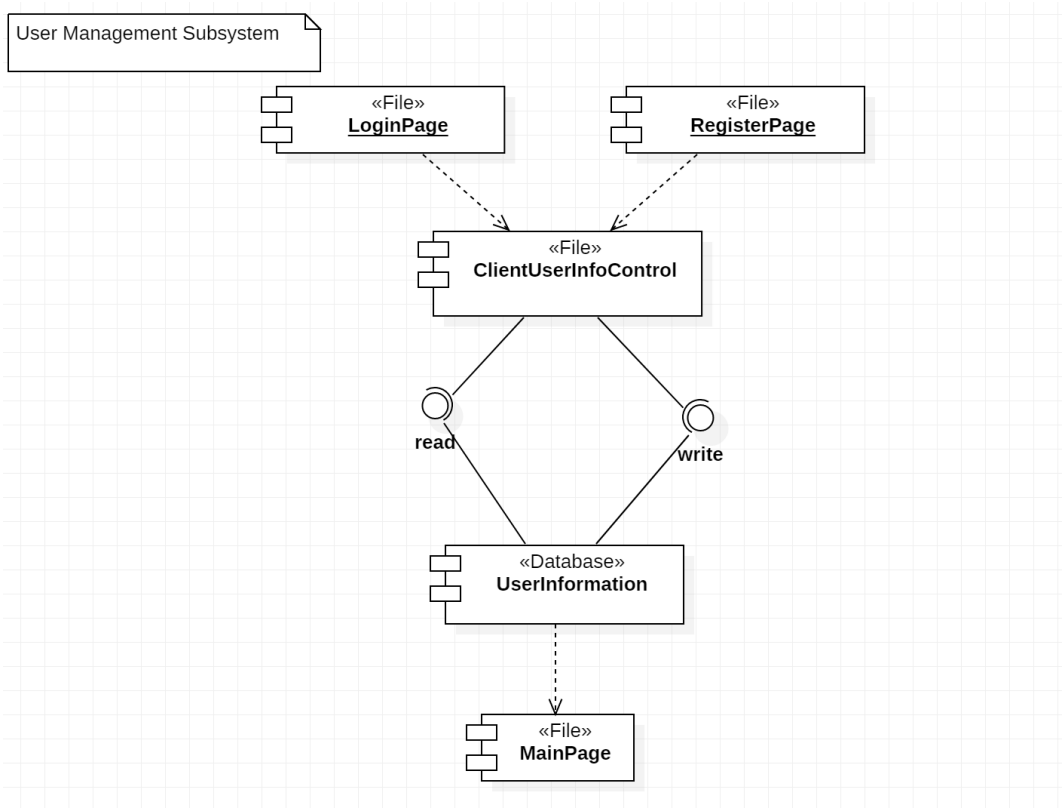

Figure 15: User Management Subsystem

4.1.2 Journey Management Subsystem

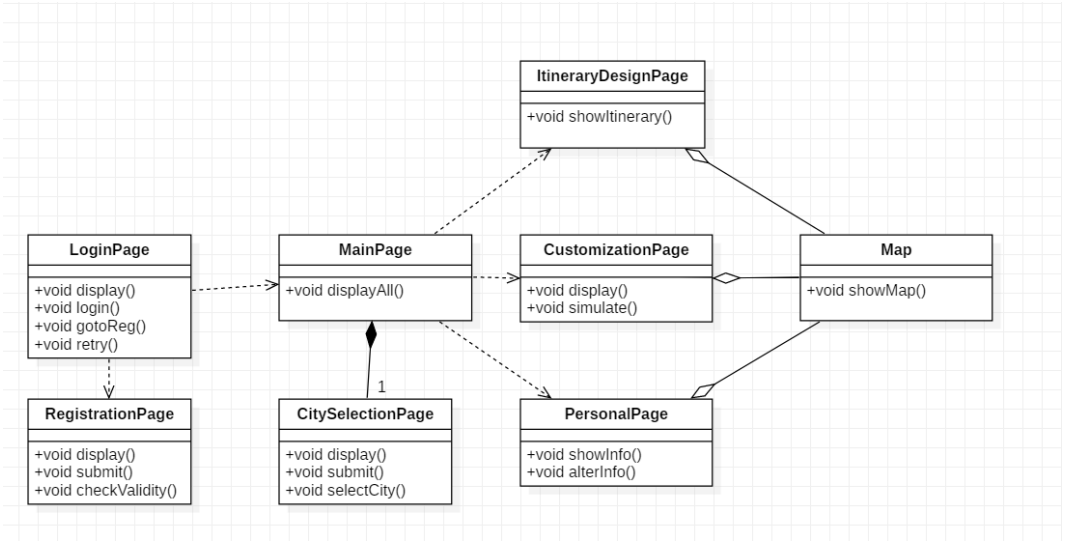

Figure 16: Journey Management Subsystem

4.1.3 Feedback Subsystem

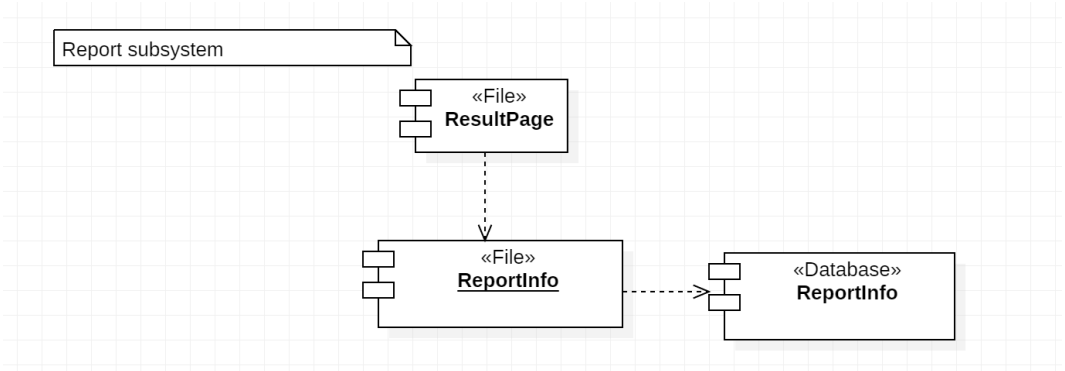

Figure 17: Feedback Subsystem

4.1.4 Commmon Service Subsystem

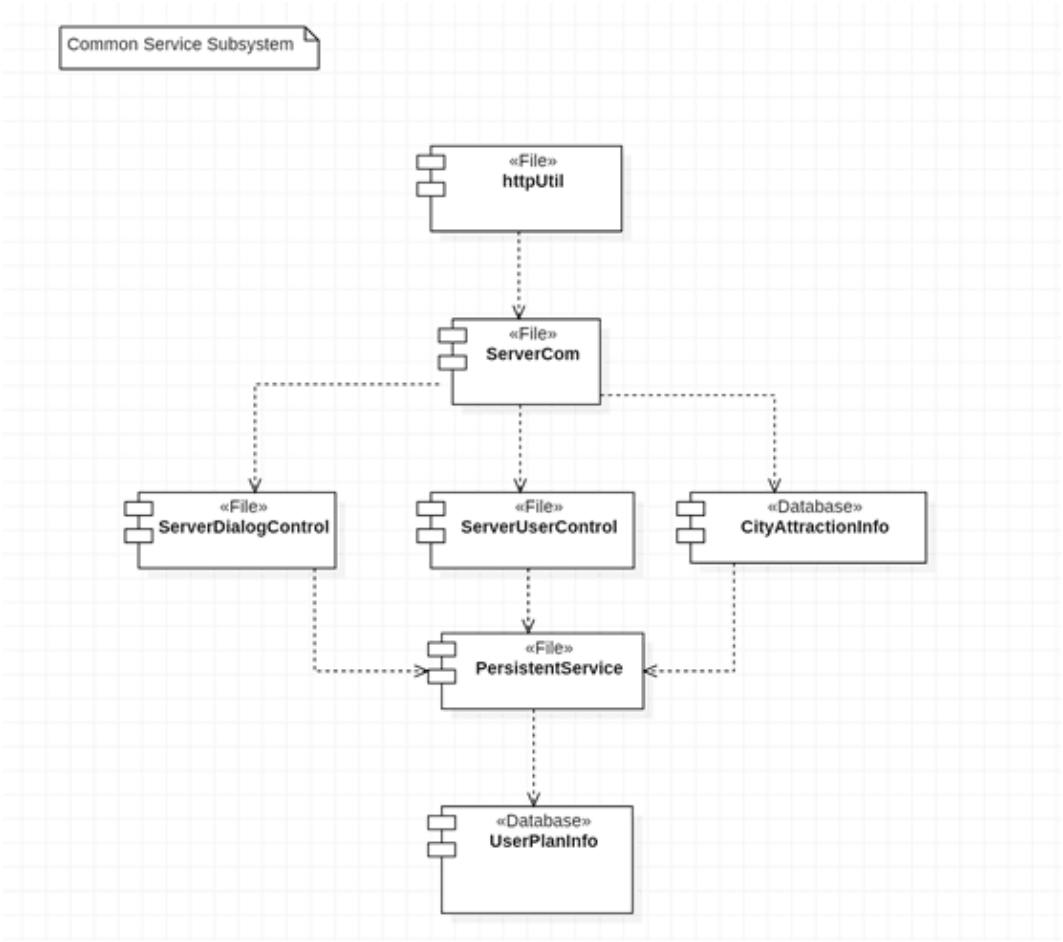

Figure 18: Commmon Management Subsystem

4.2 After Compiling

4.2.1 User Management Subsystem

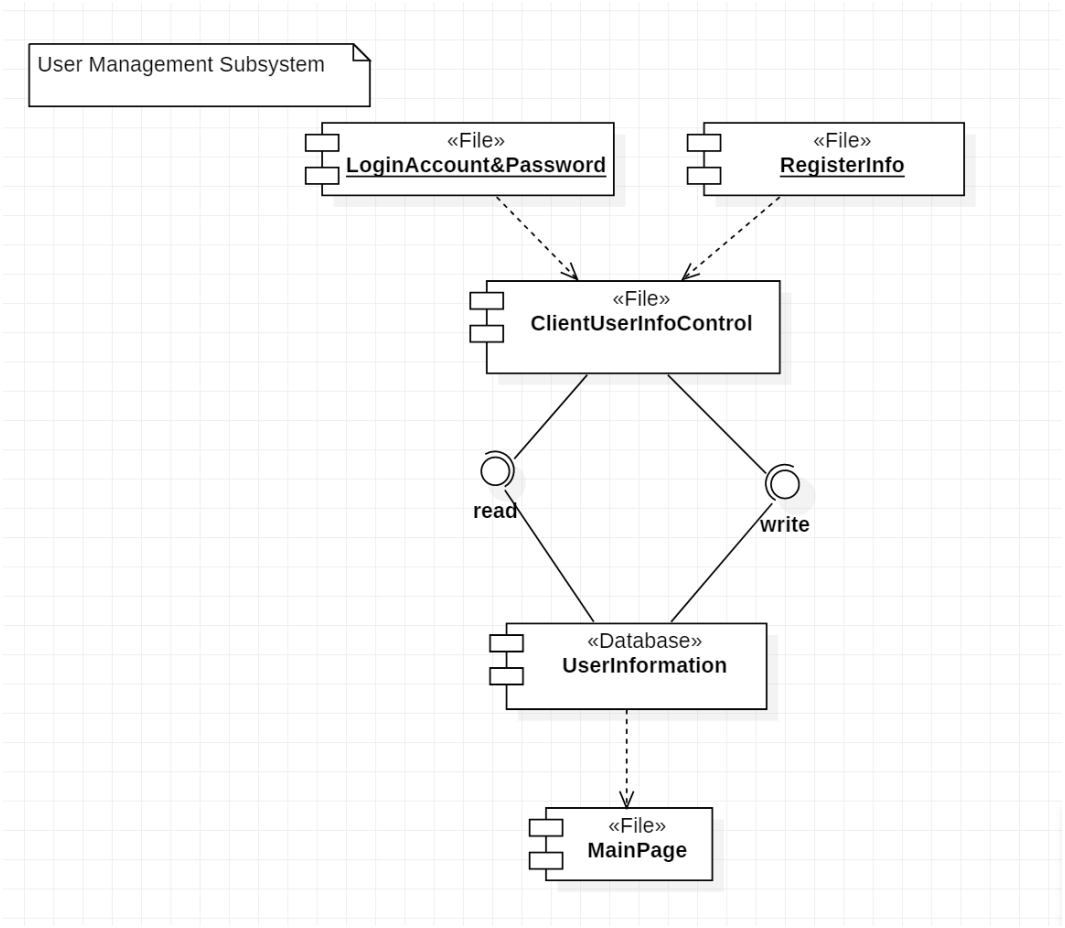

Figure 19: User Management Subsystem

4.2.2 Journey Management Subsystem

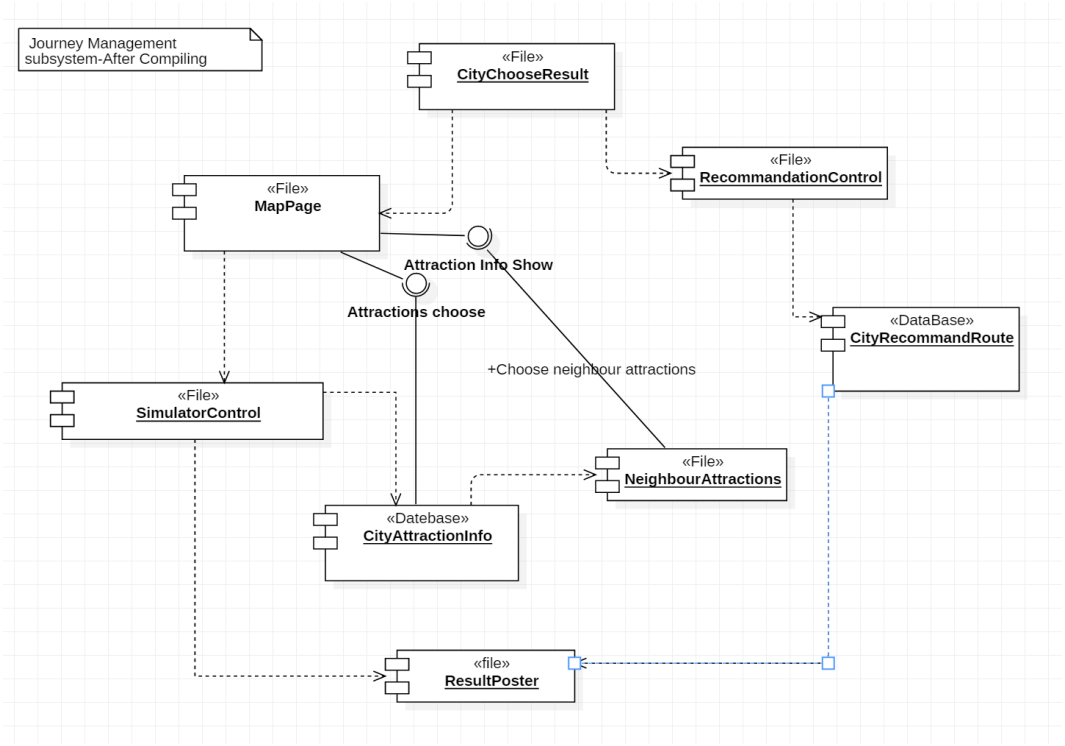

Figure 20: Journey Management Subsystem

4.2.3 Feedback Subsystem

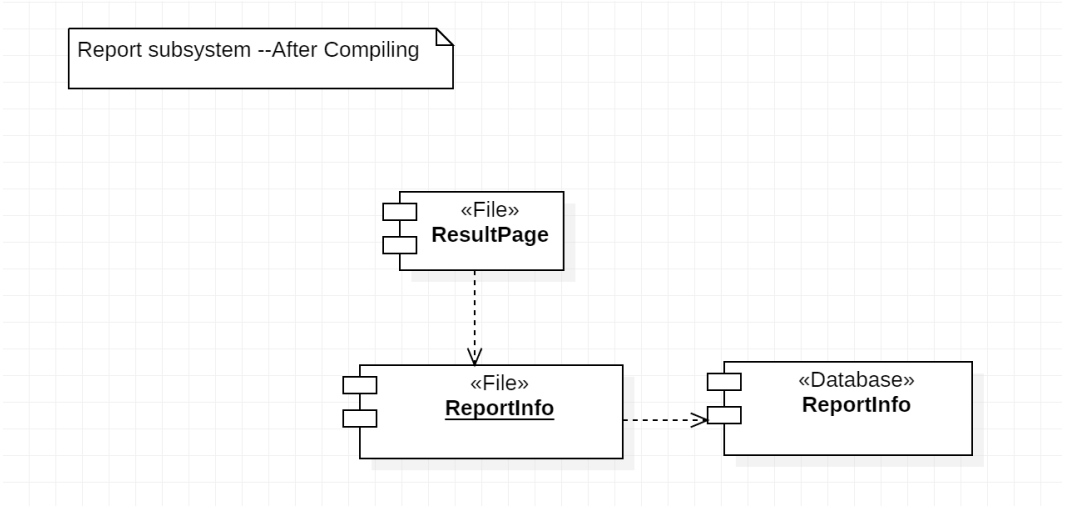

Figure 21: Feedback Subsystem

4.2.4 Commmon Service Subsystem

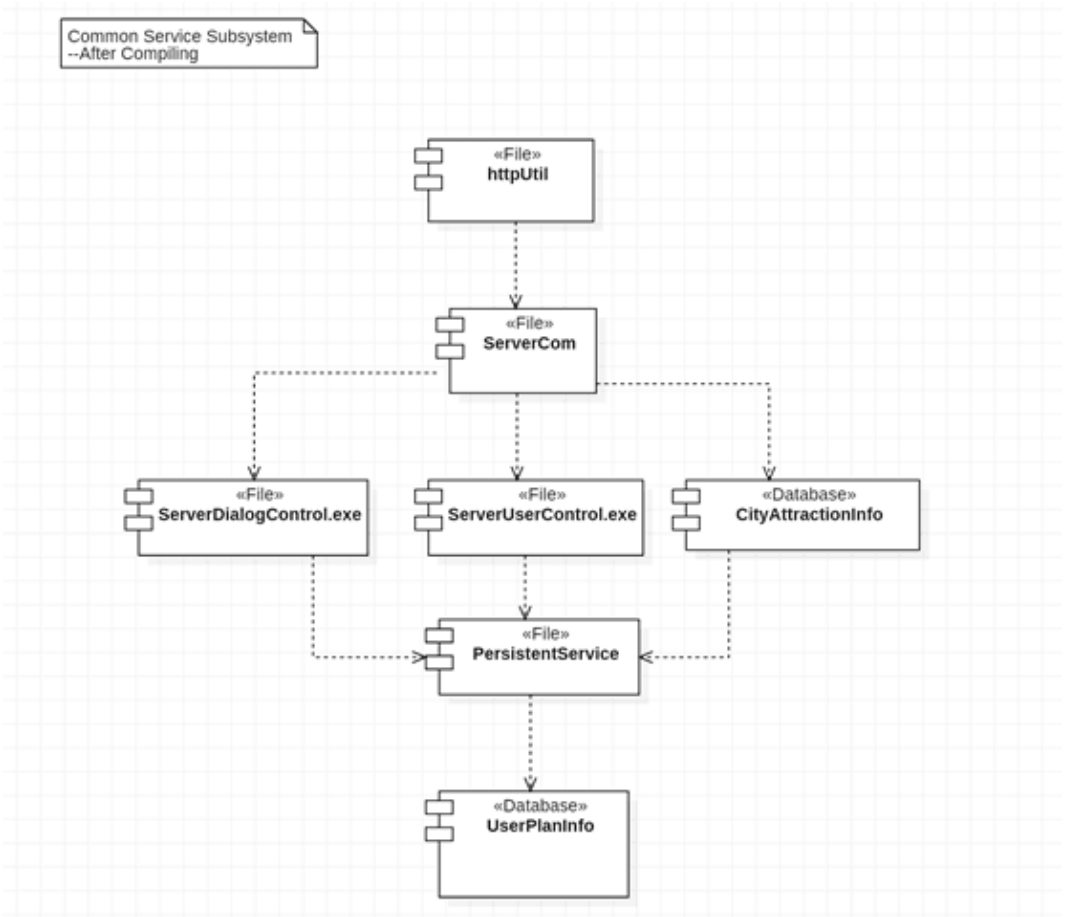

Figure 22: Commmon Management Subsystem

5 Process View

We use class diagram to present process class, thread class, and their interrelation.

5.1 Client Process Diagram

To improve user experiment, UI thread is only in charge of interaction between the user and boundary objects. For every controller and communication module, we use one individual thread. In this case application would not be hindered by frontend and communication tasks.

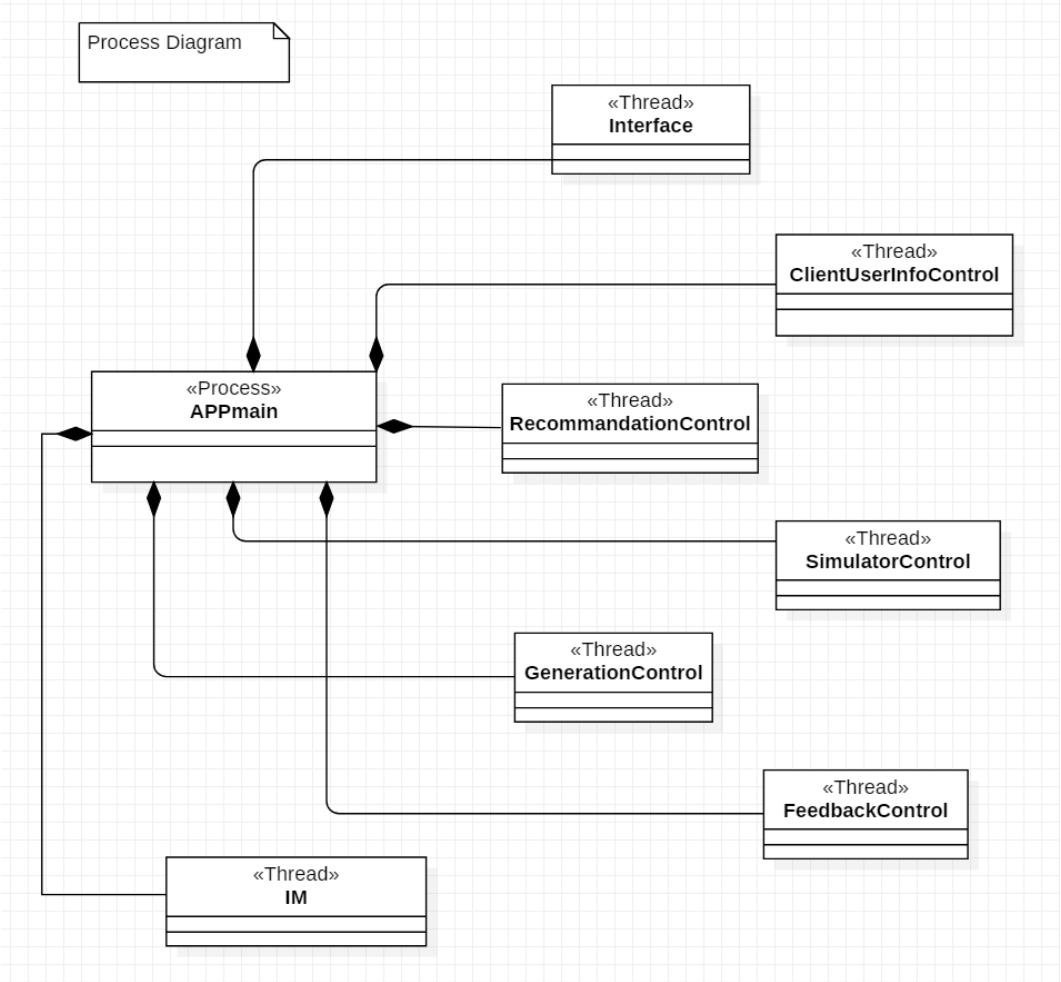

Figure 23: Client Process Diagram

5.2 Server Process Diagram

Because server process is very similar to client process, we omit it here.

6 Deployment View

Since our client app runs on mobile phone, we need the server play the backend role so that data access can be done directly via database server.

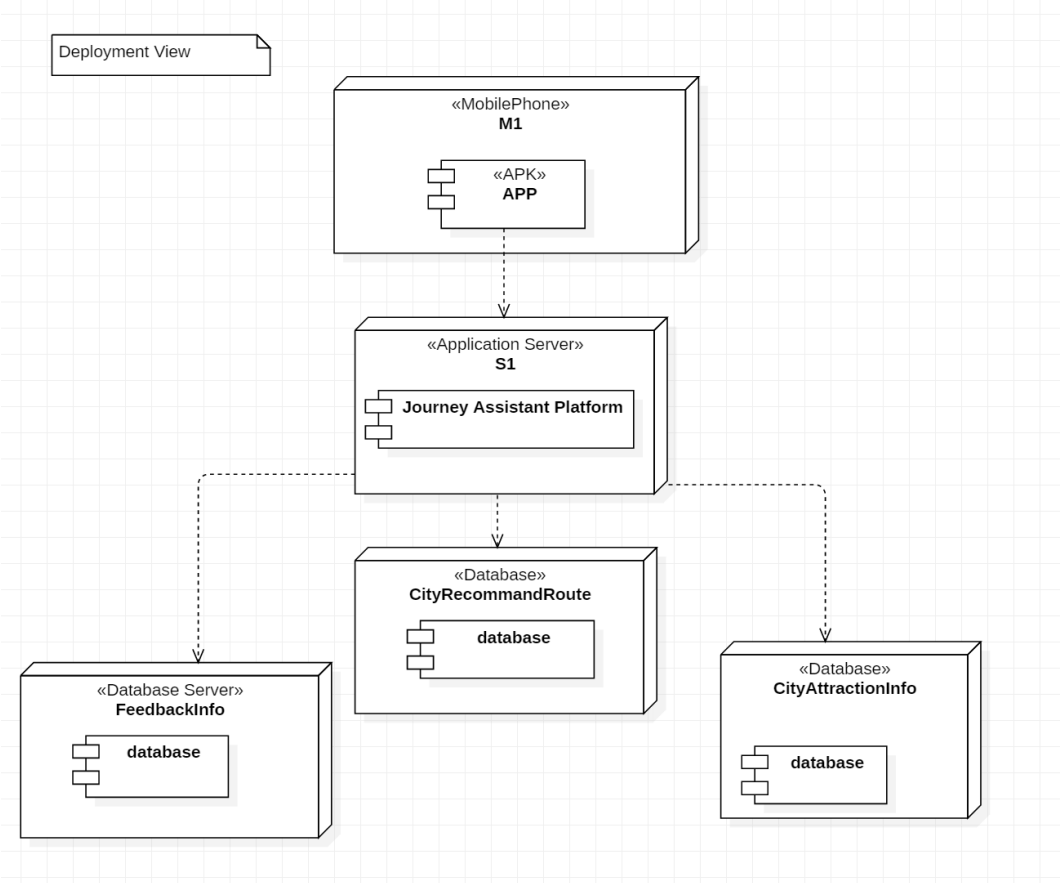

Figure 24: Deployment Diagram