Intern Proktikum TeleKooperation

Sha Teng tstina2016@gmail.com
Zile Li littslerli@gmail.com
Jia Hu chyneya@gmail.com
Tianlei Zhu zhutianlei2b@gmail.com

TECHNISCHE UNIVERSITÄT DARMSTADT

FB20 Informatik

Page reserved for some purpose.

Contents

Abstract

Our project aims to develop a social application based on Foursquare. Foursquare is a location-aware mobile application and provides information about locations. We notice that Foursquare has some drawbacks and it is not user-friendly enough. With the requirement analysis, we keep some functions in Foursquare and make some changes to turn it to fit our requirements.

1 Introduction

Golf is an Android application for real time recommendation of nearby locations. Given different searching keywords from users and their real time positions, the App returns a list of locations in the nearby. Users can access the detailed information of the location, and also contribute comments and photos, which serve as suggestions for other users. Furthermore, users can check in and rate a location.

1.1 Motivation

All our group members are the students from Asia and we travel all over the European when no lectures and no exams. Usually we need to turn to tourism information for help when we travel in a new city. Actually it is inconvenient to only look at the city map because normally it provides only information about the attractions but not venues like restaurants, café, bars or shops. City map can not provide information about how good these viewpoints are, either. Sometime we follow the city map and come to a "unworthy" attraction. A new way is needed to get the information.

As students from computer science, we are motivated to develop a system to offer those information. As we use mobile phones everyday and everywhere today, the most convenient way to publish and receive those information is to design a smart phone application, which will be available if we have the phones in hand and can connect to Internet.

Such an application to solve those problems and it could also be interesting for other travelers. Considering about that Android has the biggest percentage of smart phone market, we decide to deliver the application on Android operating system.

With this application we should be able to get the basic information about the city's attractions and also comments from other travelers. During traveling we also want to share our experience about the tour, e.g., whether the food tastes good. Sharing with others, we can provide some advice to others and receive suggestion from others. This App should work as not only information resource but also social media. In conclusion, our goal is to develop the Android application to make the traveling more cheerful and more pleasant.

1.2 Project Scope

We are a four-member group and every member takes responsibility for different parts of the project and we also work together for some particular modules. Table 1 shows the concrete division of the project.

Module	Responsible
Requirement analysis	All team members
Server architecture	Zile Li
Server development	Zile Li & Tianlei Zhu
Android architecture	Sha Teng
Android development	Sha Teng & Jia Hu & Tianlei Zhu
Document	All team members
Testing	All team members

Table 1: Project scope.

2 Device and Environment Setup

2.1 Android

- Device platforms: Android 5.5 and later
- Development platform: Android studio
- Art design: Photoshop

2.2 Database

- DB: MongoDB

2.3 Server

- Node JS 7.10 (Express 4.15.2)

3 Dependencies

- Hardware: Android Smart Phone
- Map: Google Map
- Service/API documentation
- Profile/account/platform credentials
- Third-party software

4 System Architecture

4.1 System Overview

The overview of the Golf system is illustrated in Figure 1. The Golf system generally consists of 2 components, i.e. The Android Client end the Server end. The server end is deployed on public NodeJS cloud server. The Database is also deployed on-line using public MongoDB Cloud. Aside from database the server acquires venue data from Google Map.

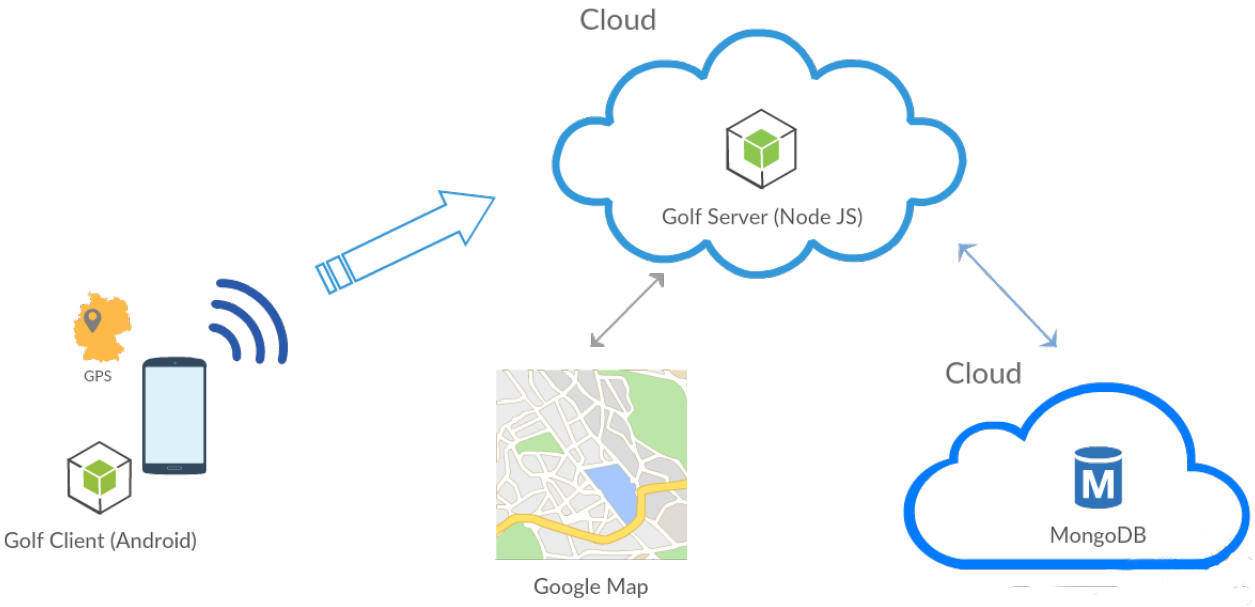

Figure 1: Architecture Overview

4.2 Client End

4.2.1 UI Module

Since all our group members come from China, we choose some Chinese elements for the App.

- Color: we apply red as the basic color for our App. In Chinese culture red represents positive and full of energy. We match it to the meaningfulness of traveling and sharing.
- Logo: in order to be consistent to the basic color, our project logo uses also red as background. Palette uses "#F40002" (R:244, G:0, B:2). We put a Chinese character " 众" in the center of the logo. It means many people together. We want to express the social function of our App. Figure ?? shows the logo.

Figure 2: Logo of our project.

- Background: we choose a photo of hotpot as background. We want to emphasize that our App can search for information about restaurants, café and bars. Hotpot is one kind of the most typical dishes in China and is an important Chinese element in Chinese culture. It also has the basic color-red, which fits the whole color style. Figure 4.2.1 shows final design version of the background.

Figure 3: Background of home page.

4.2.2 Google Map Module

We use API provided by Google. For getting the current location we use LocationRequest, and then the current location will be marked on the GoogleMap with a title "current location". We use onLocationChanged() function to listen the movement. If the location actually changes, the marker that indicates our current position will change accordingly at real time.

In order to avoid frequent and unnecessary data transfers of users android devices (and the usage amount of the Google API Keys, whose quota is limited), instead of automatically refreshing for the nearby venues, we update the markers only when user enters a search content and activates a query, and all relevant venues will be marked in the map view (also in the list view).

4.2.3 Asynchronous Image Download Module

According to the needs of the application, we use two kinds of strategies for the image asynchronous download. For downloading the avatar picture of current logined user, we use the AsyncTask to run download task and cache downloaded picture in local storage so that the application can directly display user avatar without requesting from server again. For downloading multiple pictures(e.g., when requesting comments list), we create a thread pool to run task and use LruCache to cache downloaded image. For both strategies we used OKHttp library to optimize the network request.

4.3 Server End

Server end is constructed by a NodeJS web server and a MongoDB database.

4.3.1 Components

- NodeJS Web Server. A web server implemented with NodeJS serves the request from Android endpoints. For requests of Venue searching the server queries Google Map service. The structure of the web server is demonstrated as Figure 4.
- MongoDB. A MongoDB cloud instance serves as the database of Golf.

Note that the web server and the MongoDB are not necessarily deployed on the same cloud service.

4.3.2 Implementation

The Web Server is implemented in Node JS. In order to develop the server logic efficiently, we used the Express framework to simplify the MVC works.

For the Modeling layer, we use Mongoose library to coordinate with the remote MongoDB.

Specially to avoid the code style of "Callback Hell" of NodeJS, we apply the coroutine package.

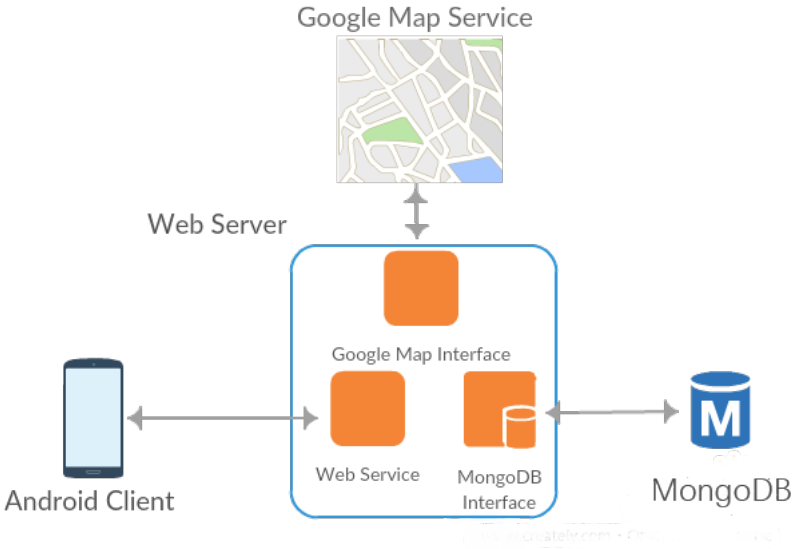

Figure 4: Web Server

4.3.3 Interaction and Adaption with Google Place API

The information of venues that Golf system retrieves are provided by Google through the public Google Place API. The server uses the Web Service version of Google Place API to communicate with Google Place, including searching for venues around a given position, retrieving detailed information about a venue and downloading photos of a venue. Golf server transforms the user request in the form of the internal RESTful APIs into a valid query for Google Place, and also converts the JSON result from Google Place into the internal format of RESTful APIs.

In the nearby searching, Google Place paginates the results, returning at most 20 venues each request. And at most Google place returns 3 pages for one query. The Golf server will concatenate all the results, and respond the client at one time.

One particular adaption from Google Place is the photo reference. Google Place provides photos of the venues in a form of a sequence of strings named photo reference. Server should append a photo reference to the API of photo downloading, in order to download the corresponding photo. On the other hand Golf allows users upload their own photos, which will be stored locally in Golf system. In order to distinguish the two sources of photos when user requests a photo, Golf server maintains an internal mechanism of photo referencing. The naming rules of the internal photo reference is listed in Table 2. The direct request for Google photos happens at the nearby searching, when Google returns the venues each associated with at most one title photo.

Source/Usage of the photo	Naming Rules
Local photos (photos in database)	Simply the ID of the photo document in database
indirect request of Google Photos	Google—-<location ID>—-<index in the photo array in Location document>
direct request of Google Photos	direct—-<Google photo reference>
users avatar	avatar—-<user ID>

Table 2: Naming Rules of Internal Photo Reference

4.3.4 Database Design

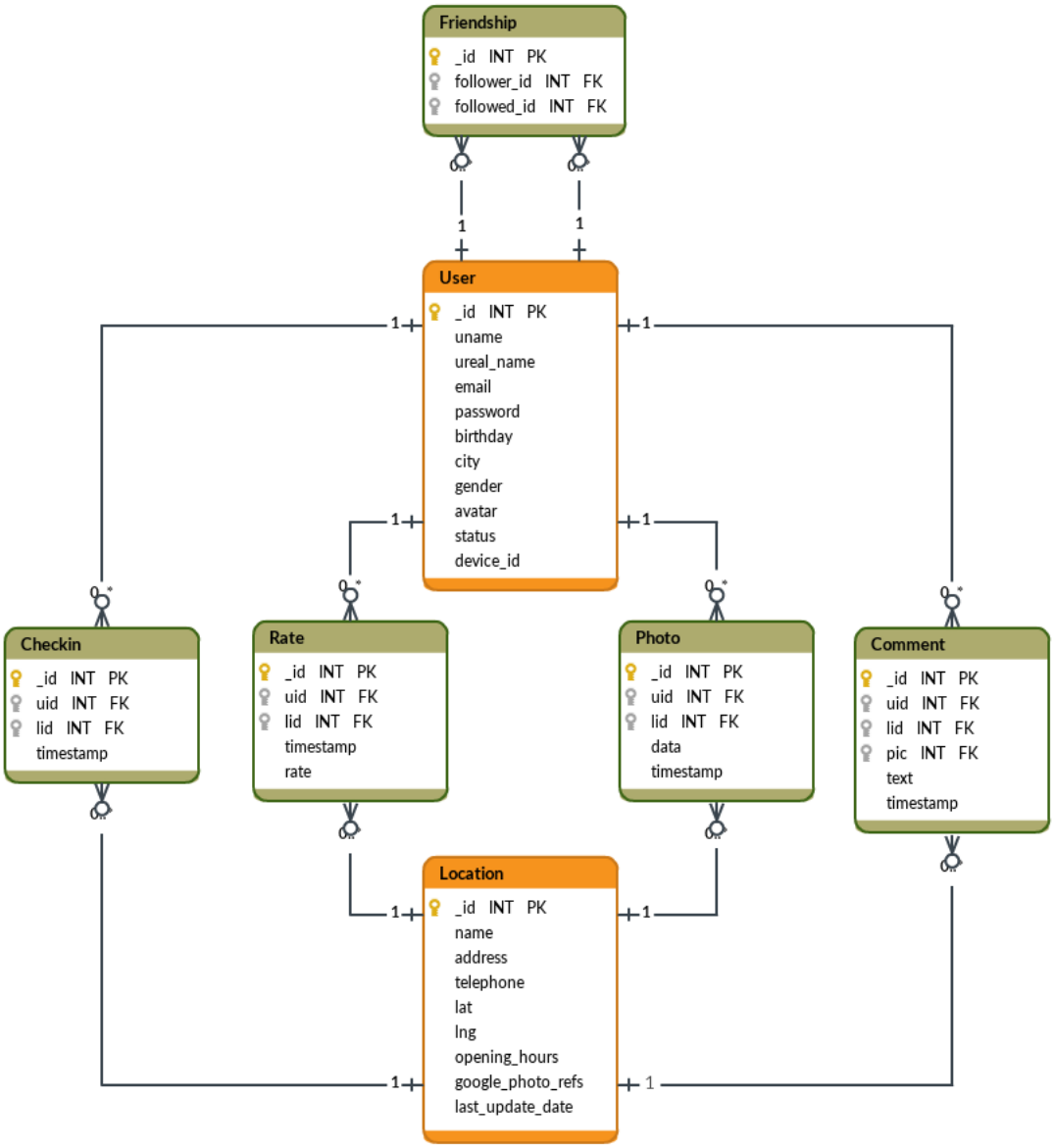

Figure 5: Database Design

The overview of the database design is demonstrated in Figure 5. The User and Location models are 2 central models, all the other models are relation models of these 2 models.

Location model is basically a cache of the venue detailed information from Google Place. When a user acquires detailed information of a venue, the detailed information will be added to the Location collection if this information is not yet saved before. This cache mechanism is for the consideration of saving repeated queries to Google Place, especially when user requests photos of a specific venue. Without the cache the server would have to request Google Place for the photo references of this venue, and then acquire the photos. With the cache of photo references the server saves the first request. As a trade-off of efficiency and up-to-date of information, a period of expiration of the cache information is set, which is 30 days as default. In case the information of one venue is expired, when its detailed information is visited at the next time, the information in the database will be refreshed.

Regarding to the deregistration of user account, all the corresponding historical records of the user will be deleted cascaded, except the comments and the photos, since the comments and photos are valuable informational assets for the product. In this case a special user name will be returned when the comments are queried.

4.4 RESTful API

The APIs are mainly divided into 2 groups, i.e. the user related and the venue related. Furthermore the incoming HTTP requests are routed into 2 corresponding controllers, which execute the actual logics.

A complete detailed list of all defined APIs is provided in the Appendix section 6.

4.4.1 User related APIs

User related APIs cover the following functionalities:

- User account related operations, including basic authenticating operations like user registration, user sign in and sign out, user deregistration. etc.
- Users interactions, including viewing other users profiles and contributions, following other users.

The requests are categorized mainly into 2 sets, i.e. the requests of HTTP "GET" method and the requests of HTTP "POST" method. The 2 categories are listed in Table 3 and Table 4.

Particularly, the request of user deregistration is submitted using "DELETE" method.

As strategy of authentication, we use the criteria of email plus password. In addition, a device ID is also used to identify a session.

API	Description
/user/usercomments/:uid	List Comments of Specific User
/user/:followed_id/followers	List Followers of Specific User
/user/:follower_id/followings	List Followings of Specific User
/user/profile/:uid	Viewing profile of specific User
/user/:uid/checkins	List Check In records of specific User
/user/search	Search for User given Keywords

Table 3: User Related APIs of "GET" Method

API	Description
/user/register	User Registration
/user/login	User Login
/user/logout/:uid	User Logout
/user/updateprofile/:uid/	Editing user profile
/user/password/:uid	Resetting user password
/user/deletecomment/:comment_id	Deleting user comment
/user/:follower_id/follow/:followed_id	user A (represented by the former parameter) Following user B (represented by the latter parameter)

Table 4: User Related APIs of "POST" Method

4.4.2 Venue related APIs

The venue related APIs cover the following functionalities:

- Venues searching, i.e. searching for venues nearby a specific position and customized keywords.
- Venue detail discovering, including querying venue detailed information, CRUD operations of comments, checking in and rating a venue.

Similarly, the requests are also categorized into 2 groups, i.e. the requests of HTTP "GET" method and the requests of HTTP "POST" method. A summary is listed in Table 5 and Table 6.

API	Description
/location/search	Search Locations by Position
/location/:lid/summary	Get Location Summary by Location ID
/location/:lid/details	Location detail
/location/:lid/photos	List Photo References
/location/photo	Download Single Photo
/location/:lid/comments	List Comments per Location

Table 5: Venue Related APIs of "GET" Method

API	Description
/location/:lid/user/:uid/addphotos	Add Photos
/location/:lid/user/:uid/addcomment	Add comment
/location/deletecomment/:comment_id	Delete comment
/location/:lid/user/:uid/checkin	Check in
/location/:lid/user/:uid/rate	Rate a location

Table 6: Venue Related APIs of "POST" Method

4.5 Android Opensource Libraries

The Android Client installs the following Opensource project from Github:

- EasyPermission: a wrapper library to simplify basic system permissions logic when targeting Android M or higher.
- ImagePicker: a custom album, which completely imitates WeChat UI, to achieve the camera, picture selection (radio / multi-election), cutting, rotation and other functions.
- ScrollParallaxImageView: extends ImageView, and provides parallax effects when it scrolls in the screen.It can be use in any view which can scroll its content, like ListView, RecyclerView, ScrollView, etc.
- Smiley Rating: a simple rating bar for android. It displays animated smileys as rating icon.
- LRecyclerView: a RecyclerView that supports addHeaderView, addFooterView, drop-down refresh, paged load data.
- OkHttp: an HTTP and HTTP/2 client for Android and Java applications.
- SwipeMenuListView: a swipe menu for ListView.
- AVLoadingIndicatorView: a collection of nice loading animations for Android.
- Transferee: just use GlideLoader of this library.

5 User Manual

5.1 System Requirements

Android 5.5 and later.

5.2 Getting Started

User installs the application from Google Play and opens it and user can see two main parts in the App, one is related to venue search, the other is about user profiles (The first time user opens the map, it may lack of user's current position and it may show a world map, user can go back and try again then everything works fine).

5.3 User Account

The recommended first step to use our application is to register. User should fill at least 3 text fields, that's email, user name, password, the others are optional and can be modified later in the profile page. A user can change his/her password, upload avatar or even delete the account.

5.3.1 Registration

User can register with Email address. User should provide some information.

- Necessary information: user name, Email address and password are necessary for registration.
- Unnecessary: real name, birthday, gender, city could be unavailable.

5.3.2 Sign In, Sign Out

With an account, user can sign in and sign out the App.

- Sign in: user can sign in with the registered account. Email address and password are necessary. After signing in, user can do some particular operations. See section 5.4.5.
- Sign out: user can sign out. After signing out, user can not edit profile and change settings.

5.3.3 User Profile

Two main kinds of information will be showed in "ME" page.

- Basic information: including user name, gender, birthday, city, avatar, real name.
- History: history of check-ins and comments.

5.3.4 Settings

5.3.5 Reset Password

User can reset password in "Settings". User should provide the old password and confirm the new password.

5.3.6 Delete Account

User can delete the account as he or she wants. After deleting, the account will not be available anymore.

5.4 Search for Venues

This application offers two possibilities to search a venue.

- The App provides 6 most common topics as default, namely breakfast, lunch, dinner, coffee and tea, night life, things to do. User can simply click on them and it will present the relevant results.
- User can enter some particular topic in the search box, either a real venue's name or a common concept like café. ATTENTION! The search radius is fixed so user can only search something that nearby him/her (Filter function will be released in the next version).

All relevant nearby venues will be presented in the map view. User can also change map view into list view to see the list of those retrieved venues.

5.4.1 Venue Detailed Information

From the map, the user can choose one location and will be located to the location detail page. The detail page will offer the following information.

- Photos: photos uploaded by the users who have been there before. In the page, these photos are showed in the form of rolling gallery. User can click on the gallery to see more photos (if more photos available) and the photos can be saved to local.
- Average score: the scores rated by users will be calculated averagely and showed in this page. The score will be maximal 5 and minimal 1.
- Venue basic information: information provided by the venues, including address, telephone number and whether it is opening or closed now.
- Top visitor: top visitor is the visitor who has the most check-in times. It will show the top visitor' avatar, name and check-in times. If there is nobody checked in before, it will give the information that nobody checks in here before.
- Comments: comments leaved by the users who were here will be showed and provide some tips for others. Comments can include text and pictures.

5.4.2 Check In, Rate, Contribute Tips and upload pictures

User can do some actions on the detail page and user should sign in before they do the following operations.

- Check in: user can click on the "CheckIn" button and the times of check-in will be added up. The user who has the most check-in times will be presented in the "Top Visitor".
- Rate: the user can rate the venues by choosing the rating stickers ("Great" for 5 points, "Good" for 4, "Okay" for 3, "Bad" for 2 and "Angry" for 1).
- Contribute tips and photos: user can leave some text and pictures as comments. The comments could be deleted by the submitter.
- Upload pictures: user can upload some local photos of the venues.
- Saving pictures: user can save the locations' pictures to local.

5.5 Friends System

In this App, all followings and followers are defined as friends. Followings are the people followed by user. Followers are the people who are following the user. User can follow others and can be followed by other. All followings and followers are listed in friends view.

- In personal profile page, click on "View Friends" button, all following friends and user's followers will be presented in form of list. Click on one friend item in the list, it will show the friend detail page.
- In friend detail page, user can click the follow button to follow this guy. After user has successfully followed someone, he/she can see the friend in friend list.

5.6 Summary Flow of Use Cases

This section we show the flow of use cases.

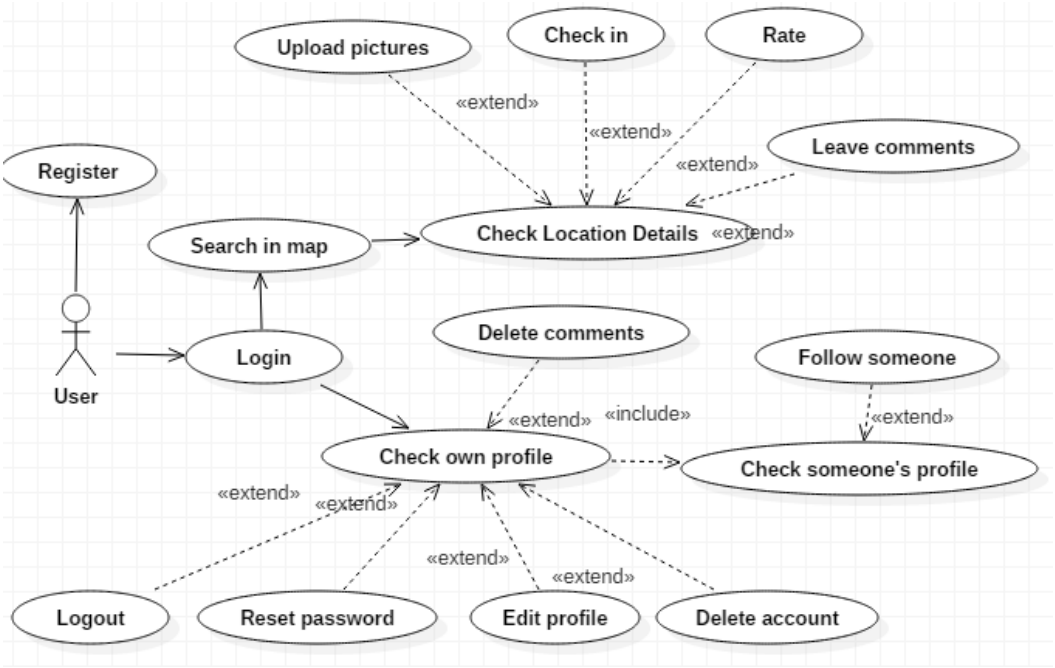

Figure 6: Use Case

6 Features List

Here we present a feature list.

Module	Functions	Descriptions
Account	Register	Register an account.
	Log in	Log in with the account.
	Delete	Delete account.
	Log out	Log out the account.
Profile	Edit Profile	Edit his profile after log in.
	History of operation	Show history of check-ins and comments.
Settings	Reset Password	Reset password with old password.
	Delete Account	Delete account.
Friends	Friend Search	search friends by entering others' user name.
	Friends detail	View friends' profile.
	Follow friends	Follow friends.
	View friends	View followings and followers.
Map	Map mode	Show venues in map mode.
	List mode	Show venues in list mode.
Venues	Search Venues	Search some particular topic.
	Venue details	Show location detail and pictures.
	Venues photos	Photos provided by Google and users.
	Top visitor	The user has the most checkins.
	Comments	Show comments leaved by users.
	Rate	User rates a venue.
	Check in	User checks in a location.
	Leave Comments	User leaves comments.

Table 7: Feature List

Appendices

A RESTful APIs

A.1 User

A.1.1 Register

URL	/user/register
Method	POST
Parameters	"uname" * "email" * "password" * "ureal_name" "gender" (with default "unknown") "birthday" "device_id" * "avatar" "city"
Responses	"ret_code": "0": success; "1": error "err_msg": "Error text" If success => "uid": the uid of this registered user account "uname": user name

Table 8: User register.

A.1.2 Login

URL	/user/login
Method	POST
Parameters	"uname" * "email" * "password" * "device_id" *
Responses	Success: "ret_code": "0" "uname" "uid" "avatar": photo ref of avatar, could be unavailable "gender" "city" Error: "ret_code": "1" "err_msg": "Error text"

Table 9: User login.

A.1.3 Logout

URL	/user/logout/:uid
Method	POST
Parameters	"device_id" *
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 10: User logout.

A.1.4 Update

URL	/user/updateprofile/:uid/
Method	POST
Parameters	"uname"* "password" "ureal_name" "gender" "birthday" "city" "avatar"
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 11: User update.

A.1.5 Reset Password

URL	/user/password/:uid
Method	POST
Parameters	"orig_password" * "new_password" *
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 12: User reset password.

A.1.6 Delete Account

URL	/user/:uid/delete
Method	DELETE
Parameters	"device_id" *
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 13: User delete account.

A.1.7 List user's comments

URL	/user/usercomments/:uid
Method	GET
Parameters	"count" * "page"

Responses	"ret_code": "0" for success, "1" for error
	"err_msg": "Error text"
	json:
	{
	"comments": {[
	{
	"comment_id"*: ID of comment
	"timestamp" *: "..."
	"text" *: "..."
	“lid”*: location ID
	"name" *: name of location
	"comment_photo": photo ref of comment, could be unavailable
	"location_photo": could be unavailable, photo ref of location
	}
	]}
	"total_pages":
	}

Table 14: List user's comments.

A.1.8 Delete comment

URL	/user/deletecomment/:comment_id
Method	POST
Parameters	"uid": "my" user ID *
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 15: User delete comment.

A.1.9 Get followers

URL	/user/:followed_id/followers
Method	GET
Parameters	None

Responses	"ret_code": "0" for success, "1" for error
	"err_msg": "Error text"
	{
	[{
	“friendship_id”:* ID of this friendship
	“uid”:* UID of follower
	“uname”:* username of follower
	“ureal_name”: could be unavailable, real name of follower
	}, {...},...]
	}

Table 16: Get followers.

A.1.10 Get followings

URL	/user/:follower_id/followings
Method	GET
Parameters	None

Responses	"ret_code": "0" for success, "1" for error
	"err_msg": "Error text"
	{
	[{
	“friendship_id”:* ID of this friendship
	“uid”:* UID of follower
	“uname”:* username of follower
	“ureal_name”: could be unavailable, real name of follower
	}, {...},...]
	}

Table 17: Get followings.

A.1.11 Follow Someone

URL	/user/:follower_id/follow/:followed_id
Method	POST
Parameters	
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 18: Follow someone.

A.1.12 Get User's Profile

URL	/user/profile/:uid
Method	GET
Parameters	"current_id" *: user id of "me"

Responses	{
	"ret_code": "0",
	"uid"*: "598970ab1e1e200213bb895f",
	"uname*": user name
	“avatar”: avatar of user, could be unavailable
	"ureal_name": real name, could be unavailable
	"birthday": "DD/MM/YYYY",
	"city": could be unavailable,
	"gender": could be unavailable,
	"can_follow"*: 1,yes; 0,no
	}

Table 19: Get user's profile.

A.1.13 List My Checkins

URL	/user/:uid/checkins
Method	GET
Parameters	"count" * "page"

Responses	{
	"ret_code": "0",
	[{
	“lid”*: location id
	“name”*: name of location
	“photo”: photo ref of a photo of this location, could be unavailable,
	}]
	"total_pages": # of total pages
	}

Table 20: List my Checkins.

A.1.14 List My Historys

URL	/user/:uid/history
Method	GET
Parameters	"count" * "page"

Responses	{
	"ret_code": "0",
	"histories": {[{
	"type": "checkin"
	"lid" *: location id
	"name" *: name of location
	"photo": photo ref of a photo of this location, could be unavailable
	"timestamp": DD/MM/YYYY
	},
	{
	"type": “rate”
	"lid" *: location id
	"name" *: name of location
	"rate" *: rate score
	"photo": photo ref of a photo of this location, could be unavailable
	"timestamp": DD/MM/YYYY
	},
	}]
	"total_pages": # of total pages
	}

Table 21: List my history.

A.1.15 Search for User

URL	/user/search
Method	GET
Parameters	"keyword" * "uid": user ID of "me"

Responses	{
	"ret_code": "0",
	[{
	"uid" *: user ID
	"uname" *: user name
	"ureal_name": user real name, could be unavailable
	"avatar": photo ref of avatar, could be unavailable
	"can_follow"*: "1" for yes; "0" for no
	}]
	}

Table 22: Search for user.

A.2 Location

A.2.1 Search Locations by Position

URL	/location/search
Method	GET
Parameters	"lat" * signed double "lng" * signed double "radius" "types" # [cafe, restaurant, bar, bakery, night_club] "keyword" # keywords concatenated with +, e.g. kebab + reis + pommes "name":# name of the location #: at least one of these fields should be defined

Responses	{
	"ret_code": "0" for success; "1" for error
	"err_msg": "Error text"
	
	json:
	{
	[{
	"name" *: "..."
	"lid" *: "..."
	"lat" *: double
	"lng" *: double
	"photo": photo ref of this location, could be unavailable
	}]
	}

Table 23: Search locations by position.

A.2.2 Get Location Summary by Location ID

URL	/location/:lid/summary
Method	GET
Parameters	"radius" "types" # [cafe, restaurant, bar, bakery, night_club] "keyword" # keywords concatenated with +, e.g. kebab + reis + pommes

Responses	{
	"ret_code": "0" for success; "1" for error
	"err_msg": "Error text"
	
	json:
	{
	"name" *: "..."
	"address" *: address of this location
	"lat" *: double
	"lng" *: double
	"photo_ref": photo ref of this location, could be unavailable
	}

Table 24: Get location summary by location ID.

A.2.3 Location Detail

URL	/location/:lid/details
Method	GET
Parameters	"uid": "my" user ID

Responses	{
	"ret_code": "0" for success; "1" for error
	"err_msg": "Error text"
	
	json:
	{
	"name" *: "..."
	"address" *: address of this location
	"open_now": 1:yes, 0:no, could be unavailable
	"telephone": "..." could be unavailable
	"avg_rate": average rate of location, could be unavailable
	"my_rate": my rate of location, could be unavailable
	"my_checkins": # my checkins of location, could be 0
	"top_checkins": [{
	"uid"*: id of this user
	"uname"*: user name of this user
	"avatar": avatar of this user, could be unavailable
	"ureal_name": real name of this user, could be unavailable
	"times" *: how many times the user checked in
	#top checkins: maximal 3
	}]
	}

Table 25: Location detail.

A.2.4 Download Photo References

URL	/location/:lid/photos
Method	GET
Parameters	"page" *: which page to retrieve, pagination "count": # photos per page, could be unavailable, default as 10
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text" "photos" *: [{a photo ref for further downloading}, {...} ,...] "uname" *: user name

Table 26: Download photo references.

A.2.5 Download Single Photo

URL	/location/photo
Method	GET
Parameters	"photo_ref": photo reference of the target photo
Responses	Failure: JSON: "ret_code": "1" for error "err_msg": "Error text" Success: An image file with the same file name of the photo_ref

Table 27: Download single photo.

A.2.6 Add Photos

URL	/location/:lid/user/:uid/addphotos
Method	POST
Parameters	Datas of multi files
Responses	"ret_code": "1" for error "err_msg": "Error text"

Table 28: Add photos.

A.2.7 List Comments per Location

URL	/location/:lid/comments
Method	GET
Parameters	"count" *: # of comments per page "page" *: # of page

Responses	"ret_code": "0" for success, "1" for error
	"err_msg": "Error text"
	
	json:
	{
	{
	"text" *: comment text
	"uid" *: user ID
	"uname" *: user name
	"timestamp" *: DD/MM/YYYY
	"ureal_name": user real name, could be unavailable
	"avatar": photo ref of user avatar, could be unavailable
	"comment_id" *: comment ID
	"comment_photo": photo ref of comment, could be unavailable
	},
	"total_pages" *: total pages of comments
	}

Table 29: List comments per location.

A.2.8 Add Comment

URL	/location/:lid/user/:uid/addcomment
Method	POST
Parameters	"timestamp" *: DD/MM/YYYY "text" * "photo": data of photo
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 30: Add comment.

A.2.9 Delete Comment

URL	/location/deletecomment/:comment_id
Method	POST
Parameters	"uid": "my" user ID
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 31: Delete comment.

A.2.10 Check-In

URL	/location/:lid/user/:uid/checkin
Method	POST
Parameters	"timestamp": "DD/MM/YYYY"
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 32: Check in.

A.2.11 Rate a Location

URL	/location/:lid/user/:uid/rate
Method	POST
Parameters	"timestamp": "DD/MM/YYYY" "rate": score of rating
Responses	"ret_code": "0" for success, "1" for error "err_msg": "Error text"

Table 33: Rate a location.

B Android Opensource Project

B.1 EasyPermission

https://github.com/googlesamples/easypermissions

B.2 ImagePicker

https://github.com/jeasonlzy/ImagePicker

B.3 ScrollParallaxImageView

https://github.com/gjiazhe/ScrollParallaxImageView

B.4 Smiley Rating

https://github.com/sujithkanna/SmileyRating

B.5 LRecyclerView

https://github.com/jdsjlzx/LRecyclerView

B.6 OkHttp

https://github.com/square/okhttp

B.7 SwipeMenuListView

https://github.com/baoyongzhang/SwipeMenuListView

B.8 AVLoadingIndicatorView

https://github.com/81813780/AVLoadingIndicatorView

B.9 Transferee

https://github.com/Hitomis/transferee