Bolan Peng
CS496 - Fall 2016

Cloud & Mobile Development
Final Project Report

Video Link:

http://www.dailymotion.com/video/x5442s7

Note:

- Cloud code is inside `Diary/backend/` directory
- Mobile code is inside `Diary/app/` directory

Overview:

My final project is a diary application on Android platform. The frontend provides basic functionalities for users to sign in to their own accounts, add new diaries that only they can see, make change to the diaries and delete the diaries. The backend is implemented using Google App Engine with Firebase Authentication, which is used to support all the frontend functions.

Account System:

As mentioned in the Overview, I used Firebase Authentication to manage my application's account system. Firebase is a backend system that has many features, including real-time database, storage, etc. I used only the Authentication feature to authenticate users and keep track of the users' states, after the users logged in using Google Sign-In.

First time users will need to register with the application. The application identifies first time users by grabbing their user ID from Firebase Authentication. And then the user ID will be send to an Endpoints method to check against the database. If the database cannot find a user with the corresponding user ID, the user will be send to the registration page. Otherwise they will be redirected to the main page.

Restful API (Endpoints):

I built my application's backend using Google App Engine's Cloud Endpoints. The Google Cloud Module works very well with the newest version of Android Studio. I am able to generate the API backend right inside my Android application.

I created two entities, one is MyDiary, the other is called Users. MyDiary has a diary ID, title, content, and a user ID for the user that created the diary. User should be able to have many diaries, while each diary can only be created by one user.

The following are the API methods:

- GET: *listDiaries*
  ```
  GET https://diary-151121.appspot.com/_ah/api/myApi/1/mydiary/[user_id]
  ```
  - This method takes a user ID, and returns a Collection of MyDiary objects.
  - The user ID is used to filter the query, so the calls to this method only returns the diaries with the corresponding user ID. Or in other words, the logged in user can only see his/her diaries.

- POST: *insertDiary*
  ```
  POST https://diary-151121.appspot.com/_ah/api/myApi/1/mydiary
  {
   "diaryId": [diary_id],
   "title": [diary_title],
   "content": [diary_content],
   "userId": [user_id]
  }
  ```
  - This method takes a MyDiary object, and returns a MyDiary object.
  - The method checks MyDiary object's diary ID to make sure there is no duplicate diary object in the database, and then saves the diary. Users can perform frontend operations on the returned MyDiary object, or disregard it.

- POST: *insertUser*
  ```
  POST https://diary-151121.appspot.com/_ah/api/myApi/1/users
  {
   "userId": [user_id],
   "username": [user_name]
  }
  ```
  - This method takes a User object, and returns a User object.
  - The method checks User object's user ID to make sure there is no duplicate user object in the database, and then saves the user. Users can perform frontend operations on the returned User object, or disregard it.

- POST: *verifyUser*
  ```
  POST https://diary-151121.appspot.com/_ah/api/myApi/1/verifyUser/[user_id]
  ```
  - This is a utility method to verify if a user already exists in the database.
  - This method takes a String ID, and if user does not exist, returns null; otherwise it returns a User object.

- PUT: *updateDiary*
  ```
  PUT https://diary-151121.appspot.com/_ah/api/myApi/1/mydiary
  {
   "diaryId": [diary_id],
   "title": [diary_title],
   "content": [diary_content],
   "userId": [user_id]
  }
  ```
  - This method takes a MyDiary object, and returns a MyDiary object.
  - The method checks MyDiary object's diary ID to make sure there is such entry in the database, and then saves the diary. If no diary object in the database has the corresponding diary ID, the method will throw a NotFoundException.

- DELETE: *removeDiary*
  ```
  DELETE https://diary-151121.appspot.com/_ah/api/myApi/1/diary/1
  ```
  - This method takes a Long ID, and returns void.
  - The method checks the database using the ID to make sure there is a corresponding Diary object with that diary ID, and then removes the diary from the database.

Mobile Feature:

When a user registers with the application, they will be able to select an image from their mobile phone's gallery, and use that image as their user image.