< Journey Assistant > Software Requirement Specification Version 1.0

Group Member

Yiwen Song
Zhihui Xie
Weizhe Wang
Huangfei Jiang
Haoping Chen

Modification History

Date	Version	Description	Author
2019-04-02	1.0	Finish the architecture of the document.	Huangfei Jiang & Weizhe Wang

Contents

1 Intruduction

1.1 Purpose

This is a requirement specification document. In this document, we will define most of the system requirement, so that all the develop team member can have a clear picture of the whole system. We define not only the functional requirement, but the non-functional requirement as well. This is the main document of this define phase, in this phase, we also have glossary document, use case specification document and we offer a prototype.

1.2 Scope

This document is applied to our Journey Assistant System. It is a specialized document for our product.

1.3 Definition

Abbreviation	Term	Implication
JAS	Journey Assistant System	Our Proposed System
	Functional	Some requirement that need to be realized
	Non-functional	Some requirement that can not be realized but is indispensable to our system
GAN	Generative Adversarial Network	

1.4 Bibliography

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

1.5 Sketch

In this document, we are going to make some detailed descriptions on system requirements and related models and recount them by the order of function.

2 System Overview

Nowadays, people's life has become better and better because of the reform and opening-up policy. More and more people decide to take a trip instead of staying at home during the holidays. However, since the destination is new to them, people usually can't come up with a nice plan, which will influence their journey experience greatly. In the past time, people always search for different kinds of strategies and then knock them together. On the other hand, people who are busy studying and working may not have so much time to have a dreamful trip. With great pressure surrounding, a desired trip is unavailable to them.

Therefore, a network-based journey assistant system JAS is being called out. The traveler can plan their trip with this system handily, and of course get the help anytime they want. Our application will provide user with a platform which can assist them to draw up a perfect plan. Additionally, the platform is both professional and entertaining. With this application, user can not only get a detailed and highly-recommended journey plan but also have a virtual journey and feedback as they want. It will help people to fulfill their travel experience.

We have following functions.

Make Plan on City After user's selecting a certain city, system will produce a travel plan according to user's different requirements.

Simulate a Trip Users can select scenic spots one by one, then system will simulate a trip and give user a fact-based experience feedback.

Give a Feedback Users can report the opinions about our system or their usage experience to us by our reporting interface.

Other Non-functional Requirement

1. The data of users should be protected against vicious hacking.
2. Simulation should be close to fact as much as possible.
3. Normally, the response time of our system should not be longer than 30s.
4. The UI should be beautiful and activate user's desire to have a trip.
5. Statistics must be made.

3 Targer System

3.1 Sketch

Our project includes functional requirements and non-functional requirements. Functional requirements can be summarized as making a plan and simulation of tourism, while non-functional requirements consist of response time, security, apperance and so on.

3.2 Functional Requirement

JAS provides users a handy tool to organize and manage their journey according to their kinds of requirements.

First of all, it can help normal users to know more about the destination, such as climate, traffic situation, civilization and so on. After knowing about different places, users will make their mind to choose one or more city as their destination. Then users will get into a filter interfaces. In this interface, users can add conditions as they like which will be used to filter the plans in our alternative set. For example, the number of travelers, whether they have a pet or not, the style they want, the budget and any other requirements. All these demands will be considered in our next plan recommending.

During selecting condition, in order to produce a accurate and practical plan, system will provide users with different choices instead of letting them enter demands arbitrarily.

In addition, JAS also offer a powerful tool to simulate a real trip and give users a feedback about experience.

Sometime users may not have time to have a trip, however, they still want to get some experience. With our application, time and money will not be constraints to our users anymore. They can go through their trip virtually, while our system can give them a real experience.

When users try to enjoy our simulation service, they also need to select a city at first. After city is decided, users can begin to choose the scenic spots. Each choice will be sent to the backend. After a nice and related cartoon, system will give a real feedback about this travel experience. Of course, feedback won't be the

same all the time, since we add some related random incidents to it. For example, if you choose a college as your destination, you may have a special experience with cute cats living in it! About the feedback, we will provide three types: some sentences (more sentient), some words (more accurate) and GAN's perspective (Generative Adversarial Network).

As a wonderful application, the feedback of users can't be ignored. With a report interface, we can learn about our drawbacks and advantages in time.

Report will be conveyed to us through backend. When users begin to write a report, there will be two main types to choose: about data and about usage experience. Different types of report will be sent to different developers.

3.3 Non-functional Requirement

1. Special security should be taken because the data of users cannot be leaked.
2. As a featured function, simulation must be as real as possible, since this is why users choose our application.
3. Response time should be as short as possible and had better not be longer than 30s. A loading flash should be displayed when users are waiting.
4. Some data of users should be accessed so that we can make a better and personal recommendation.
5. Users interface should be attractive, some related information can be displayed at first.

3.3.1 Availability

1. Since our application is a handy application, there is no requirement for users themselves.
2. Our application need to be updated periodically with normal network.
3. Two huge database should be equipped.
4. Network should keep normal when users use our function.

3.3.2 Reliability

1. The system will be closed if we need to update or repair it. As for the other time, users can have a nice experience as they want.
2. To keep our application advance with the times, we need to update our database per month.
3. The information should be real absolutely. Description and real experience should be corresponding.

3.3.3 Performance

1. The system response time (response to users' request from servers to clients) must less than 30 second, if the network state is normal.
2. Our system can deal with 1000 cases simultaneously.
3. The capicity depends on our database and related arithmetic.

4. If the network state is abnormal, we can only provide users with off-line mode. In this mode, users can still search for information about scenario from local database.
5. To achieve short response time, we need to have a powerful CPU or server to compute rapidly. However, to users, a well network state is the most important.

3.3.4 Supportability

1. Our code standard and name rule will obey international rule strictly.
2. Only developer can visit our back-end directly.

3.3.5 Design Constraint

1. progamming language:Java, python, html and php.
2. Java for main code, html and php for front-end, python for arithmetic.

3.3.6 Interface

Detailed interface is to be determined.

3.3.7 Law, Copyright and Other Notice

1. This software is created by all us group members together. We, unlearninng group, reserve all the right for the final explanation.
2. Journey Assistant is a temporary name for our app, if it causes trademark infringement, welcome contact with us.

3.3.8 Applicable Standard

1. We are decided to produce an andorid software, hence it should obey the standard of andorid.
2. The main language of our app is Chinese, English version is to be determined.

3.4 System Model

This section express detail requirements with concepts, methods and model diagram in UML.

3.4.1 Scenery

In this section, a representative scenario is selected to describe each participant that initiates a use case. Follow the format below:

Scenery One

Scenery name We should select a Specific name for each scenery.

Actor instance The specific actor involved in the scenario

Event flow Follow the steps to list the detailed flow

3.4.2 Use Case Mode

The specific steps in this section are as follows:

(a) Participant

Participants Name	Explanation
User	The user who uses this software

(b) Use case

Login	Sub-function Level	For identity confirm.
Select Destination	Sub-function Level	For the software to find corresponding city information.
Itinerary Recommendation	User Target Level	In this section, user are recommended to several packaged itineraries and he can select one of them.
Simulation	User Target Level	
Generation	User Target Level	
Feedback	User Target Level	

(c) Introduce each use case in detail.

Login

Preconditions and Post Conditions

- There is no preconditions for this use case.
- When the user's input has been identified, the system will offer the access to the services, and the use case is over.

Basic Scenarios

1. The user inputs his username and password, and submit them to the system.
2. The system checks the validity of the input. If the username and the corresponding password can be found in the database, the system will give out the destination-selection panel.

Exception or Branches If the input is not valid by the validity check, the system will give out a warning and the user should try it again.

Select Destination

Definition This is the requirement description for the Select Destination use case. Select Destination use case is for everyone who wants to make their journey plan or have simulated journeys. The user can only select destination cities from a list we give, which we have their information in our database. Input a city that is not recorded is not permitted.

Preconditions and Post Conditions

- The user has successfully logged in to the system.
- After the user has selected a destination city, he/she can make journey plans in the city or have simulated journeys.

Basic Scenarios

1. The system displays a list of cities that are available.
2. The user selects a city he/she wants to go.

Exception or Branches After the user select one city to go to ,he could then choose which way he wants to pick up his travel itinerary.

Make Plan

Definition This is the requirement description for the Make Plan use case. Make Plan use case is for every user who wants the system to recommend a journey plan in a specific city. The system will recommend several journey plans for the user.

Preconditions and Post Conditions

- The user has successfully logged in and selected the destination.
- The system provides the user with several recommended journey plans.

Basic Scenarios

1. The user inputs his preference of the journey, including the rythm, cost, and number of persons.
2. The system makes several plans according to the user's preference, by our recommendation algorithms.
3. The user picks one of the journey plan and saves the plan in his account.

Exception or Branches If the user does not want any of the journey plan, he can click the "no wanted" button, and the system will recommend another journey plan for him.

Simulation

Definition This is the use case of simulation part, which is also the core part in our travel assistant. Users can choose scenic spot one by one on our simulation map, according to the spots information we posted on it.

Preconditions and Post Conditions

- (a) The user has decided which city he preferred to go to and has confirmed the arriving data.
  (b) A map with every attraction of this city should be available to the user.
  (c) Each attraction has detailed information on the basic introduction, recommended spending time, the specialty and others' comments, etc.
- When the user finished the simulation and clicked a "finished" button, this use case is over and will turn to the generation part.

Basic Scenarios

1. User enter the simulation interface.
2. User glance over all of the attractions on the simulation map
3. Therefore, can user choose the first attraction of the city on our simulation map.
4. The basic information of attractions nearby will be output automatically in the form of card.
5. User could choose to select the nearby attraction or other attractions as he wanted to be the next station.
6. Then user will repeat step d and e until he want to put an end on this journey.
7. User click the "finished" button and finish this use case.

Feedback

Definition This use case is to describe the feedback from the users who have finished their authentic journey and wanted to make a comments or post a score on this journey. And it also basically includes the ordinary usage of feedback, which includes delivering some suggestions on our software or making some complains on our service.

Preconditions and Post Conditions

- The user must have finished their journey in real time. And they should have a general impression on each place they have been to.
- No post condition.

Basic Scenarios

1. User confirms the finish of the real time journey, which means that clicking the "Journal Finished" button on the travel post generated before the real time journey.
2. User could make a score on this journey and have some general comments on it if he like.
3. Then the scenic spots list will be automatically unfolded under the travel post.

4. t is encouraged for user to leave a message on each of the spots, in order to have a conclusion to themselves as well as leaving some advices for other users.
5. When user have finished this feedback, validation box will be popped up and user could make a tick on the check box to decide whether others can see this journey.
6. Therefore, the score it marked and comments they made will be uploaded.
7. User could review their journey and others can find see it and give a like to it.

Exception Scenarios If user have something else to concern about or lose the patience during scoring. A message box would pop up to ask users whether to contain and upload these message which has been made or quit it.

Generation

Definition This use case allows admin to confirm the final travel plan and share it with others. After the admin has chosen plans we suggested or simulate by himself, he can generate a result in the form of lists or posters. Therefore, he could use it conveniently during the tour or share it with their companions or other friends before the tour.

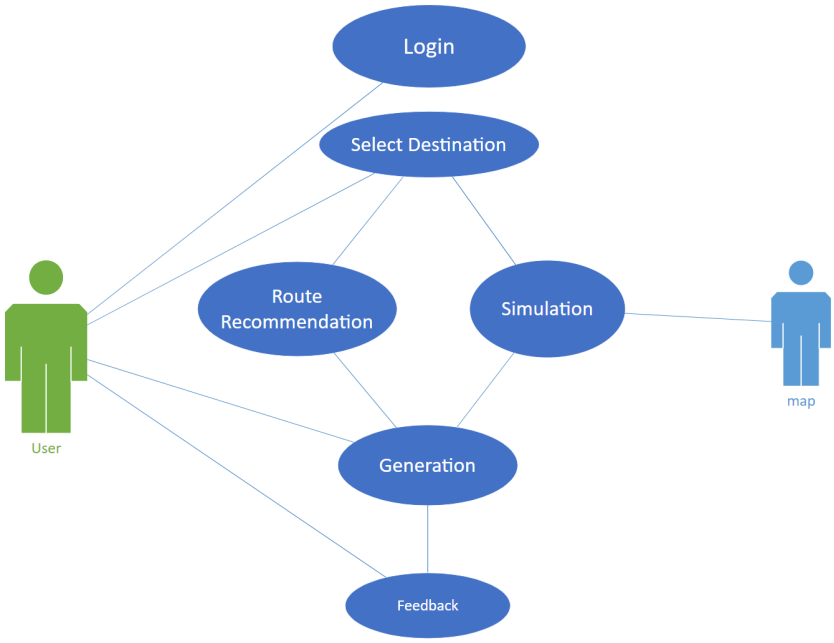
Preconditions and Post Conditions

- (a) The arriving date to the destination city, the schedule of all scenic spots including expected tour time and the time spent on road from one spots to another.
  (b) All information of the scenic spots, including the basic introduction, recommended spending time, the specialty and others' comments should be known.
- No post condition.

Basic Scenarios

1. User enter to the Generation interface.
2. A post will be generated.
3. User could click the "share" icon to share it with others (such as companion or other friends).

Use Case Diagram



3.4.3 Object model

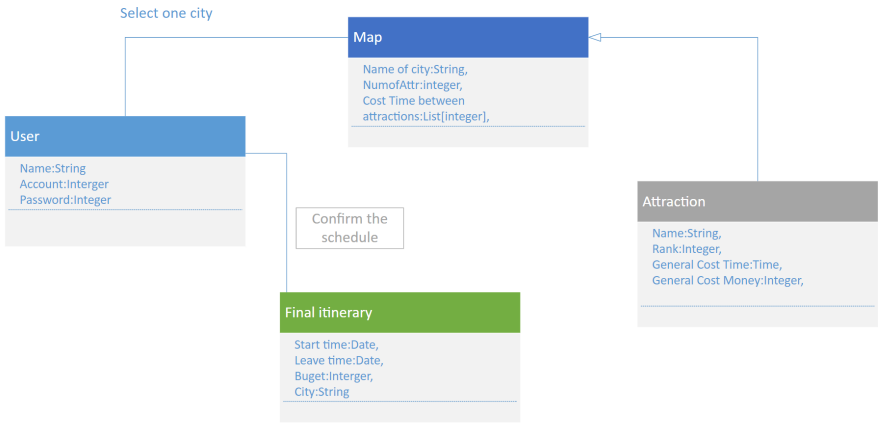
Entity Class Definition Table

Entity class Name	Attributes	Association class	Definition
User	Name:String Account:Interger Password:Integer	To Map: Select one city; To final itinerary: Confirm the schedule	The user of JAS
Attraction	Name:String Rank:Integer General Cost Time: Time General Cost Money:Integer	None	Attractions available in map
Map	Name of city: String NumofAttr: integer Cost Time between attractions: List[integer]	From User: Select one city	The visualized map on which user could select attractions
Final itinerary	Start time: Date Leave time: Date Buget:Interger City:String	From User: Confirm the schedule	List the plan of this travel

Table Boundary class definition

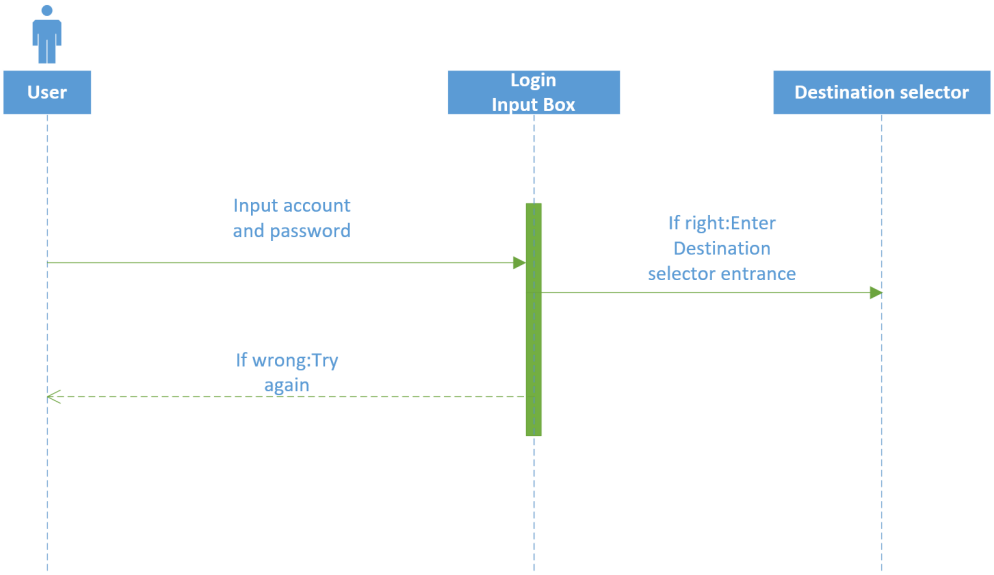
Boundary class name	Definition
menu	The menu which user could choose

Class Diagram



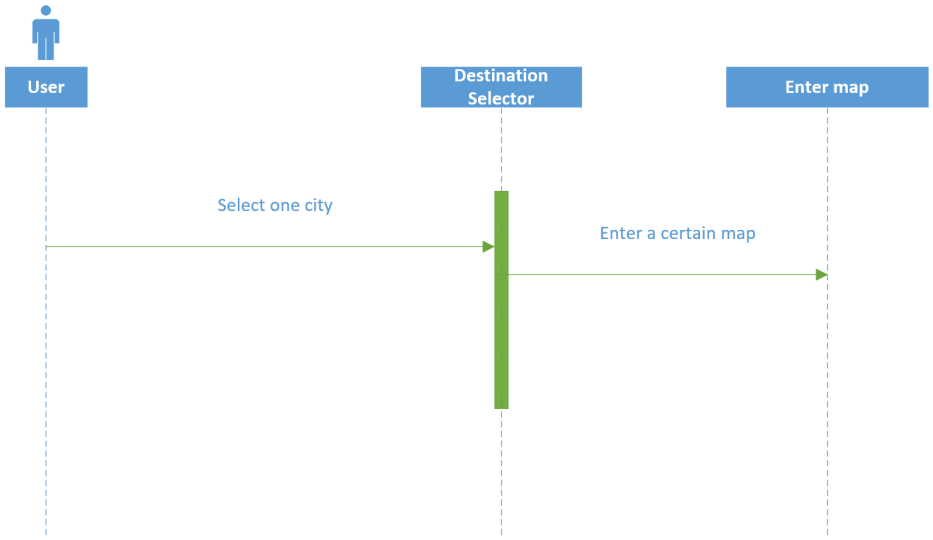
3.4.4 Dynamic model

Login Sequence Diagram

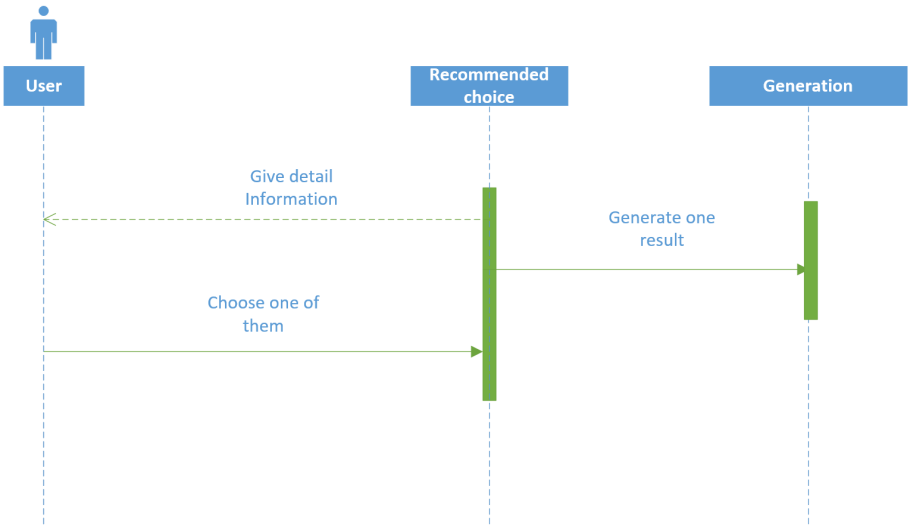


Select Destination Diagram

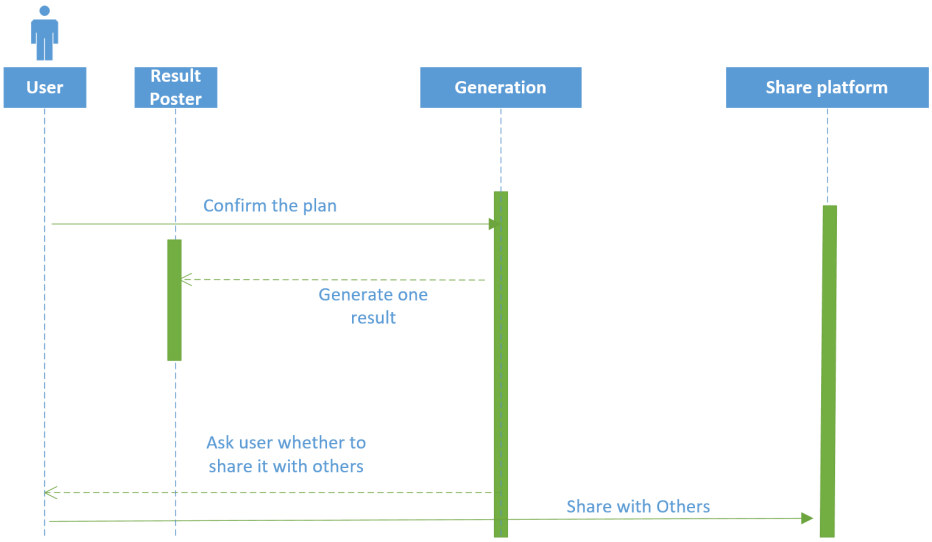
SelectDestination Sequence Diagram



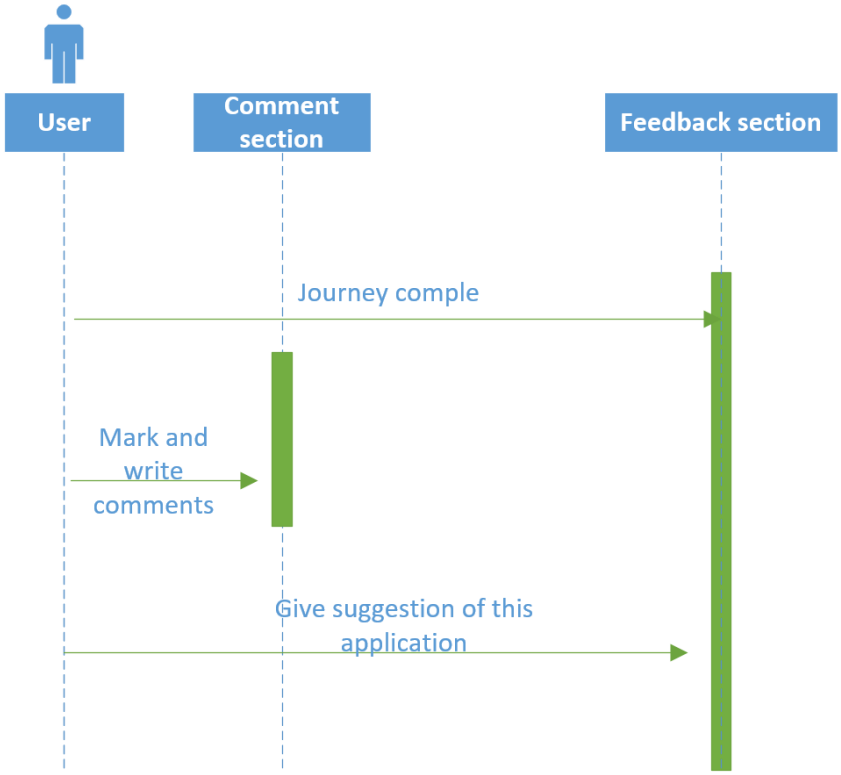
Itinerary Recommendation Diagram



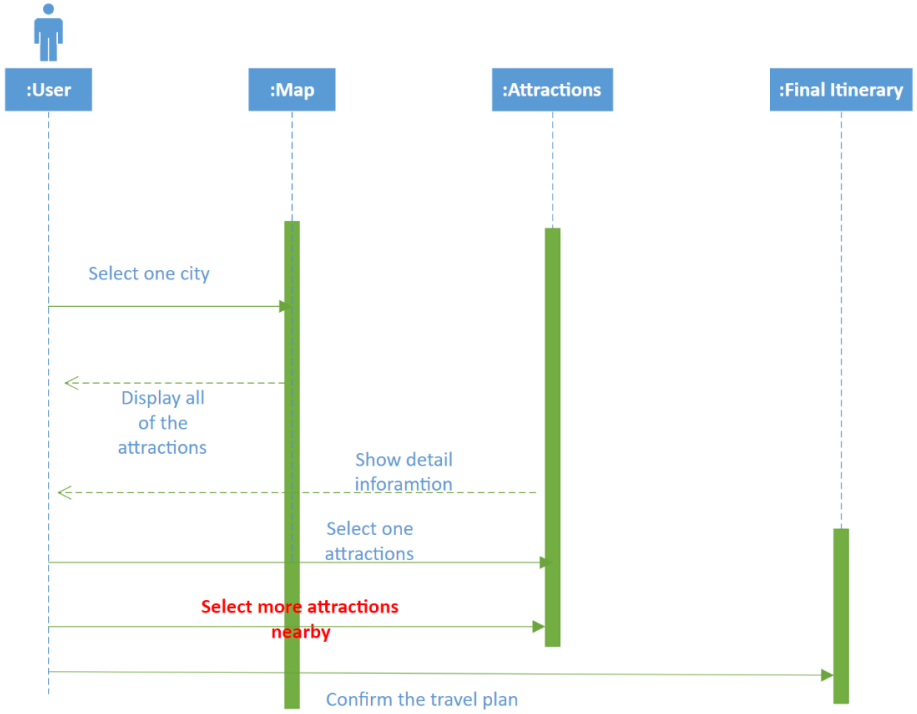
Generation Diagram



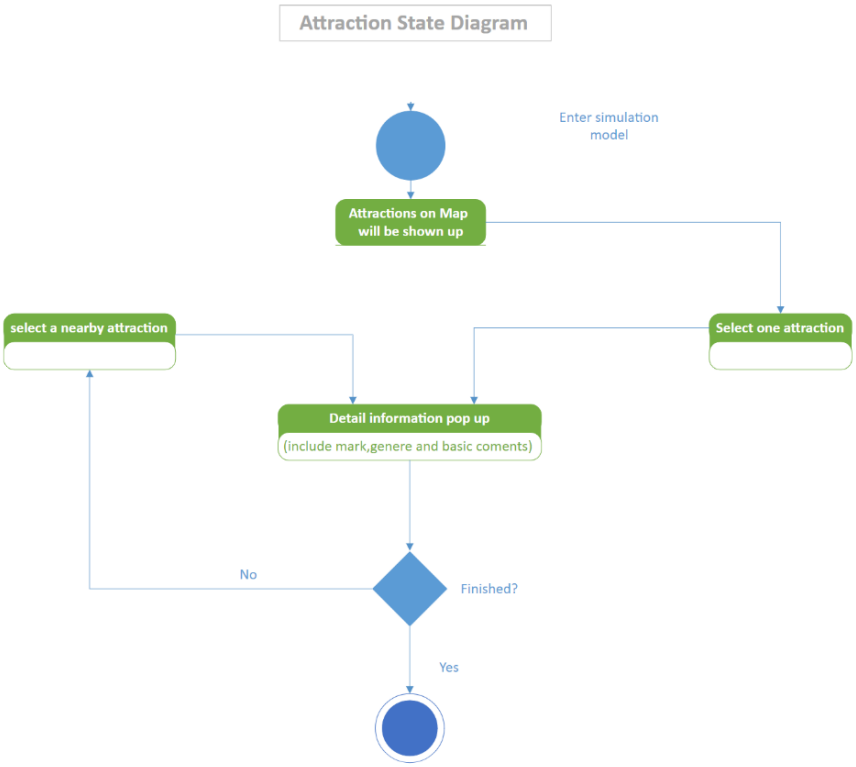
Feedback Diagram



Sequence Diagram



State Diagram



3.4.5 User Interface

None.