< Journey Assistant >
Software Project Summary Report
Version 1.0

Group Member

Yiwen Song
Zhihui Xie
Weizhe Wang
Huangfei Jiang
Haoping Chen

Modification History

Date	Version	Description	Author
2019-06-18	1.0	Finish the first version.	Haoping Chen

Contents

1 Intruduction

1.1 Purpose

This software orject summary report is aimed at summerizing the whole project of our 'Journey Assistant' App. The summery report will show the final result of our project, and summerize the experience and lessons we have learnt in this project. The document is used for summerize our project and gather experience for future work.

1.2 Scope

Software the document is applied on: Journey Assistant.

Characteristics, subsystems, models and codes related to the software all fit the contents of this the document.

1.3 Definition

The terms referred to in this document are defined in the project glossary document (Glossary.pdf).

1.4 Bibliography

1. <Object Oriented Software Engineering (Version 3)> (Tsinghua University Press)
2. <Object Oriented Software Engineering Practice Guidelines>

1.5 Sketch

This document icludes three parts: actual development results, evaluation of development work, experience and lessons. In actual develpment part, we summerize the product, cost, personnel, and etc of our project. In evaluation of development work, we look back on our development work in many aspects. Experience and lessons parts show the final reflection of our whole group. Different parts of this document complement and reference each other, and jointly shown the total outline of our 'Journey Assistant' App.

2 Actual Development Result

2.1 Product

2.1.1 Android Client Program List

Our Android client program list is shown in Table 1.

2.1.2 Server Program List

Our server program list is shown in Table 2.

2.1.3 Program System Version

The system version of our program is shown in Table 3.

2.2 Main Functionality and Performance

2.2.1 Main Functionality List

Our main functionality list is shown in Table 4.

2.2.2 Main Performance List

Our main performance list is shown in Table 5.

2.3 Basic Procedure

The core use cases are Recommedation Use Case and Customization Use Case. Sequence diagrams of them are shown below in Figure 1 & 2.

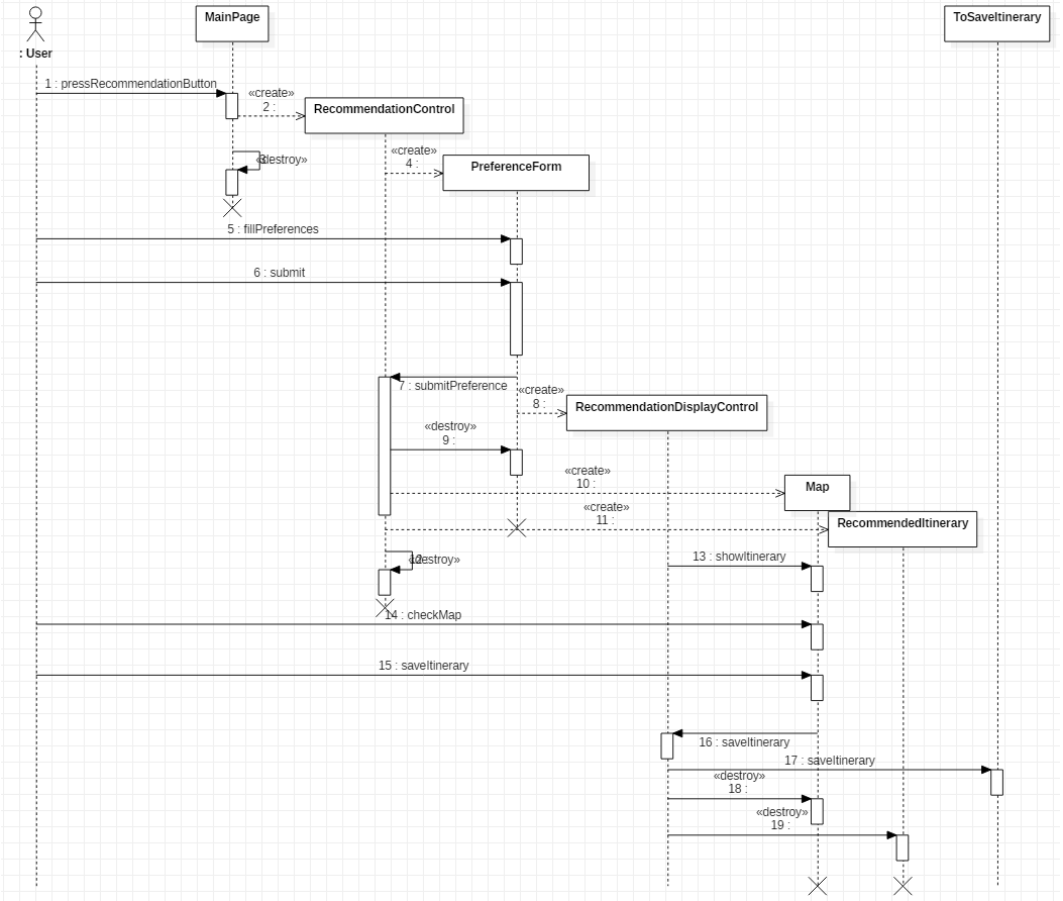

Figure 1: Recommendation Sequence Diagram

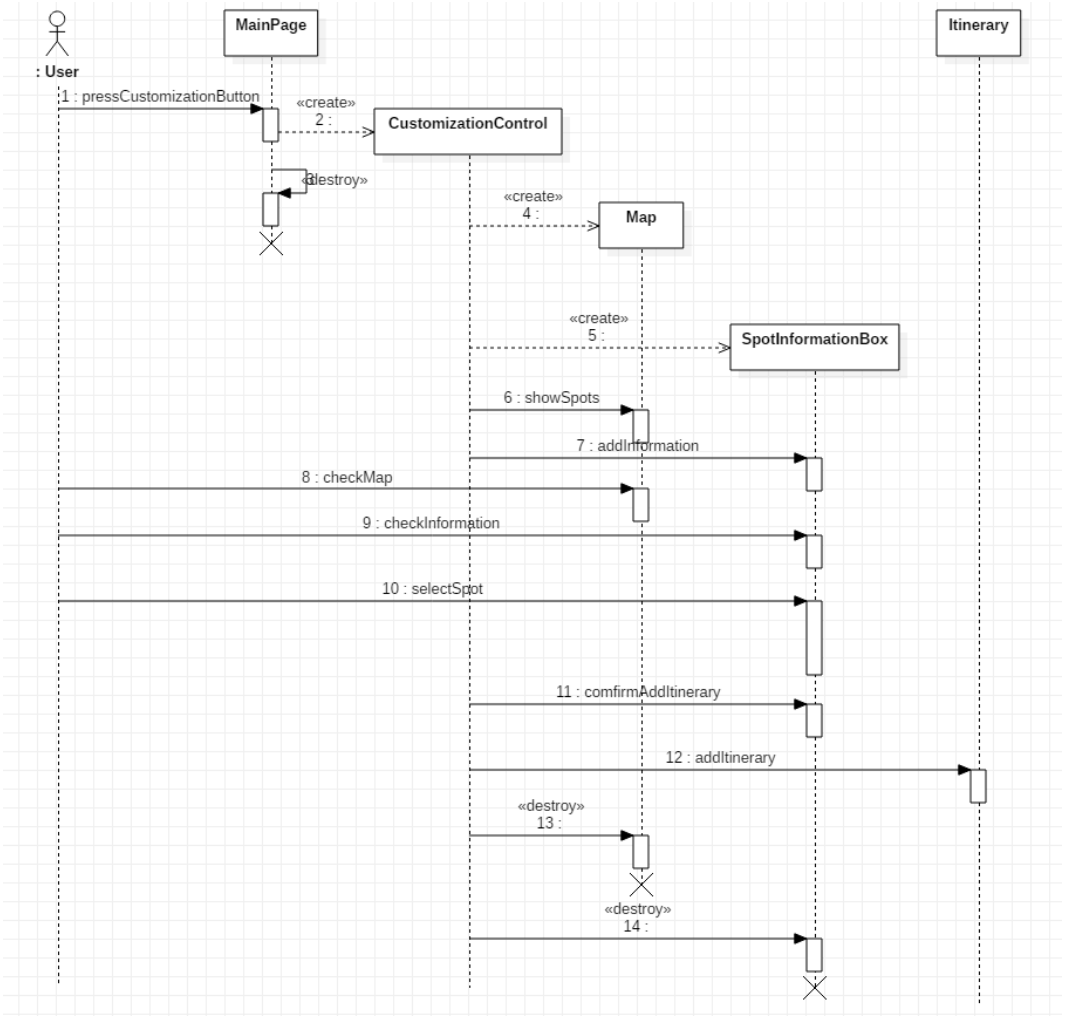

Figure 2: Customization Sequence Diagram

2.4 Progress

Our progress chart is shown in Table 6:

Although it's the first time for us to make the project, we keep pace with our schedule perfectly thanks to our careful plan and nice version control.

2.5 Cost

Since the whole project is carried out by team of students, there's no actual cost.

3 Development Work Evaluation

3.1 Evaluation of Production Efficiency

In contrast to our origin plan, the actual production efficiency is even higher given such limited time.

3.1.1 Actual Efficiency

- 3 months for system development
- Some repetitive operations
- 1200 lines of code in program per month per group member
- 4000 words in documentation per month per group member

3.1.2 Origin Plan

1000 lines of code and 3000 words per month per group member

3.2 Evaluation of Production Quality

BUG occur per 200 lines of code on average, the error rate is within estimate rate.

3.3 Evaluation of Technique

Github for Version Control It is helpful for cooperation of our group.

Client-Server Design Mode Front end is Android App, back end is Tomcat Server and SQLite3 database. This design mode accords with App development environment and is easy to manage.

MVC Design Mode Disassemble the system into Model, View and Controller. It is very common in Android development.

Three Layers of System Architecture

User Interface Layer UI of App

Application Logic Layer control the realization of functionalities

Storage Layer responsible for data storage, retrieval and searching

API Usage The usage of API enriches the content of our project.

- Baidu Map API for displaying recommendation result and trip customization
- Meizu Weather API for getting weather information.

Markov Decision Making Process in Trip Recommendation It's an effective way to recommend different trips based on user biases.

3.4 Error Analysis

- When Android Studio can not compile the project, the error is that the gradle version is not the latest, and also dependencies are not added in the gradle.
- When crash happens on the App, the reason is that there's something wrong with Intent communication

4 Experience and Lessons

4.1 Experience

4.1.1 Schedule

We should foresee the problems we may confront when making plans, and reserve time for each stage at the beginning.

4.1.2 Requirement

User requirements should be fullycollected so that we can make better analysis and development.

4.1.3 Design

Use UML in project design, which provide perfect guiding function.

4.1.4 Technique

Use of different APIs enrich the content of our project.

4.2 Lessons

What we did not do very well is version control. Although we use Github to help control our version, we sometimes did not update our codes in github, which brings out the problem that one file may be modified by more than one group member, adding to integration workload of our project. So wo should commit our change in github once we have made some changes.

Package Name	Program Name	Size(KB)
com.example.travelingagent.activity	CheckItinerariesActivity.java	4
	CustomizationActivity.java	23
	FeedbackActivity.java	13
	Itinerary.java	2
	LoginActivity.java	8
	MainActivity.java	16
	RecommendationActivity.java	4
	RecommendationDisplayActivity.java	17
	RegisterActivity.java	6
	SavedItineraryDisplayActivity.java	15
com.example.travelingagent.entity	Hotel.java	1
	Sight.java	1
	Spot.java	3
	User.java	2
com.example.travelingagent.protocol.api	CustomizationClientApi.java	1
	ItineraryClientApi.java	1
	LoginClientApi.java	1
	RecommendationClientApi.java	1
	RegisterClientApi.java	1
	WeatherClientApi.java	1
com.example.travelingagent.protocol.entity	ItemEntity.java	3
	LoginEntity.java	1
	ModeEntity.java	1
	RegisterEntity.java	1
	WeatherEntity.java	1
com.example.travelingagent.util.adapter	FoldingCellListAdapter.java	6
	ItineraryRecyclerAdapter.java	6
	ModeAdapter.java	2
	RecyclerAdapter.java	6
	SpotAdapter.java	4
com.example.travelingagent.util.easyFeedBack	EasyFeedback.java	2
com.example.travelingagent.util.listener	ItemClickListener.java	1
com.example.travelingagent.util.model	DataBean.java	3
com.example.travelingagent.util.viewHolder	BaseViewHolder.java	1
	ChildViewHolder.java	2
	ItineraryParentViewHolder.java	4
	ParentViewHolder.java	4
com.example.travelingagent.util	ReadFile.java	2

Table 1: Android Client Program List

Package Name	Program Name	Size(KB)
com.test	GetItinerary.java	4
	Gethotel.java	5
	Getsight.java	4
	Graph.java	4
	Hotel.java	2
	Itinerary.java	2
	Login.java	3
	Recommendation.java	5
	Register.java	3
	ReportMSG.java	3
	SaveItinerary.java	4
	SendItinerary.java	5
	Sight.java	1
	Simulation.java	4
	Spot.java	3
	Testjava.java	3
	Type.java	1
	User.java	1

Table 2: Server Program List

Program System Name	Version	Description	Edit Time
Journey Assistant	v1.0	Realize core use cases	5.17
	v2.0	Completely finished and pass the test	6.17
Journey Assistant Server	v1.0	Completely finished and pass the test	6.17

Table 3: Program System Version

Main Functionality	Development Goal	Comment
Register	Realized	
Login	Realized	
Select Destination	Realized	
Set Preference	Realized	
View Map	Realized	
Recommendation	Realized	
Customiztion	Realized	
Check saved itineraries	Realized	
Feedback	Realized	

Table 4: Main Functionality List

Main Performance	Development Goal	Comment
Response Time Requirement	Realized	The average response time is under 0.4s.
Throughput Requirement	Realized	Under 2500 requirements per second.
Capacity Requirement	Realized	The maximum number of users and itineraries is around 140000.
Resource Requirement	Realized	The number of items in database is under 500000, the memory usage is no more than 300MB, and the bandwidth server needs is around 5Mbps

Table 5: Main Performance List

Milestone Events	Expected Deadline	Actual Finish Date	Schedule Variance
Software Requirement Specification Finished	4.20	4.15	5 days ahead
Software Architecture Document Finished	4.20	4.17	3 days ahead
Module Development Finished	5.18	5.16	2 days ahead
System Integration Finished	5.18	5.18	Exactly on time
System Test	6.18	6.17	1 day ahead
Whole Project Finished	6.18	6.17	1 day ahead

Table 6: Progress Chart