Project Report:

Here You Go! (HuG)

Sushant Paygude
Ajinkya Shendre
Pankaj Tekwani
Intekhab Naser
Kaustubh Agrahari
Huma Saudagar

Introduction:

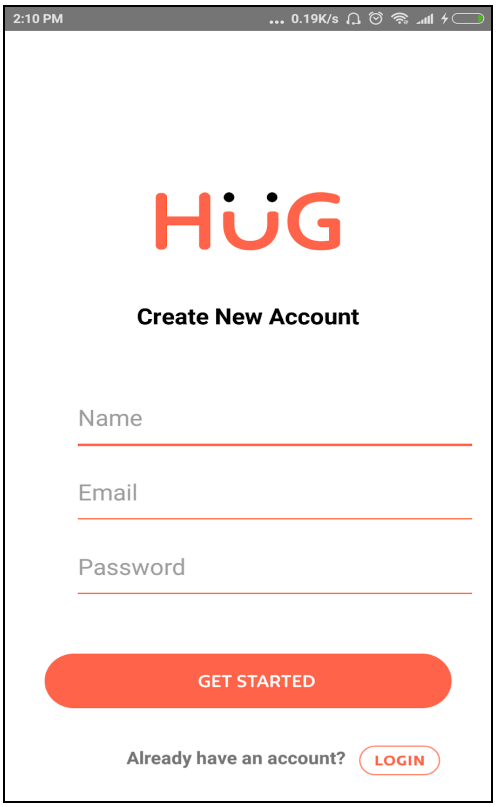
HuG can be described as a multi-utility application that provides a list of nearby feeds to the user based on his current location. We developed an application which helps the user to find his nearby feeds. The app provides user friendly environment in order to access various events based on his preference .The app tracks the user's current location and using it, provides the interesting feeds such as nearby Restaurants, Events, ATM, Banks, Gas Station.

Design:

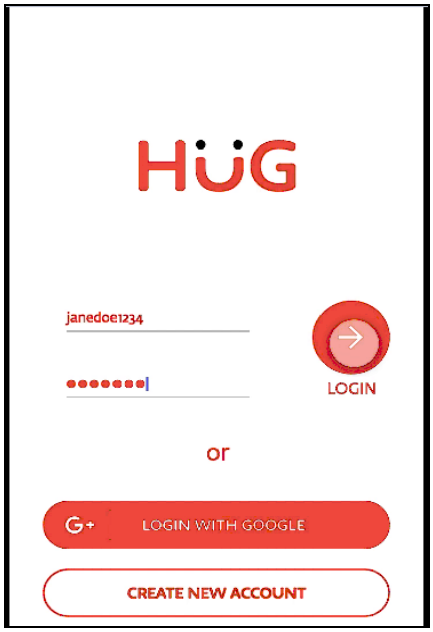
Front End

We proposed this idea to aggregate features of different apps like Google map, yelp, and event finder. The app provides user with ability to access all these features in single platform. The project contains following features:

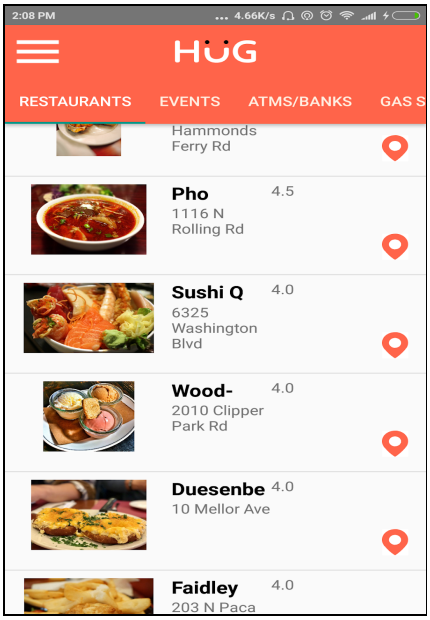
1. User Registration - The user will be able to register to the app by creating new account.



2. User Login - After successful registration, the user will be able to login to app using his username and password. An alternative with login using google account is provided for user to get access to app instantly.



3. Feeds - Once logged in, the user will be able to see nearby interesting feeds such as nearby restaurants that are open, ATM or banks that are more reachable to user, events that are organized in near future, reachable gas station.



4. Direction guidance - With all the above feeds, an additional facility for the user to find the route for event he is interested in, is provided. The user will be able to see the route for the selected direction and will be able to navigate himself for the destination using our navigation feature.

5. WishList/ ToDo list- A feature to remind the user to purchase items at his convenience. The user will be able to add the items that he wants to purchase in this list. This list will work as reminder for the user. Whenever user reaches a location where a store to any of the items in the list is more reachable to him, then he will be notified by app showing the respective store/mall/shop for the desired item to be purchased. Through the notification, the user will be able get the route and can navigate himself to the destination.

Web-Services and Database:

Web-Service:

Our mobile app calls the php web service and this webservice speaks to database over the internet using Volley (an Android library to make POST and GET calls).

The webservice has the facility as below features:

1) Login/Signin
2) Register/Signup
3) Update User Data

Login / Signin:

HUG connects to the web service hosted server over internet. The login details (i.e. Username and Password) are taken from the user and then using Volley library the app makes POST call with these parameters. The php service takes these parameters and queries executes query to the local database. The response from the database for this query is successful if the login username and password were existing in the database else database responds with no data. Depending upon the response from database server the webservice response to the user. If response was successful then it returns the username and emaild of the user else it returns, “No User Found”. In both the cases the response from webservice is in JSON String.

Call to the server:

Username, password, action= login

Response from server:

If successful:{


```
  "users": {
    "Success": 1,
    "User": "kk",
    "E-mailId": "kaus1@umbc.edu"
  }
}
```

If failed:

```
{
  "users": {
    "Success": 0,
    "Message": " No User Found.",
  }
}
```

Register/Signup:

When user wants to register an account with us then the app “HUG” takes the details from the user and sends them as parameters to the web service by making POST call. This POST call is then listened by the webservice and it executes the query with the passed parameters in the POST call. Now we have two cases either it is saved in database or not. If it is saved in the database then it is successful if not then the error message is returned. The webservice reads these results of this action from database and responds to HUG either Successful or Error message in JSON String format.

Parameters for registration:

Username, password, email_id, action = register

Response from server:

If successful:

```
{
  "users": {
    "Success": 1,
    "Message": "Account created for the user kk."
  }
}
```

If failed:

```
{
  "users": {
    "Success": 0,
    "Message": "Error: " <Respective Error Message from server>
  }
}
```

Update User Data:

This module was made with an idea to save users photo if the user creates manual account in the previous module. Just like the method of registering account, in this module takes collects the photo from the POST call and passes to the database to save it against the respective user. The photo must be in base64 format which is then converted to binary while saving in database and when it is queried the photo is again converted back to base64 format. This conversion takes place at the server end.

Parameters for registration:

Username, password, action = update

Response from server:

If successful:

```
{
  "users": {
    "Success": 1,
    "Message": " Photo updated for the user: kk."
  }
}
```

If failed:

```
{
  "users": {
    "Success": 0,
    "Message": "Error: " <Respective Error Message from server>
  }
}
```

Database:

We have used MySQL as a database to store data. The database is deployed on the same server the web service is hosted on.

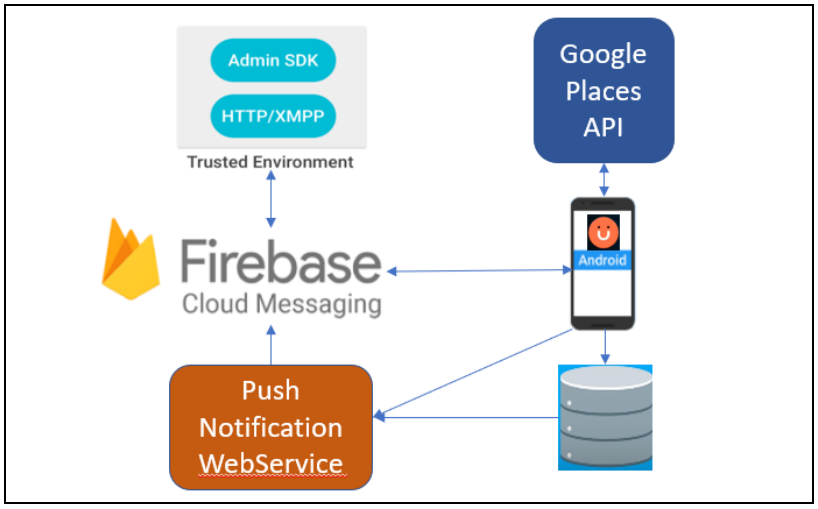
To create database follow below:

1) Create a database name db_hug.
2) Execute below script on it

```
DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `idusers` char(36) NOT NULL,
  `userName` varchar(45) DEFAULT NULL,
  `passWord` varchar(45) DEFAULT NULL,
  `photo` binary(255) DEFAULT NULL,
  `emailId` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`idusers`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Firebase for Push Notifications:

In order to enable push notifications for HuG, we used Firebase Cloud Messaging (FCM) service which is a cross-platform messaging solution that lets us deliver messages reliably. By using FCM, we are able to notify a client application that relevant data is available to be used. We can send notification messages to drive user re-engagement and retention in which a message can transfer a payload of up to 4KB to a client application.



WorkFlow:

An FCM implementation includes two main components for sending and receiving:

- A trusted environment such as Cloud Functions for Firebase or an app server on which to build, target, and send messages.
- An Android client app that receives messages.

The steps can be summarized as follows:

- First, we connected our app to firebase. This is done by creating a project in Firebase console and linking it with our application in Android Studio.
- Then we added FCM to our app. To add FCM to our application, we add the corresponding google-services.json file to the app module in Project view and modify a bunch of dependencies.
- Once, the dependencies are set, we write code to access the device registration token. On initial startup of our app, the FCM SDK generates a registration token for the client app instance. We access the token's value by creating a new class which extends FirebaseInstanceIdService and add the service to our manifest file.
- In that class, we call getToken() within onTokenRefresh(), and log the value in the respective database we created to store tokens. The onTokenRefresh callback fires whenever a new token is generated, so calling getToken in its context ensures that we are accessing a current, available registration token. FirebaseInstanceID.getToken() returns null if the token has not yet been generated. After we have obtained the token, we can send it to our database server using an async task. The first few rows in the database are seen below:

	ID	Device_ID	Token
Edit Copy Delete	1	My_ID	Testing_Token
Edit Copy Delete	2	Intekhab	cIcUkXHOcAg:APA91bGUnEP_kwP-cBpJDf8SGJT5Rj-mWgDvRe...
Edit Copy Delete	3	Intekhab	c3UlwN795Jg:APA91bHfV2IzZ2Kv9K2ViwbNDp56ZrgUOk-5H4...
Edit Copy Delete	4	Intekhab	eV4HxBJW-DE:APA91bEI-Xe-cYvmgVw7Qig0uBPiNx5XT8SRNd...
Edit Copy Delete	5	Intekhab	c2qaTjjX5F8:APA91bGp61bKESZvRZmV9whPRwfW41KVGK78G_...

- Next, we need to handle the messages. If we wish to do any message handling beyond receiving notifications on apps in the background, we create a new Service that extends FirebaseMessagingService and declare the service in our application's manifest. This service is necessary to receive notifications in foreground apps, to receive data payload, to send upstream messages, and so on. In this service, we create an onMessageReceived method to handle incoming messages.
- In this method, we use NotificationManager, NotificationCompat and NotificationBuilder to define how to build and display the notification message. With the Pending Intent we define the action which needs to be performed on clicking the notification.
- Once the client app is set up, you are ready to start sending downstream messages by calling Firebase console.

Calling Firebase console:

In HuG, we have a ToDo List and we pick items from it and send it to the Google Places API along with our location in terms of Latitude and Longitude. From this API, we receive a list of stores or restaurants in the nearby area where we can obtain the item from the ToDoList. We fetch the most relevant place from this Google Places List and send it to our PushNotifications WebService. This webservice constructs a request and sends to Firebase console, which in turn pushes the notification to the respective devices using HuG.

The components in this can be explained as follows:

- We write a service called 'LocationService' to keep running in the background to access the device's current location.
- We access the current location by calling 'LocationService' method from main activity:

  mylocation = **my_loc_service**.getlocation();
- We then access the items from the ToDoList as:

  ArrayList<String> todoArrayList = **tinyDB**.getListString(Utilities.*TO_DO_LIST_STRING*);
- Then a query is constructed to send to Google Places API as follows:

```
String api_url = getResources().getString(R.string.google_api);
<string name="google_api">https://maps.googleapis.com/maps/api/place/textsearch/json?</string>
String parameters = "";
parameters += "location=" + mylocation[0] + "," + mylocation[1] + "&rankby=distance&type=" +
type + "&query=" + query + "&key=" + getResources().getString(R.string.key);
<string name="key">AIzaSyBer4uEGhaM6RK9eYMz4OoptFg3cxU9bGk</string>
```

- The corresponding most relevant place obtained from the Google Places List is then sent to PushNotifications WebService. This WebService uses API_ACCESS_KEY and Authorization Key to connect with Firebase Console. Also, it fetches all the tokens from database to send notification message with Place Name and Address to all respective devices.

- The code snippets in the webservice can be seen as follows:

```
$url = 'https://fcm.googleapis.com/fcm/send';
$message = array(
    'title' => $data['mtitle'],
    'message' => $data['mdesc'],
    'channelid' => $channelid,
    'time' => $time,
    'subtitle' => $data['msubtitle'],
    'tickerText' => '',
    'msgcnt' => 1,
    'vibrate' => 1
);
```

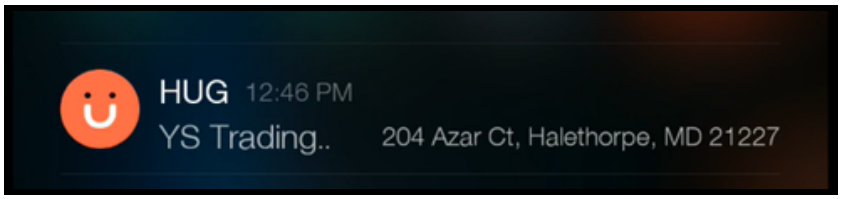
```
$headers = array(
    'Authorization: key = AAAAebTzd5Q:APA91bGGxTuP9SSmbGRpuOZM5BwwxxEgOgs1BNrT-
    wfFwORPWqVz971clbBEla3uBhXSbZb4Yc3NhTFnjGfSVDHiviArMzU0SqHIAZm2aqhmwaZvbNvFSB2bFcBwMpN6NgooSy6f51my',
    'Content-Type: application/json'
);
$fields = array(
    'registration_ids' => $reg_id,
    'data' => $message,
);
$ans = useCurl($url, $headers, json_encode($fields));
```

```
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
// Disabling SSL Certificate support temporarly
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
if ($fields)
{
    curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);
}
// Execute post
$result = curl_exec($ch);
```

```
$conn = mysqli_connect("localhost", "root", "", "fcm_test");
$sql = "SELECT Token FROM devices";
$response = mysqli_query($conn, $sql);
$tokens = array();
if(mysqli_num_rows($response) > 0)
{
    while($row = mysqli_fetch_assoc($response))
    {
        $tokens[] = $row['Token'];
    }
}
```

Notification:

- Finally, the notification can be seen as follows:



APIs:

1. Ticketmaster:

It is basically a primary ticket outlet which provides bunch of rest api endpoints that allows developers to develop various apps as per their needs. The developer only has to create an account with them in order to get their access api key to get access to all the available functionalities provided by the ticketmaster.

Following are the few examples of common use cases that most developers build apps against Issues:

- Searching events based on a keyword in a certain location(using the latitude and longitude of the particular location).
- Getting events for a particular artist or by venue.
- Getting high resolution images of the particular images.
- Search events of a certain genre in a particular location.

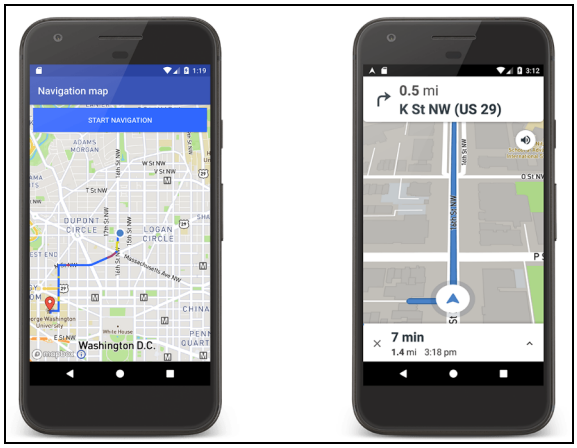
Rest Endpoint call for getting events:
https://app.ticketmaster.com/discovery/v2/events.json?countryCode=US&apikey={apikey}

Response Structure (json object):

_link (Object) - links to data sets
_embedded (Object) - Container
events(array) - lists of all the events
page(Object) - information about current page in data source for pagination

2. MapBox SDK:

Mapbox Navigation SDK is a basically a directions api which provides timed navigation instructions. It takes in the source and destination location to display the set of available routes and highlights the best possible route from source to destination. There are many more features that Navigation SDK provides like traffic routing, off-route detection, real time route progress information etc. In order to use the Mapbox sdk in their android app, the developer must signup to the Mapbox account to get their public access token key. The advantage of using mapbox sdk as the map tool as compared to available map tools is that it provides transition to navigation from the routes in same activity.



Mapbox Navigation Activity

3. Yelp

The Yelp API gives access to thousands of business of different categories. We created an account on the Yelp development console to get an API key which can be used within our application. We called a GET request to the API with the following configuration:

1. Passed the user's current location
2. Passed the radius limit to get the list of nearby restaurants.

REST GET call to the Yelp API:
https://api.yelp.com/v3/businesses/search?latitude=39.260580995402094&longitude=39.260580995402094&radius=10000
Headers: Authorization : Bearer {API KEY}

4. Google Places

The Google Places API is used to find the nearby gas stations and ATMs based on the user's current location. We also used this to get location of a nearby store that contains any item from the To Do list. This nearby store along with its address is sent as a push notification.

A REST GET call to get nearby gas stations using Google Places API:
https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=39.260501,-76.699731&radius=1500&type=gas_station&key={API KEY}&pagetoken=null

Conclusion:

The main objective of the project was to develop a multi purpose application that collects individual functionalities of different APIs and collectively brings them together. The app makes maximum use of the current location of the user and collectively displays the information relevant to the user. We were able to successfully achieve our scope for the project.

Future Work:

1. Filtered search for the events, restaurants etc based on genre, circumradius etc.
2. Categorize the items from the ToDo list.

References:

1. https://developer.android.com
2. https://stackoverflow.com
3. https://firebase.google.com/docs/cloud-messaging/
4. https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/
5. https://www.mapbox.com/help/android-navigation-sdk/
6. https://github.com/drfonfon/android-geohash