**Restaurant-Finder**

**User Requirements Document**

Jingwen Qiang & Ning Wang

**Table of Contents**

**INTRODUCTION** DOCUMENT SCOPE INTENDED AUDIENCE **SYSTEM OVERVIEW** PURPOSE

SCOPE

**GENERAL REQUIREMENTS**

SYSTEM IN CONTEXT USER CHARACTERISTICS CONSTRAINTS

SYSTEM-WIDE REQUIREMENTS (RECEIVED)

Actors

Events

SYSTEM-WIDE REQUIREMENTS (DERIVED)

Actors

**DETAILED REQUIREMENTS**

FUNCTIONAL REQUIREMENTS

User Requirements Model - User Case Overview

Detailed Use Case Specifications

TECHNICAL REQUIREMENTS Operational Environment Development Environment

**Introduction**

The objective of this User Requirement Document (URD) is to obtain agreement with Restaurant-Finder regarding the qualitative and quantitative characteristics of a proposed system. This URD is planning to initially write out our expectation and understanding of Restaurant-Finder System. Our team is still working on making development decisions so that the detailed requirements may be changed in latter development. At last, this URD avoids using technical language and describes functionality from general users’ viewpoints.

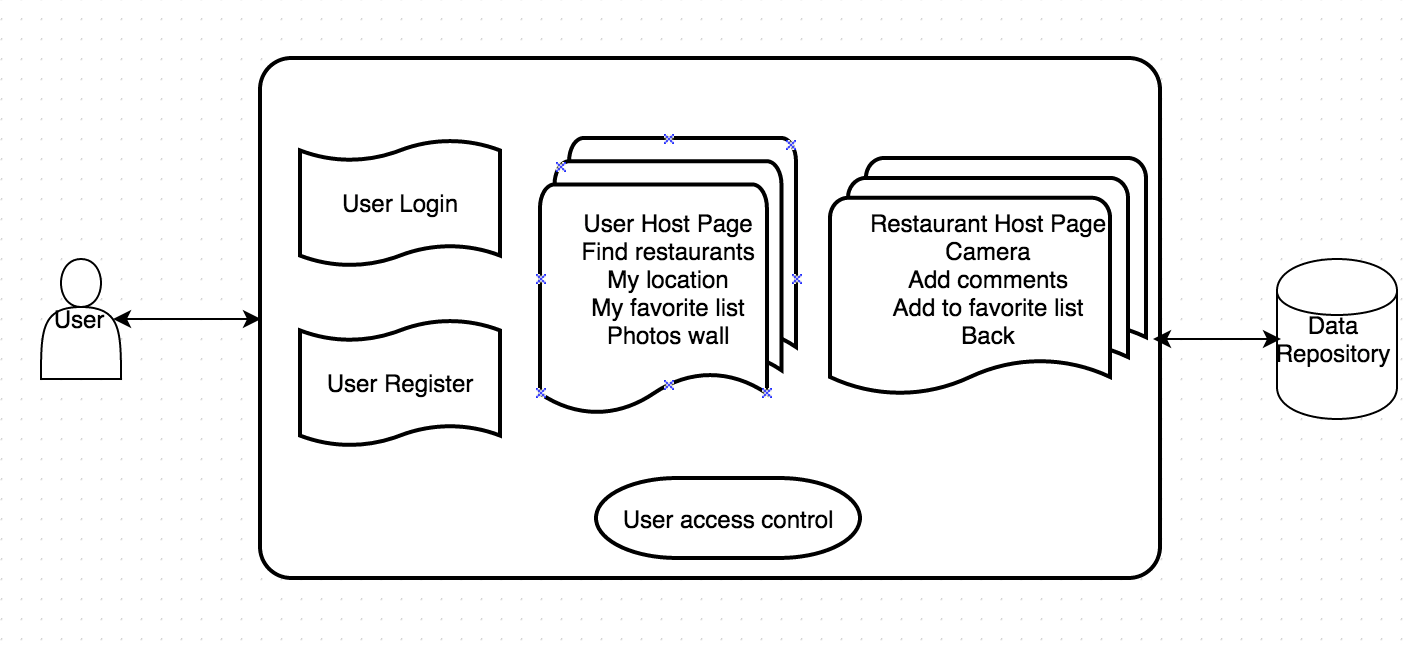
**Document Scope**

This URD provides an overall description of the Restaurant-Finder System. This URD provides detailed descriptions of all of the Restaurant-Finder System’s expected functionality. This URD outlines the sequence of events for each user cases, including detailed operation steps. Requirements for the operational and development environments also are described.

**Intended Audience**

There is only one type of users for our Restaurant-Finder System who is going to use the app to find restaurants.

**System Overview**



**Purpose**

The purpose of this Restaurant-Finder app is to help people who just come into a new place finding nearby restaurants. In addition, this app can also give people who cannot make a decision about selecting restaurants some useful advice. With the help of this Restaurant-Finder app, people can quickly search out their favorite restaurants, knows general information and customers comments about the restaurants.

**Scope**

The Restaurant-Finder app allows registered user to find restaurants based on restaurant names, addresses, and cuisine. It can help users position them and use their current location to find nearby restaurants. To help users record favorite meals, this app has a function to take a photo. In addition, this app supports users to call the desired restaurants for more information using the restaurants’ phone numbers. What’s more, this app has vibration sensor to help user make a selection.

The app also maintains lists of users’ favorite restaurants and restaurants information necessary for the users to reach and locate.

It is assumed that Restaurant-Finder System supports concurrent registered users access and requests from the users are synchronized.

**General Requirements**

In this project, Restaurant-Finder app requires a GPS to locate users’ position that can help them find nearby restaurants. In addition, this app requires a camera to help users take a photo of their favorite meals to record their experience. This app also requires a vibration sensor to capture the acceleration when user shakes a device. What’s more, this app also needs to get access users phone to make a call based on the phone number of the restaurants.

**System in Context**

The Restaurant-Finder System would realize the following functions necessary for helping users finding nearby restaurants, add comments and photos, and get contact with restaurants.

• Security function for authentication

• Location function for fixed position

• Getting access to camera for taking photo function

• Getting access to phone for call function

• Getting access to sensor for capture acceleration

**User Characteristics**

Registered users and unregistered users are non-technical individuals that are busy in their daily functions and have limited time for UI learning curves.

• Beginner to Intermediate system users

• Daily use of Restaurant-Finder app

**Constraints**

1. Secure login screen, saving username and password correctly.

2. Ease of use to system menus, pull down lists, selection tab etc.

3. Entry screen and section menus, pull-down lists etc. easily accessible, intuitive, along with consistency of UI design

4. Rely on GPS that has enough accuracy.

5. Rely on Google map working correctly.

6. Rely on device build-in camera working well.

7. Rely on device vibration sensor working well.

8. Users have to provide accurate information.

9. Rely on device connecting Internet.

10. Restaurants information is newest and correct.

**System-Wide Requirements (Received) Actors**

The Restaurant-Finder System recognizes two types of users. The first type of users is registered user. They can get access to all functions of this app. Second type of users is unregistered user. Since they do not have personal account, the system will require them to register an account before they login the app. The third type of user is app administrator, who just has one function of initializing restaurant information.

**Events**

Restaurant-Finder is a tool to assist the user to find restaurants. The most critical events are: 1) Find restaurants based on user’s input; 2) Find user’s location; 3) Find nearby restaurants based on user’s location; 3) Record user favorite restaurants; 4) Add comments; 5) Take a photo; 6) Get contact with restaurants; 7) Display all photos.

**Rules - TBD**

**System-Wide Requirements (Derived) Actors**

1. Unregistered Users

- Input their information to register.

2. Registered Users

- Find current location

The system allows user to get his current location in the map and in the text format.

- Find restaurants with current location/ restaurant name/ restaurant address/ restaurant cooking classification

The system allows user to find restaurants based on different methods.

- Go to official websites of restaurants

The system provides a link of restaurant’s official website and user can get access to it.

- Show comments

The system can provide current user with all comments by other users about a restaurant.

- Get contact with restaurants

User can call restaurants given by phone numbers of restaurants.

- Take a photo

User can use build-in camera to take a photo

- Maintain a list of user’s favorite restaurants

Registered users can add their favorite restaurants to my favorite restaurants list to record the restaurants they like, which is convenient for them to find the restaurants through the list. They can also delete restaurants from favorite restaurants list if they dislike some restaurants.

- Add comments

Registered users can add their comments for any restaurant. They can also add comments with taking a photo.

**Detailed Requirements**

**Functional Requirements**

The Restaurant-Finder System will allows users to find restaurants with name, address, cuisine and current location; to use map to see nearby restaurants; to add comments and photos and e.t.

For finding restaurants with name/ address/ cooking classification/ current location, the result of finding procedure is valid if 1) user provides a full or partial restaurant name, 2) user provides a full or partial address, 3) user provides a specific and normal cuisine style such as American, Chinese or Japanese.

For using map to see nearby restaurants, the system has to make sure that

Google map is working well and GPS has enough accuracy, which means that distance error should less than 1000 meters.

For adding comments and photo, users have to authorize the app to get access a camera. To upload the photo successfully, the size of photo should be smaller than 2M. To add comments successfully, the number of words should less than 100.

**User Requirements Model – Use Case Overview**

The table below offers a list of use cases. Detailed descriptions of the use cases are given in the Specifications section following this table.

|  |  |  |
| --- | --- | --- |
| Use Case ID | Use Case Name | Priority |
| AUTH-USR | Authenticate User | High |
| USR-REG | Register User | Low |
| FIND-RST | Find restaurants | High |
| FIND-LOC | Find location | High |
| REV-FAVR | Review my favorite restaurants | Medium |
| ADD-COMT | Add comment to a restaurant | High |
| ADD-FAVR | Add an restaurant to my favorite | Low |
| REV - PHO | Review photos | Low |
| TAK-PHO | Take a photo | Low |

Use Case Details – AUTH-USR: Authenticate User

|  |  |
| --- | --- |
| **Use Case ID:** AUTH-USR | **Use Case Name:** Authenticate User |
| **Primary Actor(s):** | All authorized users of Restaurant-Finder |
| **Description:** | Requests user name and password and authenticates user to login  Restaurant-Finder System. |
| **Preconditions:** | User login display dialog in home page is available and waiting for user input. |
| **Normal Flow of Events:** | 1. User enters user ID in ID field textbox and then tabs or clicks into password field and enters password 2. User clicks *login* |
| **Postconditions:** | After step 2, the User page appears. |
| **Frequency of Use:** | High |
| **Alternative Flows:** | User can click *New User* to choose to register a new account |
| **Exceptions:** | User does not enter data in the username and password dialog box and is given a message that all fields need to contain information or has entered an invalid username or password |
| **Assumptions:** | Restaurant-Finder System is up and running, user has obtained a valid username and password from the system administrator. |
| **Issues:** | User never has username or password. |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

|  |  |
| --- | --- |
| **Use Case ID:** USR-REG | **Use Case Name:** Register User |
| **Primary Actor(s):** | All new users of Restaurant-Finder without registration |
| **Description:** | Register a new user account for Restaurant Finder System |
| **Preconditions:** | User registration button in home page is available and waiting for user to choice. |
| **Normal Flow of Events:** | 1. User clicks *New User* button. 2. User enters username in username field textbox and then tabs or clicks into password field and enters password 3. User click *register* button |
| **Postconditions:** | After step 3, the User page appears. |
| **Frequency of Use:** | Low |
| **Alternative Flows:** | User can click physical return button back to login page or clear a misspelled username or password |
| **Exceptions:** | User enters data in the username field that is already used by other user |
| **Assumptions:** | Restaurant-Finder System is up and running. |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – FIND-RST: Find restaurants

|  |  |
| --- | --- |
| **Use Case ID:** FIND-RST | **Use Case Name:** Find restaurants |
| **Primary Actor(s):** | All users of Restaurant Finder System |
| **Description:** | User finds a restaurant using Restaurant Finder System. There are four types of finding restaurants methods: find nearby restaurants with my location, find restaurants with cuisine, find restaurants with address, and find restaurants with name. |
| **Preconditions:** | User is already logged into Restaurant Finder System and the  User page is displayed. |
| **Normal Flow of Events:** | 1. User selects *Find Restaurants*.  2. The User page displays three buttons with the following selections:  a. *Find restaurants by name*  b. *Find restaurants by address*  c. *Find restaurants by cuisine*  3. User inputs a text and clicks one of the three buttons.  4. The user page displays a list of restaurants.  5. If user shakes the device, go to step 6.  If user clicks a restaurant item, go to step 7.  6. The User page randomly selects a restaurant in the list.  Go to step 7.  7. Go to the Restaurant page. |

|  |  |
| --- | --- |
| **Postconditions:** | After step 7, the Restaurant page appears. |
| **Frequency of Use:** | High |
| **Alternative Flows:** | User can click other three tabs or clear a misspelled restaurant name, address, or cuisine. |
| **Exceptions:** | User clicks a button without entering any text. |
| **Assumptions:** | Restaurant-Finder System is up and running. User has logged into the User page. |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – FIND-LOC: find location

|  |  |
| --- | --- |
| **Use Case ID:** FIND-LOC | **Use Case Name:** find location |
| **Primary Actor(s):** | All users of Restaurant Finder System |
| **Description:** | User finds his location using Restaurant Finder System and then find nearby restaurants with his current location. |
| **Preconditions:** | User is already logged into Restaurant Finder System and the  User page is displayed. |
| **Normal Flow of Events:** | 1. User selects *My location*.  *2.* The User page displays a map and shows user’s current location on the map using a marker. |

|  |  |
| --- | --- |
|  | 1. The User page shows all nearby restaurants on the map if there are available restaurants nearby current location. 2. If user clicks his current location’s marker, a dialog will show text address.   5. If user clicks one of the restaurants on the map, a toast will show restaurant name, price, cuisine and address.  6. User clicks a toast of a restaurant and goes to the restaurant host page. |
| **Postconditions:** | After step 6, the Restaurant page appears. |
| **Frequency of Use:** | High |
| **Alternative Flows:** | After knowing his current location, user selects *Find Restaurant*  Tab to find a restaurant with his current location. |
| **Exceptions:** | There is no nearby restaurant based on user current location. |
| **Assumptions:** | Restaurant-Finder System is up and running. User has logged into the User page. |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – REV-FAVR: review my favorite restaurants

|  |  |
| --- | --- |
| **Use Case ID:** REV-FAVR | **Use Case Name:** review my favorite restaurants |
| **Primary Actor(s):** | All logged in user of Restaurant Finder System |

|  |  |
| --- | --- |
| **Description:** | User reviews a list of his favorite restaurants in My favorite restaurants page. User can delete and go to restaurant through this page. |
| **Preconditions:** | User is already logged into Restaurant Finder System and the  User page is displayed. |
| **Normal Flow of Events:** | 1. User selects *My favorite restaurant*.  *2.* The User page displays a list of restaurants.  3. If user clicks one of the restaurants, go to step 5.  4. If user slides a restaurant item in the list view, the restaurant will be deleted from favorite restaurant list.  5. Go to the restaurant page. |
| **Postconditions:** | After step 5, the Restaurant page appears. After step 6 and 8, keeps the User page with *My favorite restaurant* selection. |
| **Frequency of Use:** | Medium |
| **Alternative Flows:** | User can select *Find my location* and *Find restaurant* instead. User can also click *cancel* within the input textbox display to cancel or clear a misspelled restaurant name. |
| **Exceptions:** | When user clicks *Delete this restaurant* button, there is no |

|  |  |
| --- | --- |
|  | restaurant in my favorite restaurant list. User clicks *Add this restaurant* button but input an invalid restaurant name. |
| **Assumptions:** | Restaurant-Finder System is up and running. User has logged into the User page. |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – REV- PHO: Review photos

|  |  |
| --- | --- |
| **Use Case ID:** REV- PHO | **Use Case Name:** Review photos |
| **Primary Actor(s):** | All new users of Restaurant-Finder without registration |
| **Description:** | User review all photos taken by users |
| **Preconditions:** | User is already logged into Restaurant Finder System and the  User page is displayed. |
| **Normal Flow of Events:** | 1. User clicks *Photos Wall tab*.  2. The photo page displays all photos taken by users.  3. User clicks one of the photos and it will zoom in.  4. If user slides the screen leftward, he can see previous photo.  5. If user slides the screen rightward, he can see next photo.  6. If user clicks the photo again, it will reduce back to normal size and all photos can then show out. |
| **Postconditions:** | TBD |
| **Frequency of Use:** | Low |
| **Alternative Flows:** | User can click other three tabs in the user host page |
| **Exceptions:** | TBD |
| **Assumptions:** | Restaurant-Finder System is up and running. |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – ADD-COMT: Add comment to a restaurant

|  |  |
| --- | --- |
| **Use Case ID:** ADD-COMT | **Use Case Name:** Add comment to a restaurant |
| **Primary Actor(s):** | All users who have been chosen a specific restaurant and are authorized by Restaurant-Finder. |
| **Description:** | User chooses a restaurant by Restaurant Finder System and forward this restaurant page, then you can choose Add Comment in an action bar. User can write one to ten lines of comment. When you submit a comment, you can view the comments with your username right away in the restaurant host page . |
| **Preconditions:** | User is already chosen a specific Restaurant, and the Restaurant  Page is displayed. |
| **Normal Flow of Events:** | 1. User clicks action bar and selects *Add Comment*. |

|  |  |
| --- | --- |
|  | 2. The Restaurant page shows an input textbox. User inputs text comments and click *submit*  3. The Restaurant page shows a toast to notify that the comment has been added.  4. The Restaurant page shows new added comments and the username. |
| **Postconditions:** | The new comment displays in Comment list. |
| **Frequency of Use:** | High |
| **Alternative Flows:** | User can click other selection in action bar such as Camera, Add to favorite list and Back. |
| **Exceptions:** | User does not enter any text in textbox or has entered an invalid content or enter more than 10 lines of comments. |
| **Assumptions:** | Restaurant-Finder System is up and running, user has logged |

|  |  |
| --- | --- |
|  | into the User page successfully. |
| **Issues:** | User never has no id or password. |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – ADD-FAVR: Add a restaurant to my favorite

|  |  |
| --- | --- |
| **Use Case ID:** ADD-FAVR | **Use Case Name:** Add a restaurant to my favorite |
| **Primary Actor(s):** | All users who have been chosen a restaurant and are authorized by Restaurant-Finder. |
| **Description:** | User adds a restaurant into his favorite list. |
| **Preconditions:** | User is already logged into Restaurant Finder System and the  User page is displayed. |
| **Normal Flow of Events:** | 1. User clicks action bar and selects *Add to my favorite*.  2. The Restaurant page displays a toast showing the restaurant has been added to favorite list successfully. |
| **Postconditions:** | A restaurant is added in my favorite list. |
| **Frequency of Use:** | Low |

|  |  |
| --- | --- |
| **Alternative Flows:** | User can click other selection in action bar such as Camera, Add comments and Back. |
| **Exceptions:** | The restaurant already exists in the user’s favorite lsit. |
| **Assumptions:** | Restaurant-Finder System is up and running, user has logged into the User page successfully. |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

Use Case Details – TAK-PHO: Take photo

|  |  |
| --- | --- |
| **Use Case ID:** TAK-PHO | **Use Case Name:** Take photo |
| **Primary Actor(s):** | All users who have been chosen a restaurant |
| **Description:** | User chooses a restaurant by Restaurant Finder System and forward this restaurant page, then you can choose to take a photo. The photo will be automatically added to photos wall and can be seen by all other users. |
| **Preconditions:** | User is already chosen a specific Restaurant, and the  Restaurant Page is displayed. |
| **Normal Flow of Events:** | 1. User clicks action bar and selects Camera. 2. A build-in device camera page shows out. 3. User takes a photo. 4. If user select physical return or cancel button in the camera page, the photo will not be saved. 5. If user select physical accept button in the camera page, the photo will be automatically added to photos wall and can be seen by all other users. 6. After step 4 or step 5, user goes back to Restaurant host page from current camera page. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Postconditions:** | | User finds an appropriate routine to the selected restaurant. | |
| **Frequency of Use:** | | Low | |
| **Alternative Flows:** | | User can click other selection in action bar such as Add to favorite list, Add comments and Back. | |
| **Exceptions:** | | When user click camera, the device does not have a build-in camera. | |
| **Assumptions:** | | Restaurant-Finder System is up and running. | |

|  |  |
| --- | --- |
| **Issues:** | TBD |
| **Source:** |  |
| **Includes:** | TBD |
| **Associated Requirements:** | TBD |

**Technical Requirements**

The purpose of this section is to obtain agreement regarding the platforms to be used for deploying the working systems and for developing the working system. This is input to assist developmental and managerial decisions (i.e., finding restaurants with nearby location) as well as technical analysis (i.e., constraints imposed by the platform).

**Operational Environment**

The Restaurant-Finder app should be implemented on an Android Smart Phone with basic capabilities. To find users location and nearby restaurants, the app should get access to a valid map application (i.e., Google Map). To take a photo, the app should also be authorized to use the camera of the smart phone. To capture a shaking action, the app should be authorized to use sensor.

Before login the Restaurant-Finder app, the smart phone has to connect with

Internet.

Users can get contact with restaurants via phone call or goings to official websites.

**Development Environment**

Developers must have access to a modern object oriented development environment and support libraries with their corresponding APIs.

Developers should have access to GUI development tools for efficiency.

Developers may employ visual modeling tools. These tools, however, should not be employed if the developer is not familiar with them, as the time required to master the tools will detract from the experience. The modeling should still be done. It should be done using more basic drawing tools.