Join Me: An Event Management Platform

Advanced Software Engineering Project Proposal

Team: MissingOne

Jiayi Li(jl4924), Chi Zhang(cz2465), Chengyun Yu(cy2468)

Abstract

Join Me is an event management platform where event organizers can post some group events, such as hiking, cycling or even study activities, search for the target attendees for their events and gather the event attendees' personal information. From the aspect of attendees, they can search and register the events they are interested in.

Part I

Background

With the rapid living pace of the modern society, people are always struggling with their works, lives and many other issues, which make them have less time to broaden their social circle. Sometimes when you want to have some group activities with your friends but they happen to have some personal issues and not able to attend, it's really hard for you to find someone else who shares the same interests with you and is willing to join in. Under such situations, you have to abandon the idea.

Goals

Our goal is to build an event management platform, *Join Me*, helping to reduce the practical problems shown above. No matter whether users plan to attend an event or organize one, users can use our platform to post or register the events they are interested in. From the perspective of organizers, we hope *Join Me* will make them able to collect or arrange attendee's personal information and organize the events in a much easier way. From the perspective of attendees, given so many different kinds of events in different fields on the *Join Me*, we hope attendees can quickly find the events they are interested.

Language, Platform and Tools

Language:

Python (or Java if needed).

Platform:

The platform will be run in Windows. (Other OS will be used for development if needed.)

External API usage:

1. MailGun: “A powerful Transactional Email APIs that can send, receive, and track emails.” [1] We will use this API to realize the group email function of our platform.
2. Mysqlclient: “Mysqlclient is an thread-compatible interface to the popular MySQL database server that provides the Python database API.”[2] We will use this API to do the data transmission.
3. Google Sign in API: The Google Sign in API allows users to sign in our platform with their Google Account.
4. Google Maps API: The Google Maps API can make the event organizers able to share the location of their activities and help attendees to find the activity location easily.
5. Google Calendar API: The Google Calendar API allows users to integrate the data from their applications into and use data from the Google Calendar. A

Data Store:
For data storage, we will hold a relational database on Amazon Web Service’s Relational Database Service. This means that all our data will be storage in a server from AWS. The server has a storage space of 5GB. We can connect to the database using the “mysqlclient” python package.

Product Logic
Organizers:

- ➢ Register an account
- ➢ Sign in
- ➢ Create a new event
- ➢ Post the event
- ➢ Get the list of registered users and kick out attendees when needed
- ➢ Edit the event and notify all attendees.
- ➢ Contact the attendees if needed.

Attendees:

- ➢ Register an account
- ➢ Sign in
- ➢ Type keywords in the search box
- ➢ Browse all events and register the one you like
- ➢ Unregister the event if needed
- ➢ Contact the organizer

Innovative Function Point
Organizers:

- ➢ Group Email:

Update the events information or send emergency notices by sending the group emails to all the attendees.

➢ One-click Download:

Download the personal information of the attendees in form of the csv file with just a single click.

➢ Online Check-in:

Help organizers to do the check-in work in a more convenient way.

➢ Others:

To be decided.

Attendees:

➢ Easy Register:

Register the event with just a single click. The register information will be automatically gathered from attendee's profile.

➢ Others:

To be decided.

Part II

User Stories

1. **As an** outdoor enthusiast, **I want** to invite my friends to go cycling in the Central Park on this weekend. However, all of my friends can't come because of some personal issues. In this case, I want to find someone else to cycle with me, **so that** I post my cycling route and schedule on *Join Me*.

   My conditions of satisfaction of this case are that:

- I want my events to be attractive, so that I should be able to post not only plain words but also pictures or even videos on the billboard.**(a)**
  (Test case: 2.1)

- I want to inform all attendees in time rather than post public messages on the platform, the platform should allow me to send emails to all attendees when I want to send some notices or updated information.**(b)**
  (Test case: 2.3.1; 2.3.2; 2.3.5)

- I hope I can connect with each attendee personally. In *Join Me*, I can download all attendees' personal information in form of Excel file.**(c)**
  (Test case 2.3.4)

2. **As a** normal user, **I want** to find some interesting activities on weekends and make some new friends. **So that**, I open *Join Me* and try to find some events by typing some keywords.

   My conditions of satisfaction of this case are that:

- I want to find the most related and popular events that match with my searching keywords.**(d)**
  (Test case: 3.1)

- I want to check the profile of the organizer to have an overview of the host and connect with the host to get more detail information about the involving activity.**(e)**
  (Test case: 3.3)

- I want to register in the event when I'm interested in it and unregister it when I am not interested in it anymore.**(f)**
  (Test case: 3.2; 3.4)

3. **As a** graduate student in Columbia, **I want** to find some other students who have registered in the same class as me to discuss some questions after class or the teammates for the group projects. **So that** I post my requirements on *Join Me*.

   My conditions of satisfaction of this case are that:

- I want to find the students who have registered in the same class as me. As a result, I need to communicate with each attendee after I get their personal information. And I hope I can kick off the attendees who are not matched with my demand. **(g)**
  (Test case: 2.3.2)

In each user story, each user action is linked with an index. Each user-level function is associated with certain test plan in Part III.

Part III

Acceptance testing Case
1. Personal account
1.1 Sign up
Input:

In the sign-up section, users are required to enter Google account and personal information. Information includes:

- Name, Gender, Age, Location, Google account,
- Personal Tags: Tags describe your favorite kinds of events
- Self-description: Words that can best describe yourself and your favorite activities

Output:
The common output should be a new item that inserted into the database, which we can test it by searching in the server-side database. If a user is registered, his or her information could be searched by using the Google account registered.

Special case:
The special output happens and alerts when users want to enter an improper value, like:

- Enter illegal email address in the Personal email
- Enter letters rather than numbers in Age

1.2 Log in
Input:
In the log in section, users are required to log in our platform with personal Google account which they used in the sign-up section. The required input is personal Google account and password, which will be identified by Google sign-in API.

Output:
The common output should be the access to our platform, then the user could post and search events;

Special case:
The special output happens and alerts when the input are:

- Unregistered user account
- Incorrect Google account or password

1.3 Profile edit
Input:
In the profile edition section, users could edit their personal profiles which are created in the sign-up section. The input should be the information that users want to change.

Output:

The common output should be the successful submission of the updated version.

Special case:
The special output happens and alerts when users want to enter an improper value, like:

- Enter illegal email address in the Personal email
- Enter letters rather than numbers in Age

2. Organizer
2.1 Create Event
Input:
After a user sign in his account, if he wants to create an event, he should input:

- Title: The title of the event.
- Description: A brief but attractive description of the event.
- Images: Some related images that related to the event or previous similar events.
- Location: The place that the event will be held. It can be the name of the place or even the coordinate of that place.
- Date: The date that the event will be held. And the time window for registration.

Tag: The type of event. (e.g. Sports, Academy, etc.)

Output:
The ideal output should be a hashmap whose key should the id of the event which will be assigned by the system automatically, while the value should be another hashmap to store the 6 attributes of the event as mentioned above. The hashmap should look like:

{Event ID: {Title: content, Description: content, Image: content, …..}}

Then the hashmap will be stored in our database.
From the aspect of the organizer, he will get a notification saying that the event is successfully created.

Special case:
The invalid inputs may commonly be some missing in important information (such as the title, the description, etc.), the system will not store the event into the database and remind the user of the error.
From the aspect of the organizer, he will get a notification saying that the event is failed to create.

2.2 Friend Invitation
Input:

After creating the event, the organizer can invite his friend to attend the event. He should input the email address of those he wants to invite. The common input should be a valid email address and the email address should have been used to sign up our system.

Output:
The ideal output should be an invitation email with the event details sent to the target users. From the aspect of the organizer, he will receive a notification that the invitation has been sent. From the aspect of the target user, he will receive the invitation email.

Special case:
For the special input (such as an invalid or unregistered email),
the output should be a notification saying that the email is not valid.

2.3 Attendee Management

2.3.1 Edit the event

If users want to edit the original content of the event.
Input:
The organizer clicks the button “edit” and inputs the detailed content.

Output:
A group email with the editing details will be sent to all attendee. And the organizer will receive a notification saying that changes saved.

Special cases:
If the organizer’s modification misses some important information (such as the title, the description, etc.), the organizer will get a notification saying that the changes are failed to save.

2.3.2 Delete attendee

If the organizer wants to kick an unsatisfied attendee:
Input:
The organizer clicks the button “remove” after the id of each attendee.

Output:
The organizer will receive a notification saying attendee successful removed. And the removed attendee will receive an email to notify that he is kicked by the organizer.

Special cases:

There should be no special case.

2.3.3 View all attendees
If the organizer wants to view all the attendees:
Input:
The organizer clicks the button “Attendees”.

Output:
The organizer will able to see a list of attendee on the page.

Special cases:
If no one attends the event, the organizer gets an empty list.

2.3.4 Download attendees list
Input:
The organizer clicks the button “download attendees list” on the event viewing page.

Output:
The organizer gets a CSV file containing the user IDs, email addresses and all the personal information of all attendees of the event.

Special Cases:
If no one attends the event, the organizer gets an empty file.

2.3.5 Group Email
Input:
The organizer enters a message to all attendees and clicks the “send group email” button.

Output:
Each attendee receives an email from the organizer’s registering email address containing the message and the event ID.

Special Cases:
The organizer inputs no message, an alert message appears on the app.

3. Attendee
3.1 Search events
Input:

When an attendee searches for an event, he or she may include the following input:

- Event ID: The ID of a certain event.
- Tag: The type of event that the user seeks to search.
- Date: The time that the event is held.
- Location: The place the event is to occur.
- Keyword: Word contained in the title or description of the event.

Output:
The successful output should be a list of event tuples containing the ID and the basic information of the events that suit the search criteria. The attendee will be able to view all the details of the events.

Special Cases:
A special case is that the search is submitted with none of the input above specified, the output should be a list of event tuples of the default order. If no result matches the criteria, the app will display a message indicated no matches were found.

3.2 Join events
Input:
The attendee can attend an event by clicking the join button.

Output:
The attendee shall then receive a confirmation message in the app that specifies the event ID of the event joined. The attendee can also see her name on the list of attendees of the event.

Special Cases:
In special cases where the event does not exist or have already been joined by the user previously, the attendee will receive an error message in the app.

3.3 Contact organizer
Input:
To contact the organizer, the attendee can enter a message and click on the “contact organizer” button.

Output:
The organizer shall receive an email from the attendee’s registering email containing the exact message.

3.4 Quit event

Input:

The attendee inputs an optional message for the reason of quitting and clicks the “quit” event.

Output:

The organizer receives an email containing information about the attendee and the event, as well as the message given by the quitting attendee. The attendee is removed from the attendees' list of the event.

Special Case:

If the event does not exist or is not joined by the user previously, the attendee will receive an error message in the app.

Reference

[1] MailGun Official Website. Retrieved from: https://www.mailgun.com/

[2] Mysqlclient Official Website.
Retrieved from: https://github.com/PyMySQL/mysqlclient-python