COMS 6998 Cloud Computing & Big Data

Final Project Report

Yilin Xiong (yx2274) Shengyi Lin (sl3759) Mengdi Zhang (mz2472) Di Li (dl2943)

1. Motivation
2. Functionality

This APP has the following functions:

* **User login/register:** This is a basic requirement. Every user of the APP should have an account to proceed with further operations in APP. Without an account, a user should not be allowed to start an event, nor should he/she be allowed to interact with other users in an event.
* **Update user profile:** A user should have profile information including some details to help others know him/her. Currently the profile includes nickname, gender and photo. If a nickname is set, it will replace the username for all the places where a user’s name should appear.
* **Start a new event:** An event is an activity organized by a user. Whenever a user want to find some partners together for travelling, sports, concerts, ect., he/she can start a new event with information including start time, place, description, etc.
* **Join/favorite an event:** A user can easily join an existing event. Or if a user likes an event but is not sure if he/she can join, he/she can just “favorite” the event to save it. Organizers and other users can view all the participants of an event easily. Thus they can know each other in advance.
* **Rate an organizer:** After an event, participants can rate the organizer in the score from 1 to 5. We try to build a credit score for the organizer through such process. Thus if an organizer has high score, he/she is less likely to be a fraud and more users may prefer to join his later event.
* **Follow users:** If a user meets someone he/she thinks interesting, he/she may want to keep in touch with that person. In such case the user can follow that person.
* **Post comments in an event:** To communicate with each other in an event, a user can post a comment. The comment can either be a review for the event, or a place to ask questions regarding the event and chat with other participants.
* **Post pictures in an event:** After an event, participants can share their pictures during the event. It is a good way to make friends and keep wonderful memories!
* **Search events/users:** Search events and users according to given keywords. A user may want to search for a specific user to follow. And he/she may also want to search for an event of specific topic or interest. By default, the APP shows the most popular events before any search.
* **Recommendation:** According to the user’s participations, the APP can recommend events that the user might be interested in. Our recommendation algorithm based on the idea that a user may be interested in those events joined by similar users. And we define a similar user as the one who has many common participations with the user.

1. Design
   1. Data Model

The ER diagram is shown below. There are 6 entities in this project including user, profile, event, category, comment and picture.



* 1. APP Design
  2. Server Design

We designed a collection of RESTful interfaces so that the APP can communicate with the server. These interfaces can be divided into two classes, user related interfaces and event related interfaces. The APP can communicate with the server through HTTP protocol. Data returned from server are all in JSON format. Our server is implemented using Django. The interfaces design are listed as below.

**[User-related Interfaces]**

**Name:** Login

**URL:** /user/login

**METHOD:** POST

**Parameters:**

username – username input by user

password – password input by user

**Return Example:**

Success – {‘success’: true, ‘id’: 1}

Fail – {‘success’: false, ‘message’: ‘Incorrect username or password’}

**Name:** Register

**URL:** /user/register

**METHOD:** POST

**Parameters:**

username – username input by user

password – password input by user

**Return Example:**

Success – {‘success’: true, ‘id’: 1, ‘username’: ‘user1’}

Fail – {‘success’: false, ‘message’: ‘Username Exists’}

**Name:** View profile

**URL:** /user/<user\_id>/profile

**METHOD:** GET

**Parameters:**

**Return Example:**

{"username": "di", "rating": 4, "gender": true, "followings": 3, "followers": 2, "photo": "/media/photos/21.jpg", "nickname": "doggie"}

**Name:** Update profile

**URL:** /user/<user\_id>/profile/update

**METHOD:** POST

**Parameters:**

nickname (optional) – user’s nickname

gender (optional) – user’s gender (male/female)

photo (optional) – user’s photo

**Return Example:**

{"success": true}

**Name:** View Followings

**URL:** /user/<user\_id>/followings

**METHOD:** GET

**Parameters:**

**Return Example:**

{"followings": [{"username": "yilin", "photo": "/media/photos/18.jpg", "nickname": "10", "id": 18}, {"username": "mengdi", "photo": null, "nickname": "leighanne", "id": 20}]}

**Name:** Follow user

**URL:** /user/<user\_id>/followings/add

**METHOD:** GET

**Parameters:**

following\_id – the user to be followed

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid following\_id’}

**Name:** Unfollow user

**URL:** /user/<user\_id>/followings/remove

**METHOD:** GET

**Parameters:**

following\_id – the user to be unfollowed

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid following\_id or following relation’}

**Name:** View followers

**URL:** /user/<user\_id>/followers

**METHOD:** GET

**Parameters:**

**Return Example:**

{"followers": [{"username": "yilin", "photo": "/media/photos/18.jpg", "nickname": "10", "id": 18}, {"username": "shengyi", "photo": null, "nickname": "sy", "id": 19}]}

**Name:** View events a user has joined

**URL:** /user/<user\_id>/participations

**METHOD:** GET

**Parameters:**

**Return Example:**

{"participations": [{"id": 4, "name": "Walking in the city"}, {"id": 5, "name": "Travelling to LA together"}]}

**Name:** View events a user has favorited

**URL:** /user/<user\_id>/favorites

**METHOD:** GET

**Parameters:**

**Return Example:**

{"favorites": [{"id": 4, "name": "Walking in the city"}, {"id": 5, "name": "Travelling to LA together"}]}

**Name:** Rate user

**URL:** /user/<user\_id>/rate

**METHOD:** POST

**Parameters:**

rate\_id – the user to be rated

score – rating score

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid ratee\_id or score’}

**Name:** Recommend events

**URL:** /user/<user\_id>/recommend

**METHOD:** POST

**Parameters:**

**Return Example:**

{"recommendations": [{"id": 4, "name": "Walking in the city"}, {"id": 5, "name": "Travelling to LA together"}]}

**[Event-related Interfaces]**

**Name:** View event

**URL:** /event/<event\_id>

**METHOD:** GET

**Parameters:**

user\_id – current user’s id

**Return Example:**

{"category": "Leisure", "name": "Walking in the city", "participants": [{"username": "shengyi", "id": 19}, {"username": "di", "id": 21}, {"username": "asdf", "id": 22}], "place": "New York", "is\_favorited": false, "time": "2015-05-20T22:18:00+00:00", "organizor": {"username": "yilin", "rating": 4.0, "nickname": "10", "id": 18}, "is\_joined": true, "description": "Walk in the city! Join us to see all the fantacies and magics in the city!"}

**Name:** Start event

**URL:** /event/add

**METHOD:** POST

**Parameters:**

name – event name

organizer\_id – organizer user’s id

place – the place of the event

description – the description of the event

time – the start time of the event

**Return Example:**

Success – {‘success’: true, ‘id’: 1}

Fail – {‘success’: false, ‘message’: ‘Invalid Request’}

**Name:** View categories

**URL:** /event/categories

**METHOD:** GET

**Parameters:**

**Return Example:**

{"categories": [{"id": 2, "name": "Leisure"}, {"id": 3, "name": "Travel"}, {"id": 4, "name": "Arts"}, {"id": 5, "name": "Sports"}, {"id": 6, "name": "Education"}]}

**Name:** Join event

**URL:** /event/<event\_id>/join

**METHOD:** GET

**Parameters:**

user\_id – the user who is joining the event

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid user\_id‘}

**Name:** Unjoin event

**URL:** /event/<event\_id>/unjoin

**METHOD:** GET

**Parameters:**

user\_id – the user who is unjoining the event

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid user\_id‘}

**Name:** Favorite event

**URL:** /event/<event\_id>/favorite

**METHOD:** GET

**Parameters:**

user\_id – the user who is favoriting the event

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid user\_id‘}

**Name:** Unfavorite event

**URL:** /event/<event\_id>/unfavorite

**METHOD:** GET

**Parameters:**

user\_id – the user who is unfavoriting the event

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid user\_id‘}

**Name:** View comments

**URL:** /event/<event\_id>/comments

**METHOD:** GET

**Parameters:**

**Return Example:**

{"comments": [{"content": "I like New York! The walking is wonderful!", "user": {"username": "di", "nickname": "troyal", "id": 21}, "id": 4, "time": "2015-05-20T18:36:00+00:00"}, {"content": "The event was very successful! Thanks everyone!", "user": {"username": "yilin", "nickname": "10", "id": 18}, "id": 5, "time": "2015-05-20T18:37:01+00:00"}]}

**Name:** Add comments

**URL:** /event/<event\_id>/comments/add

**METHOD:** POST

**Parameters:**

user\_id – the user who is posting the comment

content – comment content

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid user\_id‘}

**Name:** Remove comments

**URL:** /event/<event\_id>/comments/remove

**METHOD:** POST

**Parameters:**

comment\_id – the comment to be removed

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid event\_id‘}

**Name:** View pictures in event

**URL:** /event/<event\_id>/pictures

**METHOD:** GET

**Parameters:**

**Return Example:**

{"pictures": [{"image": "/media/album/1.jpg", "time": "2015-05-20T23:51:15+00:00", "id": 2, "user": {"username": "shengyi", "nickname": "sy", "id": 19}}]}

**Name:** Add pictures in event

**URL:** /event/<event\_id>/pictures/add

**METHOD:** POST

**Parameters:**

user\_id – user’s id

image – the image to be uploaded

**Return Example:**

Success – {‘success’: true}

Fail – {‘success’: false, ‘message’: ‘Invalid user\_id‘}

**Name:** Search events and users

**URL:** /search

**METHOD:** POST

**Parameters:**

searchString – the query key word(s)

**Return Example:**

\*TO BE ADDED\*

**Name:** View events by category

**URL:** /category/<category\_id>

**METHOD:** POST

**Parameters:**

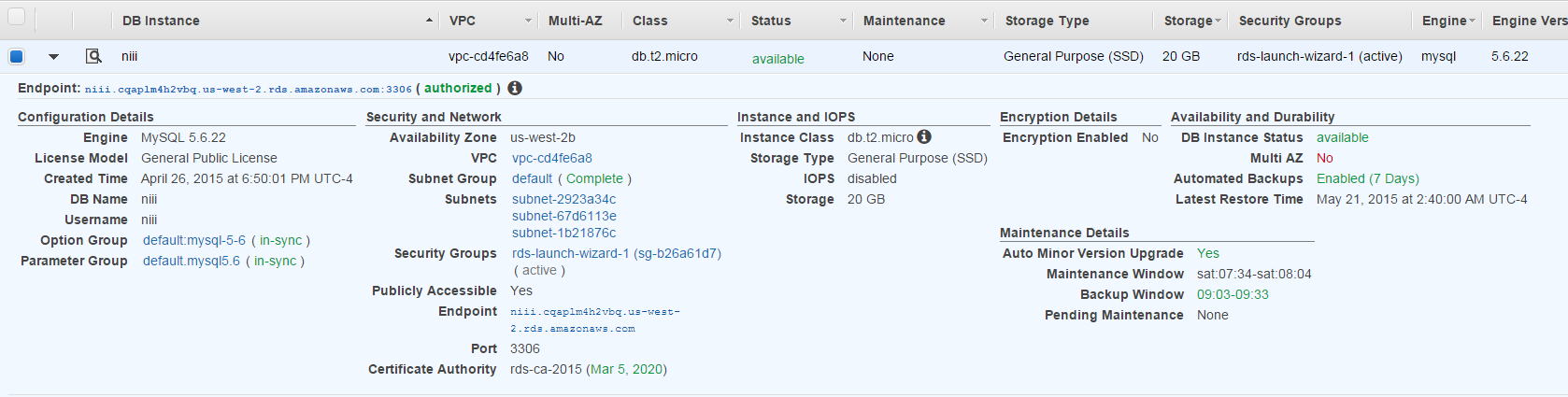
**Return Example:**

\*TO BE ADDED\*

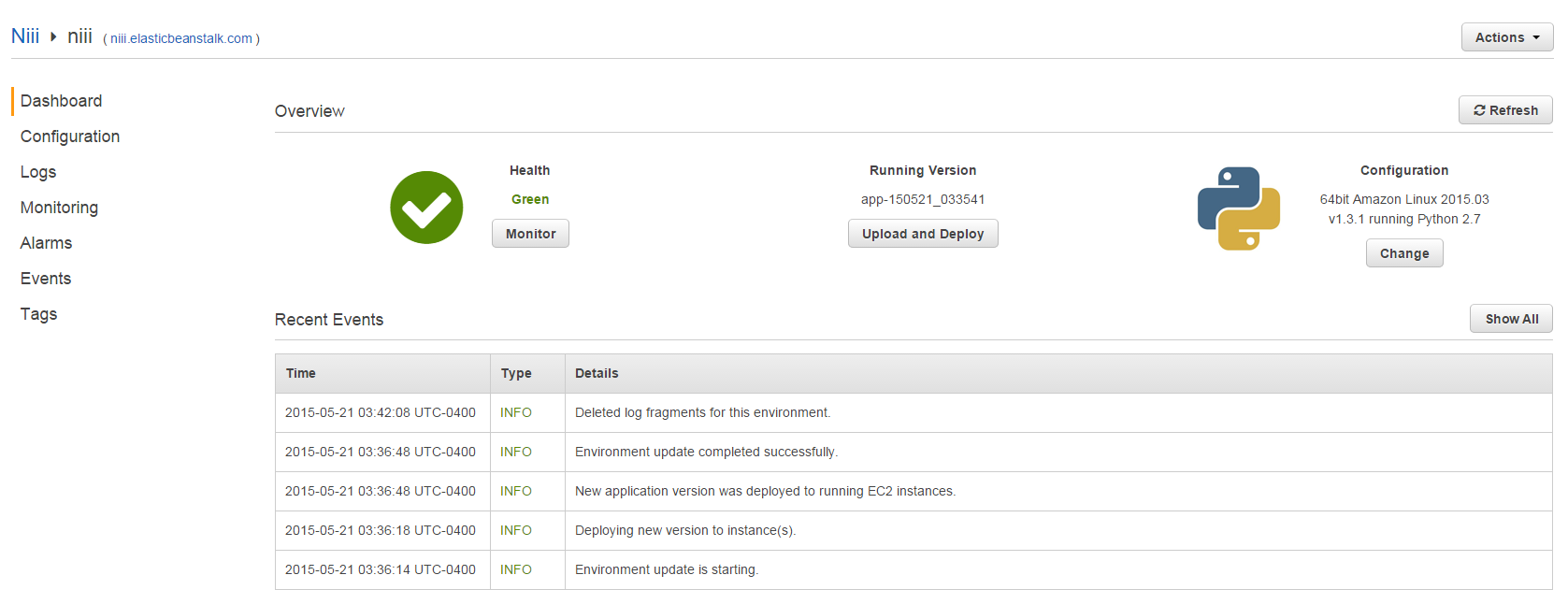
1. Deployment

We deployed our server on Amazon AWS.

We put our MySQL database on RDS. The settings are shown below.



We deployed our Web server using AWS ElasticBeanStalk. The configurations are shown below.



Our EBS extension configuration file includes the following settings:

option\_settings:

"aws:elasticbeanstalk:application:environment":

DJANGO\_SETTINGS\_MODULE: "NiiiServer.settings"

PYTHONPATH: "/opt/python/current/app/NiiiServer:$PYTHONPATH"

"aws:elasticbeanstalk:container:python":

WSGIPath: "NiiiServer/wsgi.py"

1. Achievement
2. Acknowledgement