MeFree

app for event organization and management

Abstract

MeFree is a mobile app that facilitates the organization and management of events and activities. MeFree allows users to keep track of the details of any ongoing events, and helps users to host events more easily by maintaining an in-app friend-list with which users can make use and invite friends when initiating a new event.

MeFree makes use of the Google Cloud Messaging(GCM) service to handle notification and event invites which travel between devices, together with a web server set up with XAMPP storing all user and event information.

Keywords: MeFree, event organization, event management, friendlist, invite, notification, GCM, XAMPP

Introduction

Everyone loves to hang out with friends, and some are even eager to meet new ones on joining activities and events. However, with the majority of existing social networking apps, the management and organization of events could be a dreadful task, for example on Whatsapp, organizing a group activity would easily mean scanning through pages of messages.

With MeFree comes a solution. MeFree is the app designed for organizing and managing events. With the in-app event dashboard and friendlist, users can easily keep track of any ongoing events in terms of the events details and knowledge of participants as well as inviting friends to events without the trouble of sending and checking tens or hundreds of messages.

With the help of GCM service and a XAMPP server, MeFree is here to provide an easy and effective solution for event organization and management.

Usage

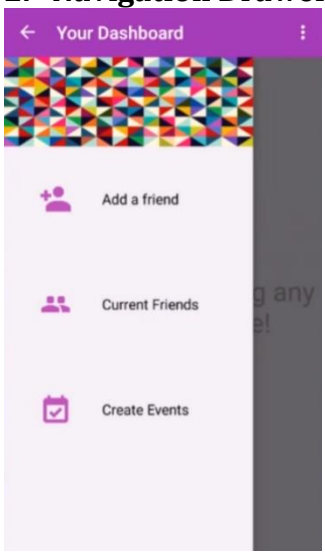
1. Login

The user enters an existing username and the corresponding password to login. A toast for incorrect password will prompt if the password is invalid.

The user enters a new username that does not exist in the database and a corresponding password to register a new user account. A toast will prompt and user will be logged in automatically on successful registration.

Fig (1)

2. Navigation Drawer



A navigation drawer which is usually hidden on the left of the screen will prompt on successful login. The drawer can be hidden again by swiping to the left or called out by pressing the menu button on the upper left.

There are 3 buttons on the navigation drawer, directing to functionalities of adding a new friend, viewing current friends and creating an event.

Upon redirection to the different layouts of activities, user can call out the navigation drawer to redirect to another layout or pressing the back button for returning to the previous screen.

Fig (2)

3. Dashboard

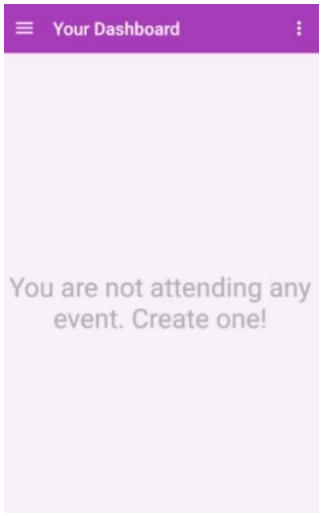


Fig (3)

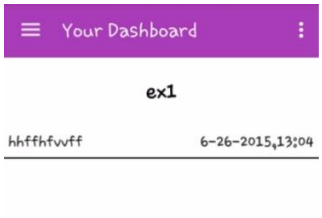


Fig (4)

A dashboard showing the events the user is attending serves as the home screen of the app.

Upon the first login of a new user, the dashboard shows no event (Fig(3)) and will be updated accordingly after the user has created or accepted invitation to any event(Fig(4)).

4. View Current Friend

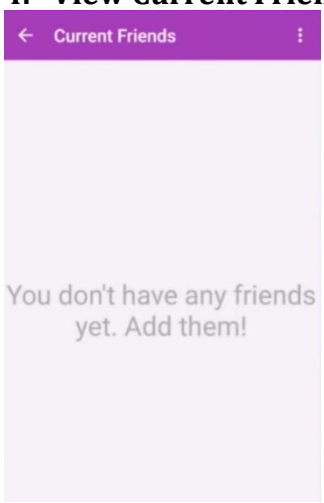


Fig (5)

Fig (6)

By clicking on the "Current Friend" button on the Navigation Drawer, the user will be redirected to the screen of a list of current friends of the user.

Upon the first login of a new user, the friendlist shows no username(Fig(5)) and will be updated accordingly after the user has made friend with any other user(Fig(6)).

Any other user who has accepted a friend request from the user or whom the user has accepted the friend request of, will have the username appeared in the current friend list of the user.

5. Add New Friend

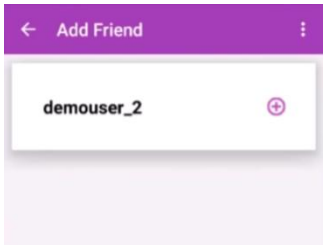


Fig (7)

By clicking on the "Add New Friend" button on the Navigation Drawer, the user will be redirected to the screen of a list of all users registered and existing in the database. (Fig (7))

Upon clicking the add button besides the username of a user, a friend request will be sent immediately to that user and the icon of the add button will change to an icon with a tick signaling a sent request. (Fig (8))

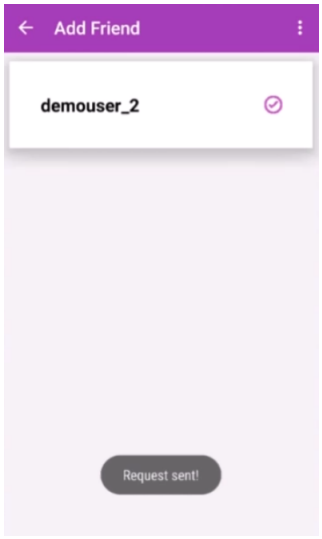


Fig (8)

6. Friend Request

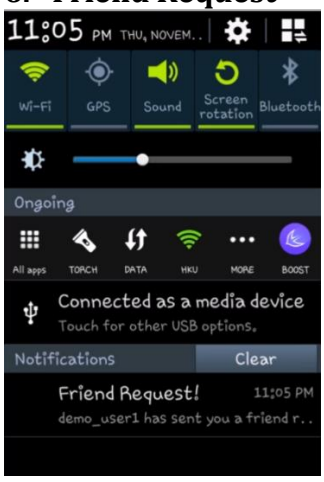


Fig (9)

When another user has sent the user a friend request via "Add New Friend", a notification will prompt on the user's device.

The notification specifies the name of the requesting user and the message of "has sent you a friend request" to distinguish itself from any other friend request or event invitation. (Fig (9))

Upon clicking on the notification, the user will be redirected to the screen of viewing "Friend Requests", where the user can choose to click on accept(tick) or decline(cross) of the specific friend request. If the user chooses to decline, the friend request will disappear from the list. (Fig (10))

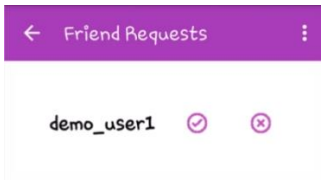


Fig (10)

If the user accepts the friend request. A toast will prompt specifying

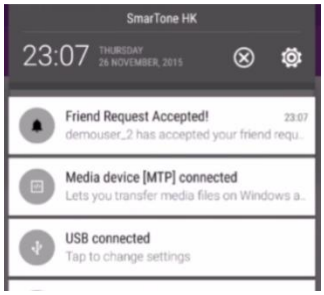
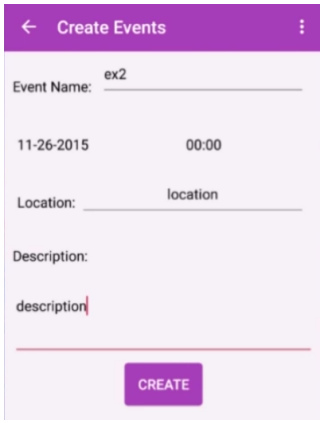


Fig (11)

made-friend success. And accordingly, a notification will be sent to the requesting user immediately saying the user has accepted the request. (Fig(11))

The users can now view "Current Friends" to see each other's username updated in the list.

7. Create Event



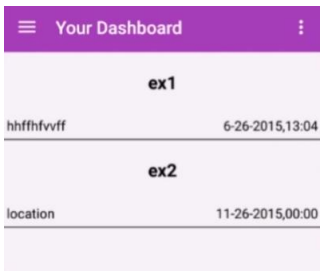
Fig(12)

By clicking on the "Create Events" button on the Navigation Drawer, the user will be redirected to the event creation screen, where the user can enter information of the event to be created including event name, date and time, location and description. (Fig(12))

All fields are compulsory for an event to be created, otherwise a toast will prompt on missing information.

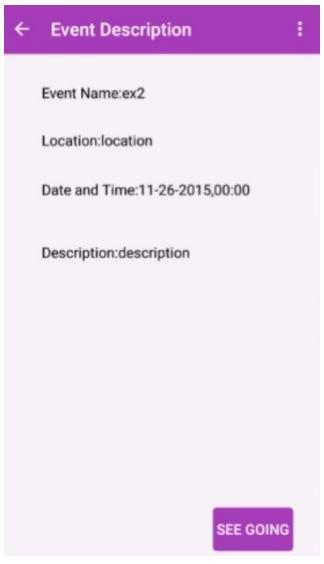
While on tapping the fields of "Event Name", "Location" and "Description", a virtual keyboard will prompt and the user can input any character, digit or symbol; a date-time picker will prompt when the user tap on the fields for editing date and time.

After all successful and valid inputs, the user can click on the "Create" button to create the event.



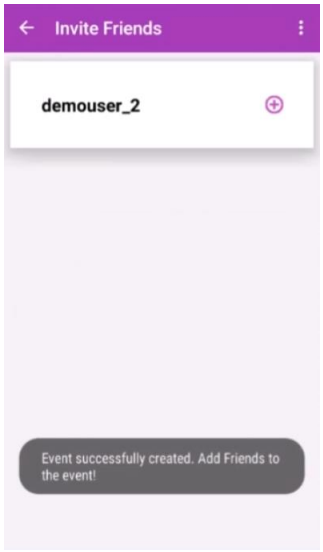
Fig(13)

After finishing the entire event creation process, including the process "8. Event Invitation", the user can then view the event in the updated dashboard(Fig(13)). By clicking on the specific event, the user will be redirected to "Event Description" where the user will find, besides the event info, a "See Going" button that will redirect to a list of current participants of the event(Fig(14)).

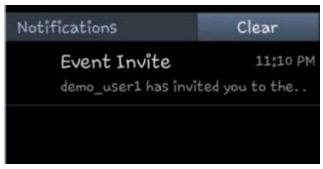


Fig(14)

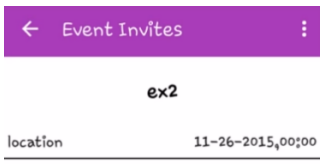
8. Event Invitation



Fig(15)



Fig(16)



Fig(17)

Fig(18)

After clicking on "Create" in "Create Events", the user will be redirected to the screen of "Invite Friends" where the user will see the list of current friends. (Fig(15))

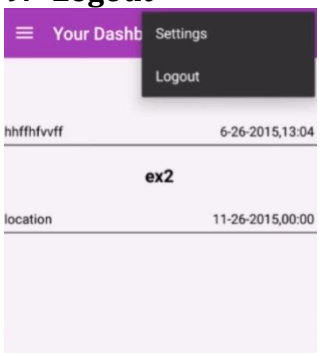
The user can then choose friends to invite to the event by clicking on the add button besides the user name. Upon clicking, an event invitation will be sent immediately to that user and the icon of the add button will change to an icon with a tick signaling a sent invitation.

Then, a notification will prompt on the user's device. The notification specifies the name of the inviting user and the message of "has invited you to the event..." to distinguish itself from any other event invitation or friend request. (Fig(16))

Upon clicking on the notification, the invited user will be redirected to the screen of "Event Invites"(Fig(17)) where the user can click to inspect details of the specific event on the "Event Description" screen.
The invited user can click on the button "Accept" to accept the invitation, "Decline" to reject or "See Going" to see who else is currently going to the event before making a decision(Fig(18)).

On accepting the invitation, the user's dashboard will be updated with the event added to it, on the other hand, the invitation simply disappears on decline.

9. Logout



Fig(19)

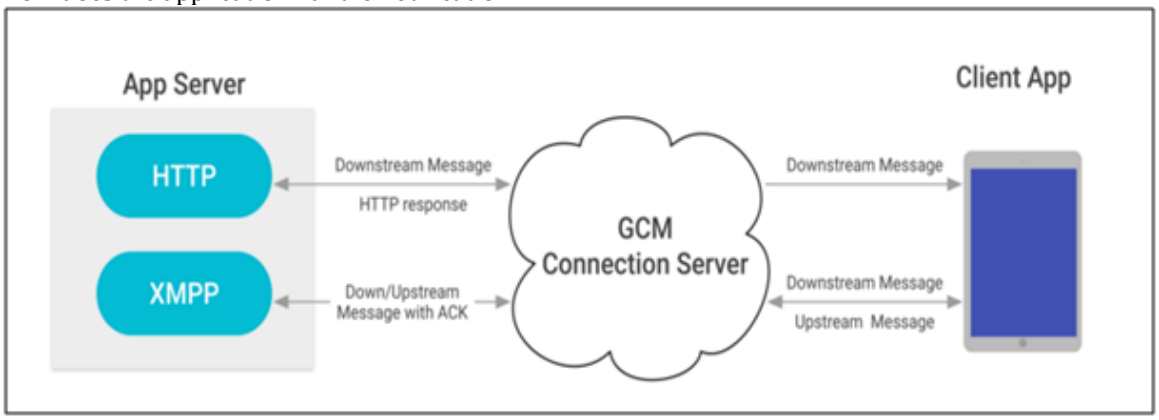
The user can always logout by clicking the logout option after expanding the menu from the button located on the upper right(Fig(19)). However, on leaving or destroying the app process without logout, the user will remain logged-in on re-entering or recreating the app.

After the manual logout, user information will be cleared and the app will be redirected to the login page. The user can then login again with the same or even another set of user info and the app will work just fine.

Implementation

The application MeFree makes extensive use of Google Cloud Messaging (GCM) to handle the notification system within the application. Since the proper and seamless notification system of the application is one of its major features, we will discuss the implementation of it first.

How does the application handle notification?



Fig(20) The Google Cloud Messaging framework

Fig(20) shows how GCM is included with the application. For our app, we have a server running in the backend and GCM connection server that acts a middle man to guarantee successful delivery of notifications.

In the application, we have two services running, namely the MyGCMListenerService and MyInstanceIdListenerService that listen to notifications sent from the GCM connection server. When the application is first installed on a phone, a registrationID token is assigned to the phone by the GCM connection server to uniquely identify each user's phone. The MyInstanceIdListenerService listens to whether the phone has successfully been registered with the GCM connection server when the app is first installed. If the registration is successful, a registrationID token is returned to the client app, which is MeFree, from the connection server. This token is saved using SharedPreferences in the application.

MyGCMListenerService listens for any other notifications from the GCM connection server. When a message is received from the connection server, the type of the message, for example "eventInvite", or "friendrequest", is determined and appropriate notifications are created using Android's NotificationManager. If the app server needs to send data to MeFree, it sends a message to GCM

connection server together with the unique registrationID token of the phone to send the notification to, and the connection server takes over the responsibility of delivering the message to the appropriate user.
By having a unique registrationID token to identify each user and by having GCM connection servers to delivers notifications to appropriate user, the application maintains a seamless notification system.

Registering user:
If the manages to successfully register with the application, the registrationID token of the user is sent to the HTTP app server to be stored in the database for future notifications. If the registration is successful, the app server echoes success, and the user is redirected to the main screen, which is the Dashboard in the application.

Adding a friend and accepting friend requests:
When the user clicks on the "Add a friend" in the navigation bar, the application sends a GET request to the app server to return a list of registered users in the form of a JSON array. When the user , say user A, decides to send a request, for example to user B, the application sends the name of the user B to the app server, which then fetches the registrationID of user B and sends the "friendrequest" message to the GCM connection server together with the registrationID token to be delivered to user B. When user B accepts the request, the application sends a GET request to the HTTP app server together with the name of user A. If the server echoes success, the two users become friends and their relation is stored in the database on the server as well as an internal copy of the relation in a content provider in the application.

Create Events, See Going, Event details:
All the other functionalities of the application use similar procedures to the one described above in adding a friend and accepting a request. When user A creates an event using the application, the details of the event are sent to the HTTP app server and stored. If user A invites user B to the event, for example, notification is sent to user B using the applications notification system and the registrationID token of user B stored on the app server. Seeing the going list just queries the HTTP server to return a list of users attending the event specified in the form of a JSON array. The GET request is done using Volley from within the application and the results are displayed in the interface. Fetching details of an event is similar and involves sending a GET request to the HTTP server and displaying the information in the application that is returned from the server.

Results and Evaluation

MeFree started carrying ambitious ideas on its functionalities, offering a variety of functions that would make the organization and management of events even more comprehensive that it is now with the current implementations. These features are to be discussed in Section 5. Future Development. We have asked a number of people to evaluate our app and below we have summarized the strength and weaknesses of our app that were pointed out.

With limited time, delivered functionalities in MeFree, though not completely as promised in the proposal, are credible with the fundamentals - and indeed way more than the mere essentials - implemented.

The notification system with the aid of GCM is swift and immediate upon test and offers guaranteed user experience with almost no delay. In addition to that, the local content providers work seamlessly under most circumstances with the server side database and offer a wide range of self-updating local contents such as the lists of current friends and participants of the events. Besides functionalities, the UI is friendly and renders a sense of professional product with the use of widgets like the navigation drawer, recycler views and custom icons and graphics.

However, despite all the hard work, there are inevitably downsides of the app besides those lackings to be mentioned and deployed in future development that worth attention in the current deliverable.

Firstly, the event invitation to friends after creating an event should be applicable not only as an one-off invite immediately following the event creation, instead, event should be made editable to the host any time so that a friend could be invited to the event even after the event is created for some time. This feature was discussed and proposed during development, however, due to the lack of time and the fact that the implementation of the feature would result in quite some changes to the program in later stage of development, this was not delivered in the end before submission.

Secondly, the local content provider does have a bug as the data are not cleaned after user logout and the current friendlist as well as the dashboard would be falsely updated on a new user login. The problem arises as a result of incomprehensive testing that the test case with multiple users on the same device without reinstalling the app was not tested, leaving behind this error which should have definitely been fixed.

As a little conclusion, the development process went smooth with dedicated effort invested. However, the presence of errors suggests improvements in planning and time management. The schedule of development should be set with more concrete milestones and indicators, for example setting individual deadlines for different concrete functionalities so that the process of development could be made more traceable and thus reduce the chance of missing any expected functionality. Moreover, the design of different modules of the app and the development sequence of which should be better reviewed and designed so that when there are changes to the code in later stages of development, the effect on written codes should be made as minimal as possible.

Future Development

In the future, using a more reliable and stable server with larger bandwidth, improving data handling, developing a more user-friendly User Interface and new functions are going to be done in order to attract new users.

With the increasing amount of users, MeFree need a stable network to be maintained so as to transmit huge amount of data. Keeping everything synced without any delay is the most difficult part at this moment with the complicated data handling. Simplifying the data handling and developing a better User Interface with some new functions are the other aspects that are going to be improved in the future. Global Events invitation and Event Nearby functions are the main functions to be developed. With these functions, users can check all global events and the events nearby that are open to all by using the google map. More than that, MeFree is going to be developed as a friendship-related theme. It is because functions like creating friend groups, sharing and friends-viewing are in designing process. They will provide user to create own friend groups to invite, share the events they want their friends know by social network platforms or email and view the other friends' profile.

MeFree will be updated to include these features in the future.

Conclusions

With a variety of functions provided, MeFree will become a useful and popular application in daily life. It can be used in various activities such as wedding parties, sports trainings and educational courses. It will change the users' habits by the unique and simple functions it provides. MeFree benefits not only the participants, but also the event holders. Although it is a hard job to provide a long term and stable service, with development, MeFree will bring a lot of new features to the users in the future.

References

https://developers.google.com/cloud-messaging/gcm

http://developer.android.com/training/index.html

Trivia

MeFree Demonstration video: https://youtu.be/KcxfnMiNv8Y