

**CSC 392/492: Mobile Applications Development for Android II****Assignment 1 – Rewards (400 pts)**

Uses: Multi-Activity, Location Services, Geocoding, Internet, APIs, Images, Camera, Gallery

A) App Highlights:

- This app allows users within a company or organization to award each other “reward points” (and comments) as a thank you or a commendation for their performance on a task.
- Users can create a profile for themselves and interact with the other users. Users can later edit/update their profile data and delete their profile.
- Users start with a fixed number of points that they can award to other users.
- When a user awards points to another user, their point value is reduced by the amount they give. Once a user is out of points to give, they can no longer add new rewards until their points are renewed. Point renewal is not within the scope of this project.
- Assume the process of resetting the points-to-award value (and the actual act of rewarding the employee) are out of scope for this project.
- You will need to create classes to represent the user profiles and the reward assignments.
- We will be using an API to as the “back end” which holds user and reward data.

B) Behavior Use Cases and Activities: *Activity/behavior diagrams follow (in section C)*

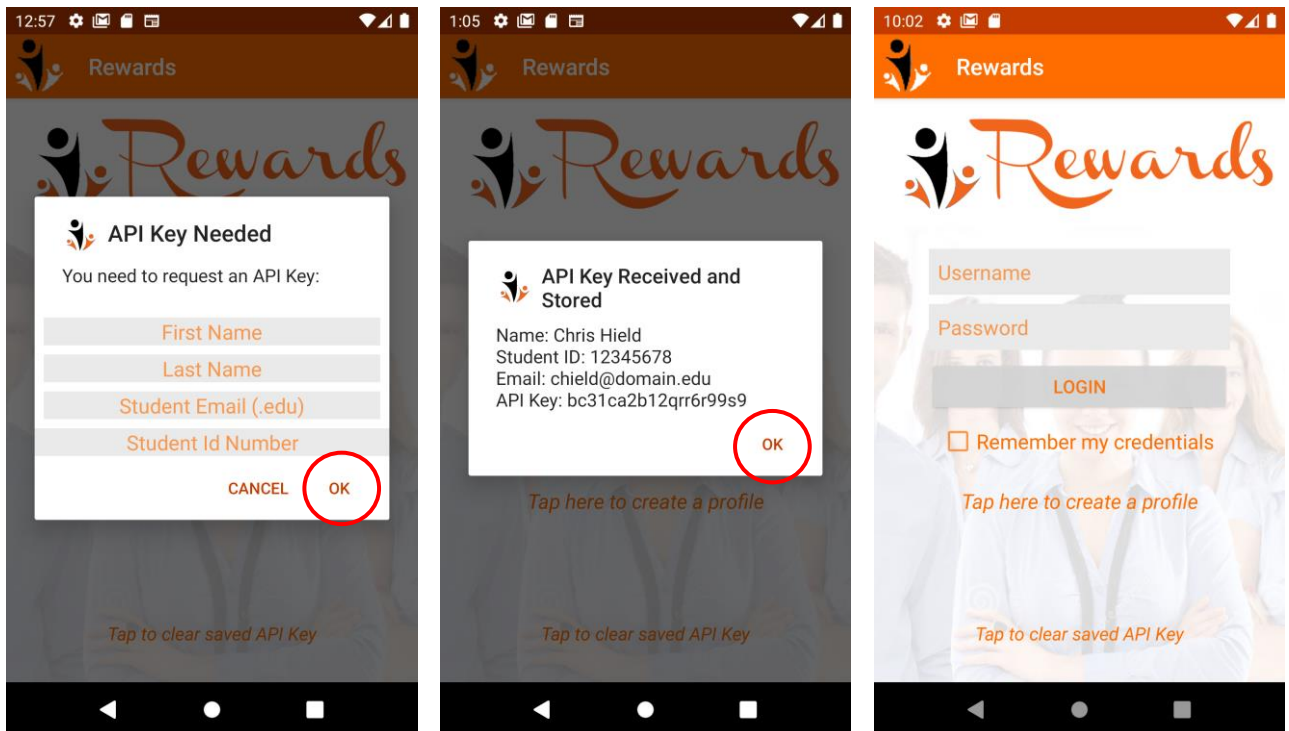
- The first time the app is run, it should ask the *student* for their **first name**, **last name**, **email** (must end in “.edu”), and student **id** number. See diagram “a” in the next section.
 - This data will be **sent to the API to register** the student and **returns their API Key** (a long alpha-numeric string like “bc38cd2b12qrr2r89s9”). The data provided by the student and the resulting API Key should be displayed to the student as confirmation. See diagram “a” in the next section.
 - This API Key must be included with every Rewards API call (besides the API Key request call).
 - The student should only need to provide this information the first time the app is run, so the key must be saved between app runs so it is always available.
 - The saved API Key *can* be cleared from the app if desired via a link on the Login Activity. Once this is done, the app should redisplay the request for the student’s data just as it did the first time the app was run. See diagram “a” in the next section.
- Users can **create a new profile** (via the Create Profile Activity) with data including a username, password, first & last name, department, position, a short self-summary, and a photo. Successful account creation logs in the user and displays their Profile Activity. The Create Profile Activity is accessed by clicking a link on the Login Activity. See diagram “b” in the next section.
- **Users login** (via the Login Activity) by providing their username and password. A **checkbox** can be checked to save the credentials so that they are auto populated whenever the app is started. See diagram “b” in the next section. Upon login, the Profile Activity is displayed.

- The data displayed in the Profile Activity includes first & last name, username, location, awarded point total, department, position, points available to award, a short self-summary, a photo, and a list of awarded points. See diagram “b” in the next section.
- From the Profile Activity, the user can open the Edit Profile Activity via Options Menu. The Edit Profile Activity allows the user to edit any profile data except the username (username cannot be changed). See diagram “c” in the next section.
- From the Profile Activity the user can delete their profile via the Options Menu. Upon deletion the app closes. The profile no longer exists and cannot be used unless recreated. See diagram “d” in the next section.
- From the Profile Activity, the user can open the Leaderboard Activity (showing all other users, **sorted by awarded point total**) via Options Menu. From the Leaderboard Activity, an individual user can be selected and then displayed in the Award Activity. From there, rewards points (and a comment) can be awarded to the selected user. See diagram “e” in the next section.

C) Application Behavior Diagrams:

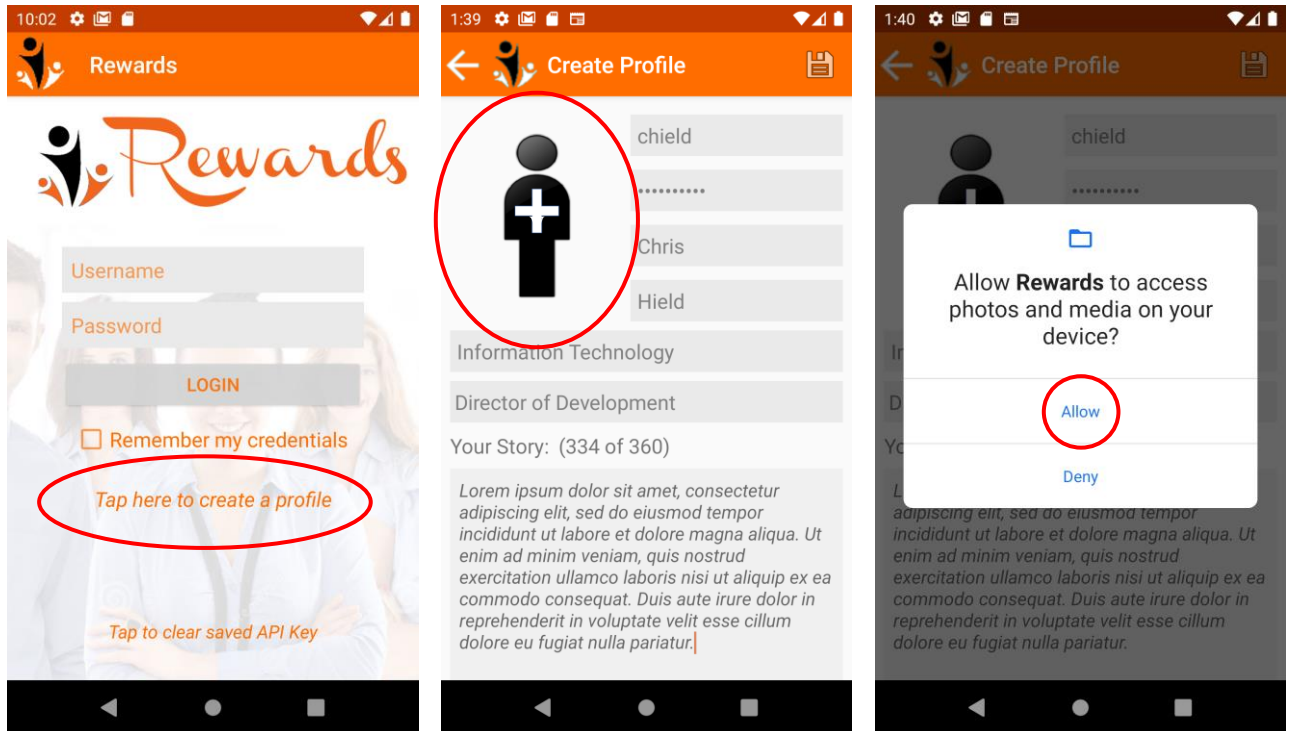
*Note, the first time the app runs, it should ask for **location permission** (before asking for the API Key)*

a. Request API Key (Displayed the first time the app is run, or if the API Key is cleared)

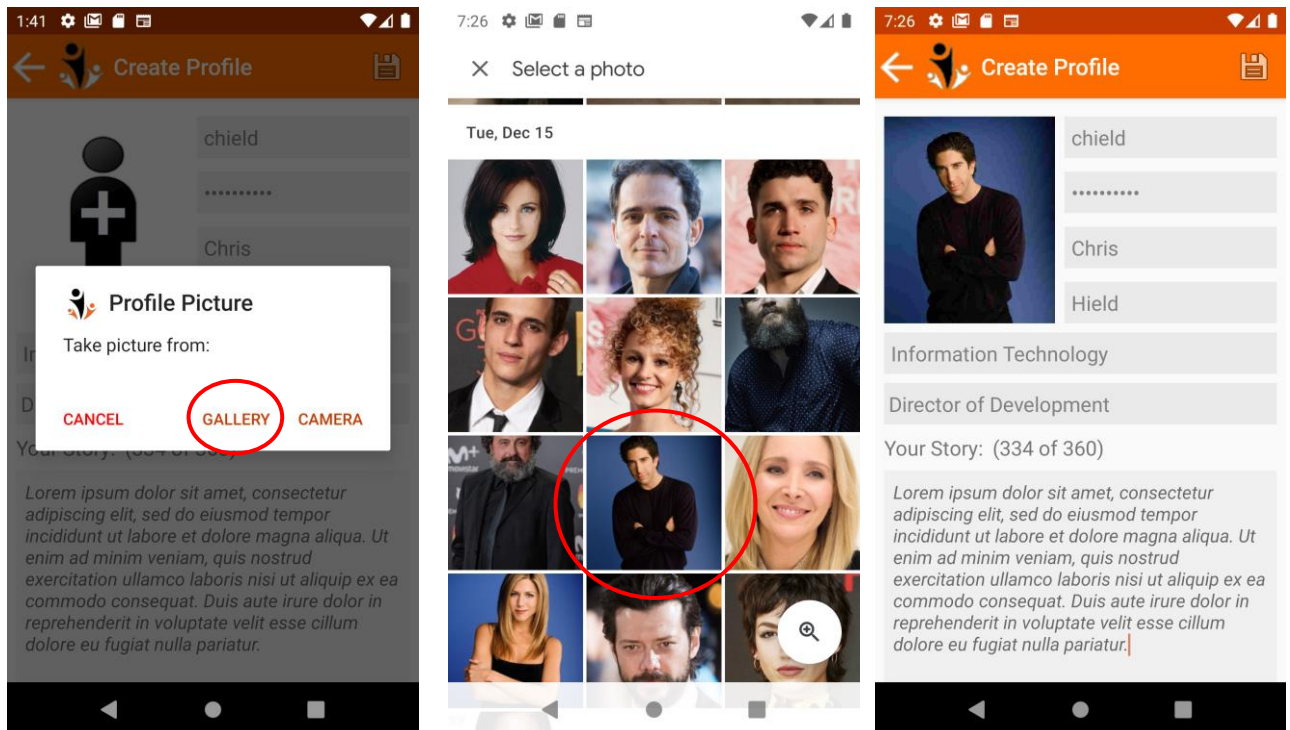


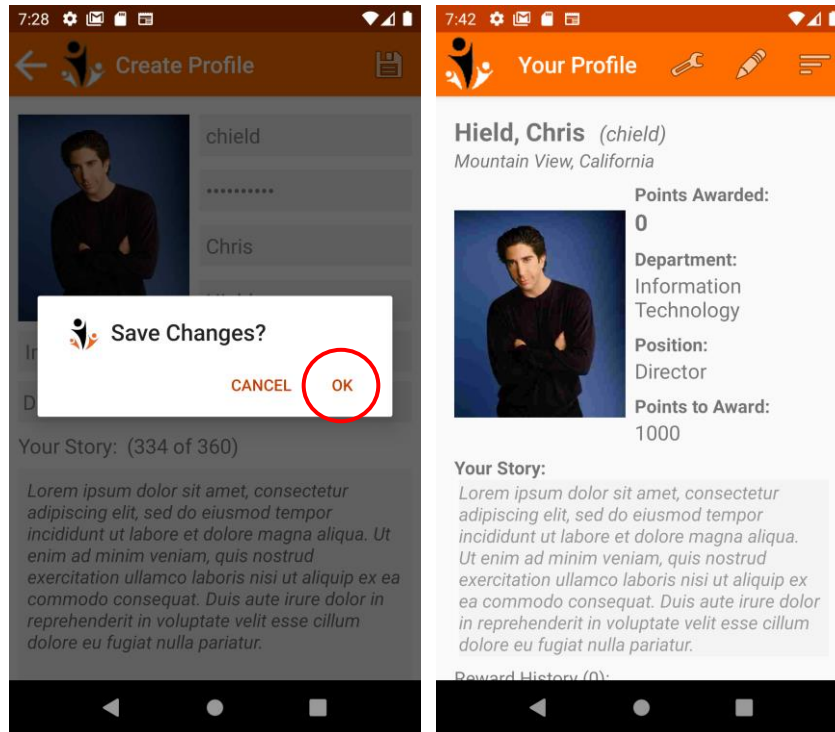
- All **four** data fields must be provided.
- The API key should be automatically saved by the app – the dialog is only used to let the student that the API Key request was successful.
- **Invalid requests** (missing data, non-edu email address, invalid student id, etc.) will be rejected by the API. The rejection text should be displayed to the student, and the data dialog should be redisplayed.
- The saved API Key can be cleared by tapping “**Tap to clear saved API Key**”

- b. **Create Profile** (Note, the first time the app needs to access the camera or gallery, it should ask for file permissions):



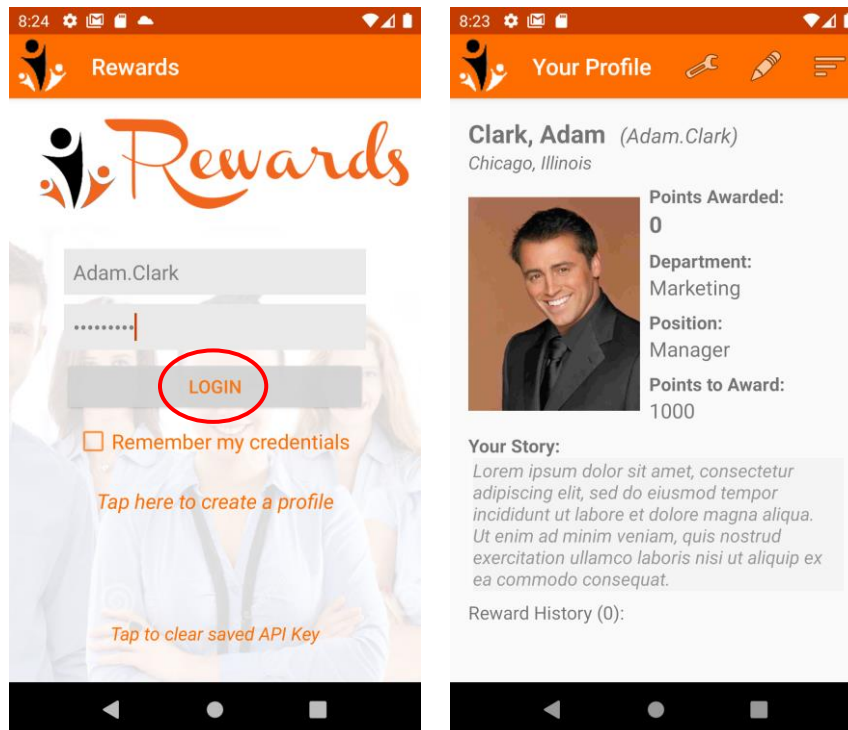
- All Profile Activity content may not fit onto the screen so the activity should be **scrollable**.
- All data fields must be provided.



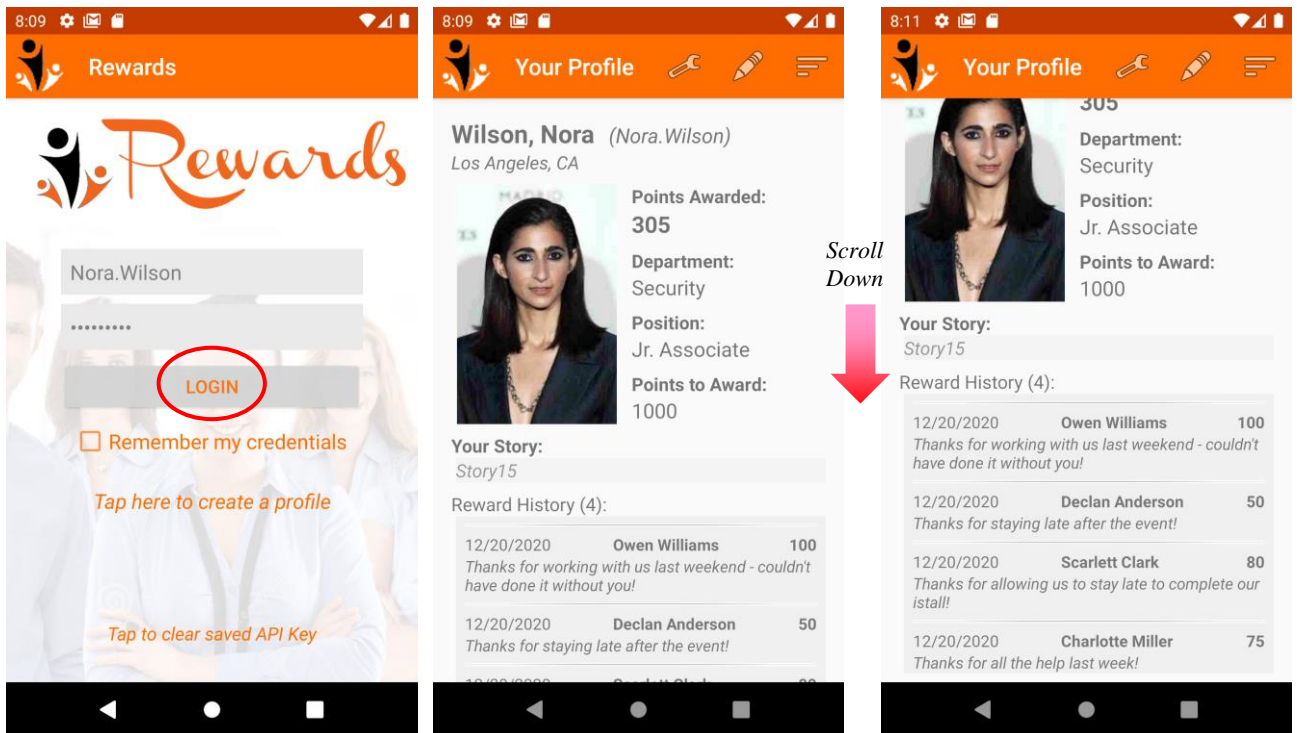


c. User Login

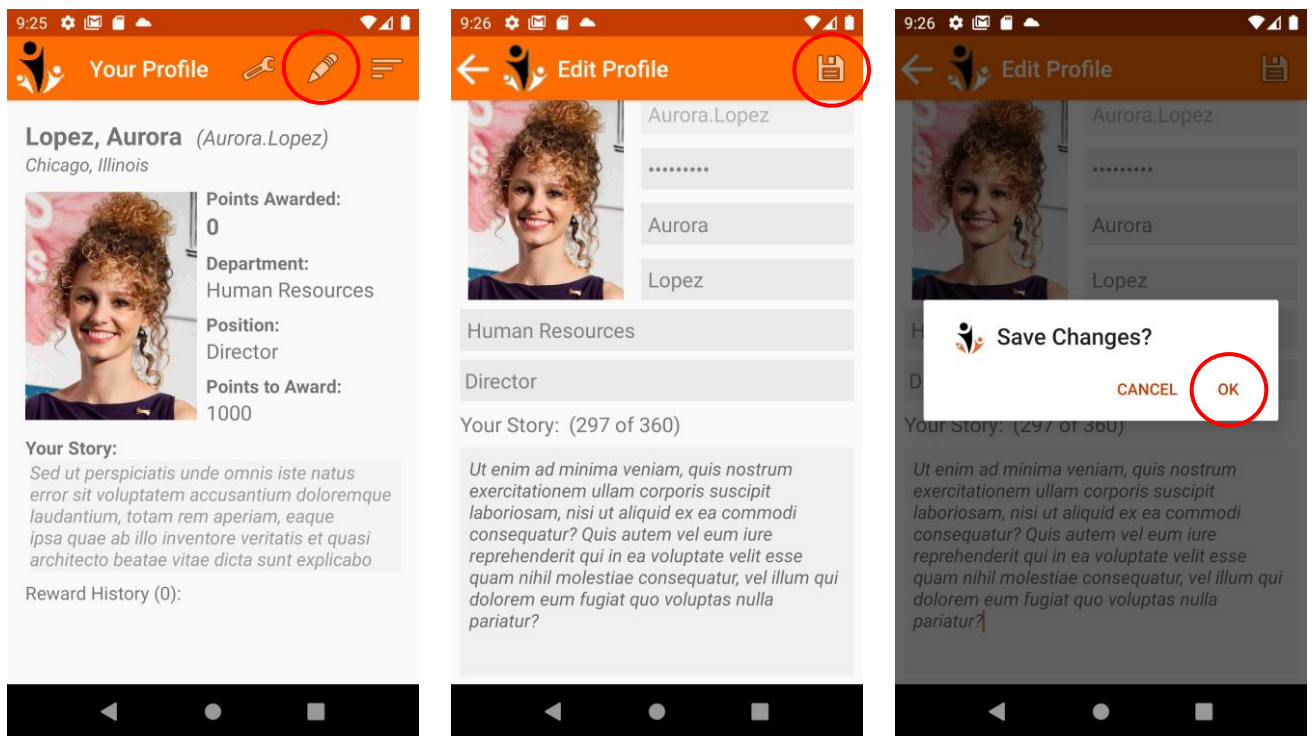
- With no Rewards yet

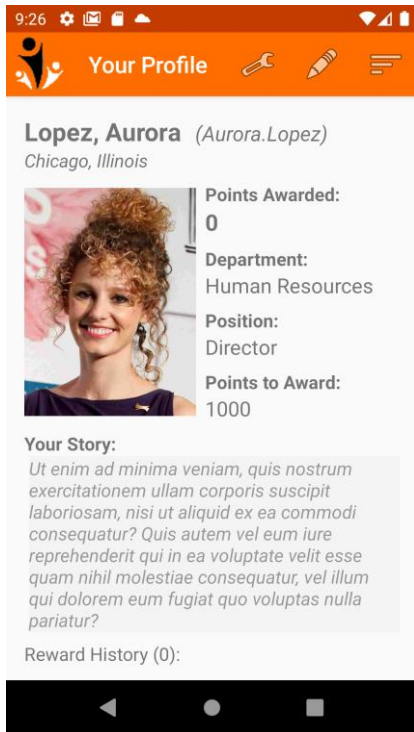


- With existing Rewards

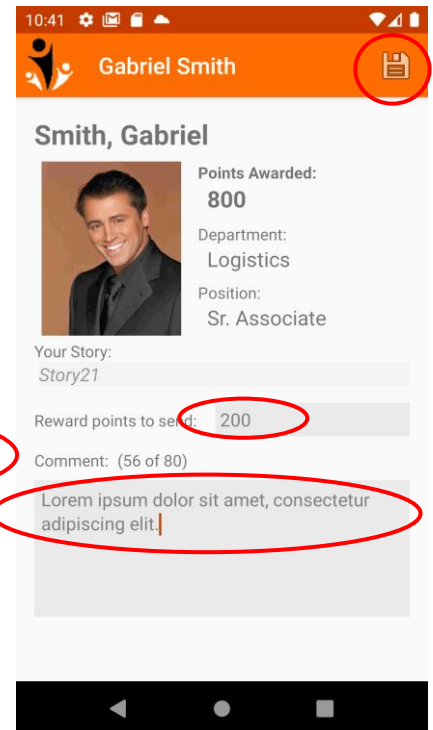
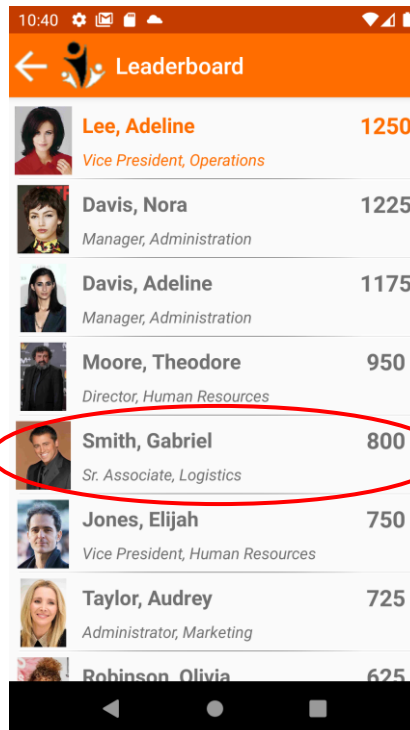
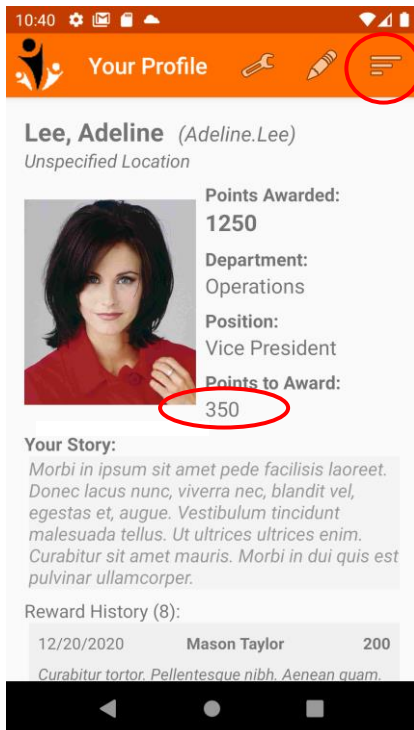


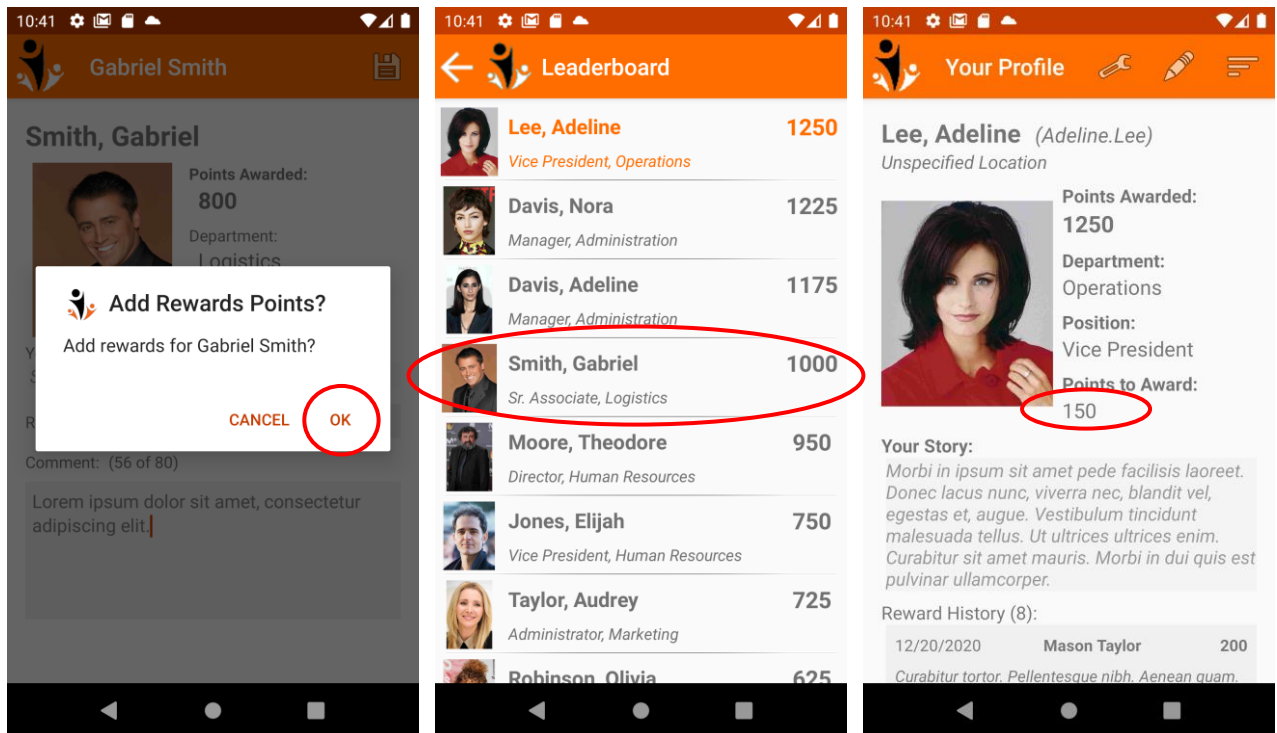
d. Edit Profile (Note, all profile fields are editable except the username – usernames cannot be changed):



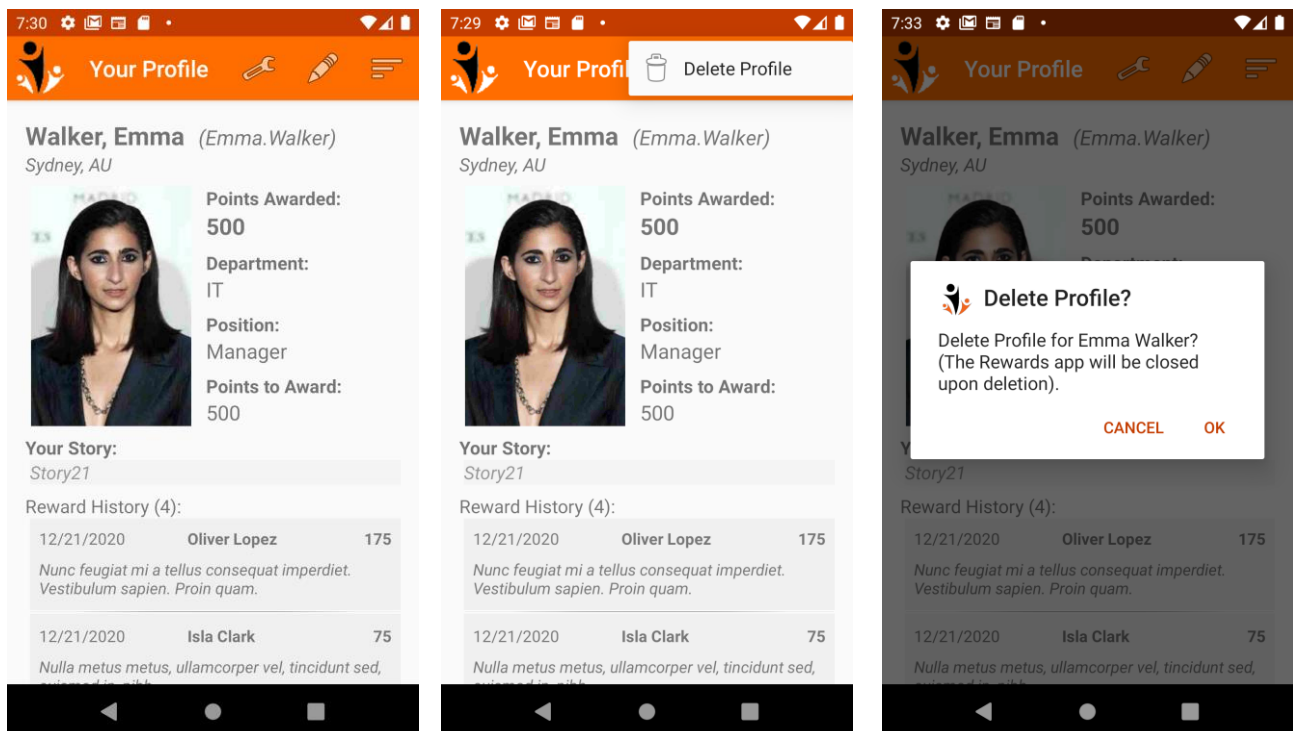


e. Add Rewards



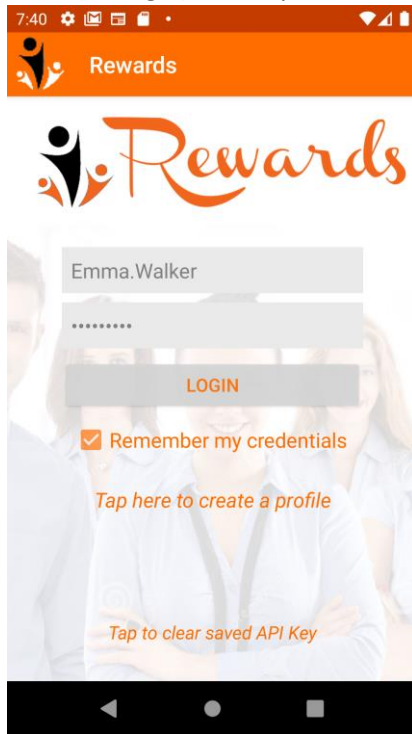


f. Delete Profile



D) Activities

Main (or Login) Activity



7:40

Rewards

Emma.Walker

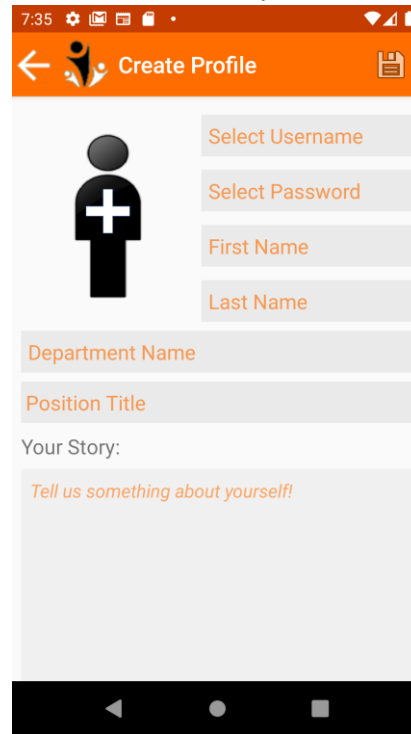
LOGIN

☒ Remember my credentials

Tap here to create a profile

Tap to clear saved API Key

Create Profile Activity



7:35

Create Profile

Select Username

Select Password

First Name

Last Name

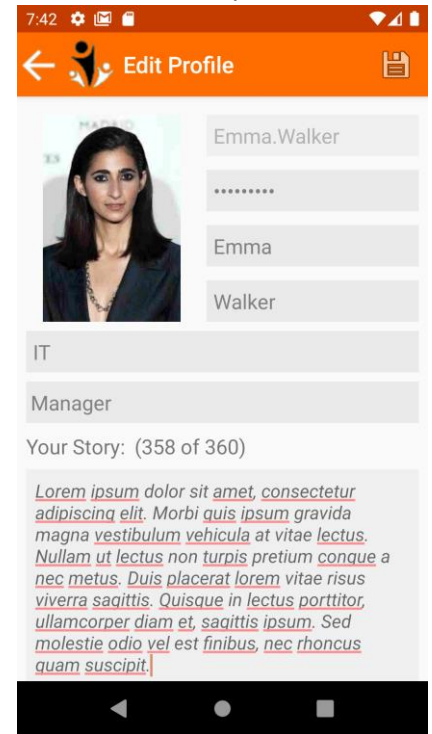
Department Name

Position Title

Your Story:

Tell us something about yourself!

Edit Profile Activity



7:42

Edit Profile

Emma.Walker

Emma

Walker

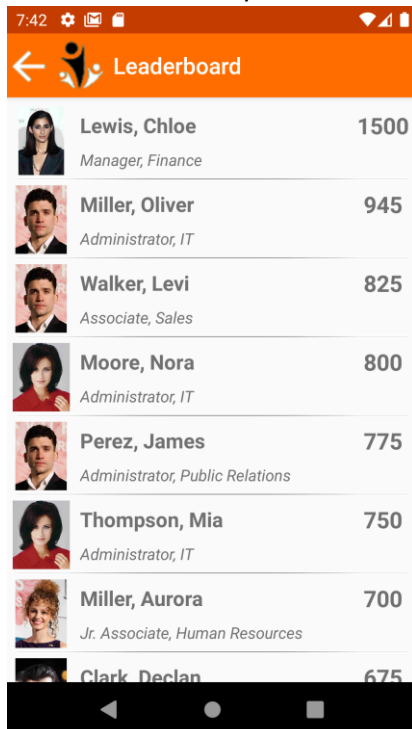
IT

Manager

Your Story: (358 of 360)









Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi quis ipsum gravida magna vestibulum vehicula at vitae lectus. Nullam ut lectus non turpis pretium congue a nec metus. Duis placerat lorem vitae risus viverra sagittis. Quisque in lectus porttitor, ullamcorper diam et, sagittis ipsum. Sed molestie odio vel est finibus, nec rhoncus quam suscipit.

Leaderboard Activity

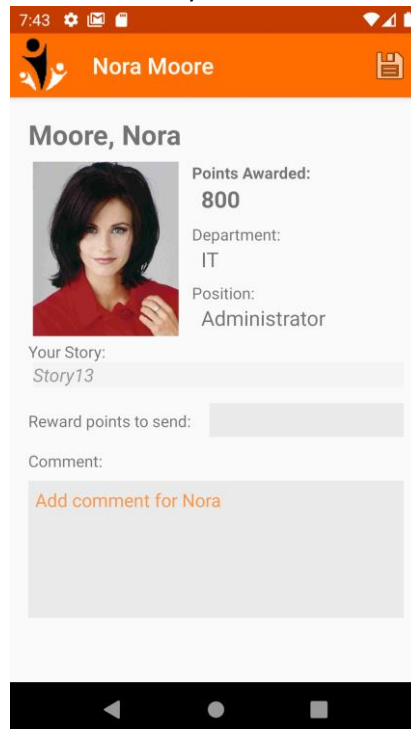


7:42

Leaderboard

	Lewis, Chloe Manager, Finance	1500
	Miller, Oliver Administrator, IT	945
	Walker, Levi Associate, Sales	825
	Moore, Nora Administrator, IT	800
	Perez, James Administrator, Public Relations	775
	Thompson, Mia Administrator, IT	750
	Miller, Aurora Jr. Associate, Human Resources	700
	Clark, Declan	675

Reward Activity



7:43

Nora Moore

Moore, Nora

Points Awarded: 800

Department: IT

Position: Administrator

Your Story: Story13

Reward points to send:

Comment:

Add comment for Nora



E) Rewards API

Data can be created, read, updated, or deleted via the Rewards API. The root endpoint for the Rewards API is:

Base URL = <http://christopherhield.org/api/>

Note: The Rewards API calls require you to first obtain an API Key. Students must provide their first name, last name, email (must end in ".edu"), and student id number. You MUST use your own university email and student id number!

The API: NOTE: Endpoints can be reviewed and tested via Swagger at: <http://christopherhield.org/index.html>

1) Request API Key

Endpoint: Base URL + `"/Profile/GetStudentApiKey"`

Connection:

Request Method: GET

Request Property: "Content-Type": "application/json; charset=UTF-8"

Request Property: "Accept", "application/json"

Query Parameters:

- firstName: Not null or empty
- lastName: Not null or empty
- studentId: Not null or empty, 20 character maximum
- email: Not null or empty, must contain "@", must end with ".edu".

Example Call:

<http://christopherhield.org/api/Profile/GetStudentApiKey?firstName=Chris&lastName=Hield&studentId=Chris.Hield&email=chield@domain.edu>

Example Response:

```
{
  "email": "chield@domain.edu",
  "apiKey": "ce01ae2cqrps799989"
}
```

Possible Errors:

HTTP_BAD_REQUEST (400) – If provided values are null, empty, exceed the maximum field size, or violate special requirements.

2) Create User Profile:

Endpoint: Base URL + `"/Profile/CreateProfile"`

Connection:

Request Method: POST

Request Property: "Content-Type": "application/json; charset=UTF-8"

Request Property: "Accept", "application/json"

Request Property: "ApiKey", "*Your API Key*"

Query Parameters:

- firstName: Not null or empty, 20 character maximum
- lastName: Not null or empty, 20 character maximum



- username: Not null or empty, 20 character maximum
- department: Not null or empty, 30 character maximum
- story: Not null or empty, 360 character maximum
 - position: Not null or empty, 20 character maximum
- password: Not null or empty, 40 character maximum
 - remainingPointsToAward: Positive integer value, 11 digit maximum
 - location: Not null or empty, 50 character maximum

Body Data:

Profile image (as Base64 String) – Not null or empty, 100000 character maximum

Example Call:

```
http://christopherhield.org/api/Profile/CreateProfile
?firstName=John&lastName=Smith&userName=jsmith123&department=Finance
&story=Tell other users about yourself...&position=Sr. Accountant
&password=MyAwesomePassword123&remainingPointsToAward=1000&location=Chicago, IL
```

Example Response:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "userName": "jsmith123",
  "department": "Finance",
  "story": "Tell other users about yourself...",
  "position": "Sr. Accountant",
  "password": "MyAwesomePassword123",
  "remainingPointsToAward": 1000,
  "location": "Mountain View, California", ← City, State from Location services
  "imageBytes": "/9j/4AAQSkA...QAAAQABAAD/", ← Shown truncated, it is very long
  "rewardRecordViews": [] ← A empty JSONArray (no rewards yet!)
}
```

Possible Errors:

HTTP_BAD_REQUEST (400) – If provided values are null, empty, exceed the maximum field size, or violate special requirements. Also results if the provided API Key does not exist.

HTTP_CONFLICT (409) – If a user already exists with the specified username.

3) User Login:

Endpoint: Base URL + "/Profile/Login"

Connection:

Request Method: GET
 Request Property: "Content-Type": "application/json; charset=UTF-8"
 Request Property: "Accept", "application/json"
 Request Property: "ApiKey", "*Your API Key*"

Query Parameters:

- username: Not null or empty, 20 character maximum
- password: Not null or empty, 40 character maximum

Example Call:

<http://christopherhield.org/api/Profile/Login?userName=Owen.Miller&password=OweMil123>

Example Response:

```
{
  "firstName": "Owen",
  "lastName": "Miller",
  "userName": "Owen.Miller",
  "department": "Finance",
  "story": "Story17",
  "position": "Manager",
  "password": "OweMil123",
  "remainingPointsToAward": 125,
  "location": "Austin, TX",
  "imageBytes": "/9j/4A QAAQABAQ...SkZJRgABA", ← Shown truncated, it is very long
  "rewardRecordViews": [{ ← A JSONArray of JSONObject
    "giverName": "Oliver Robinson",
    "amount": 175,
    "note": "Sed lacinia, urna non tincidunt mattis, tortor neque adipiscing di",
    "awardDate": "2020-12-21T18:33:14"
  },
  {
    "giverName": "Harper Gonzalez",
    "amount": 125,
    "note": "Curabitur sodales ligula in libero. Sed dignissim lacinia nunc.",
    "awardDate": "2020-12-21T18:33:16"
  }
]}
}
```

Possible Errors:

HTTP_BAD_REQUEST (400) – If provided values are null, empty, exceed the maximum field size, or violate special requirements. Also results if the provided API Key does not exist.

HTTP_UNAUTHORIZED (401) – If the provided credentials do not correctly match an existing user.

4) Update Profile:

Endpoint: Base URL + "/Profile/UpdateProfile"

Connection:

Request Method: PUT
 Request Property: "Content-Type": "application/json; charset=UTF-8"
 Request Property: "Accept", "application/json"
 Request Property: "ApiKey", "*Your API Key*"

Query Parameters:

- firstName: Not null or empty, 20 character maximum
- lastName: Not null or empty, 20 character maximum
- username: Not null or empty, 20 character maximum (*cannot be changed from current value*)
- department: Not null or empty, 30 character maximum
- story: Not null or empty, 360 character maximum
- position: Not null or empty, 20 character maximum



- password: Not null or empty, 40 character maximum
- location: Not null or empty, 50 character maximum

Body Data:

Profile image (as Base64 String) – Not null or empty, 100000 character maximum

Example Call:

```
http://christopherhield.org/api/Profile/UpdateProfile?firstName=Gabriel
&lastName=Anderson&userName=Gabriel.Anderson&department=Marketing
&story=This is a story&position=Director&password=GabAnd123&location=Boise, Idaho
```

Example Response:

```
{
  "firstName": "Gabriel",
  "lastName": "Anderson",
  "userName": "Gabriel.Anderson",
  "department": "Marketing",
  "story": "This is a story.",
  "position": "Sr. Accountant",
  "password": "GabAnd123",
  "remainingPointsToAward": 850,
  "location": "Boise, Idaho", ← City, State from Location services
  "imageBytes": "/9j/4AAQSkA...QAAAQABAAD/", ← Shown truncated, it is very long
  "rewardRecordViews": [{ ← A JSONArray of JSONObjects
    "giverName": "Freya Lewis",
    "amount": 200,
    "note": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.",
    "awardDate": "2020-12-21T18:33:19"
  }]
}
```

Possible Errors:

HTTP BAD REQUEST (400) – If provided values are null, empty, exceed the maximum field size, or violate special requirements. Also results if the provided API Key does not exist.

HTTP CONFLICT (409) – If a user does not exist, or the user attempts to change the username.

5) Get All Profiles:

Endpoint: Base URL + "/Profile/GetAllProfiles"

Connection:

Request Method: GET
 Request Property: "Content-Type": "application/json; charset=UTF-8"
 Request Property: "Accept", "application/json"
 Request Property: "ApiKey", "*Your API Key*"

Query Parameters:

- None

Example Call:

```
http://christopherhield.org/api/Profile/GetAllProfiles
```


Example Response:

```
[
  ⬅ A JSONArray of JSONObject
  {
    "firstName": "Grayson",
    "lastName": "Williams",
    "userName": "Grayson.Williams",
    "department": "Administration",
    "story": "Story20",
    "position": "Manager",
    "imageBytes": "/9j/4AAQSkZJRg...ABAQAAQ",
    "rewardRecordViews": [ ⬅ A JSONArray of JSONObject
      {
        "giverName": "Gabriel Lewis",
        "amount": 50,
        "note": "Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum.",
        "awardDate": "2020-12-21T18:33:12"
      },
      {
        "giverName": "Oliver Harris",
        "amount": 100,
        "note": "Integer nec odio. Praesent libero. Sed nisi.",
        "awardDate": "2020-12-21T18:33:14"
      }
    ]
  },
  {
    "firstName": "Freya",
    "lastName": "Sanchez",
    "userName": "Freya.Sanchez",
    "department": "Security",
    "story": "Story1",
    "position": "Jr. Associate",
    "imageBytes": "/9j/4AAQSkZJRg...kZJRgABAQA",
    "rewardRecordViews": [
      {
        "giverName": "Charlotte Thompson",
        "amount": 225,
        "note": "Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh.",
        "awardDate": "2020-12-21T18:33:17"
      }
    ]
  },
  {
    More users...
  },
  {
    More users...
  }
]
```

Possible Errors:

HTTP BAD REQUEST (400) – If the provided API Key does not exist.



6) Delete Profile

Endpoint: Base URL + `"/Profile/DeleteProfile"`

Connection:

Request Method: DELETE

Request Property: "Content-Type": "application/json; charset=UTF-8"

Request Property: "Accept", "application/json"

Request Property: "ApiKey", "*Your API Key*"

Query Parameters:

- userName: Not null or empty, 20 character maximum

Example Call:

`http://christopherhield.org/api/Profile/DeleteProfile?userName=Emma.Walker`

Example Response:

Non-JSON String: "User John.Smith Deleted"

7) Add Rewards:

Endpoint: Base URL + `"/Rewards/AddRewardRecord"`

Connection:

Request Method: POST

Request Property: "Content-Type": "application/json; charset=UTF-8"

Request Property: "Accept", "application/json"

Request Property: "ApiKey", "*Your API Key*"

Query Parameters:

- receiverUser: Not null or empty, 20 character maximum ← *The user who gets the award*
- giverUser: Not null or empty, 20 character maximum ← *The user who gives the award*
- giverName: Not null or empty, 40 character maximum ← *First & Last name of the giver*
- amount: Positive integer value, 11 digit maximum
- note: Not null or empty, 80 character maximum

Example Call:

`http://christopherhield.org/api/Rewards/AddRewardRecord
?receiverUser=Oliver.Miller&giverUser=Declan.Clark&giverName=Declan Clark
&amount=120¬e=This is a note`

Example Response:

```
{  
  "giverName": "Declan Clark",  
  "amount": 120,  
  "note": "This is a note...",  
  "awardDate": "2020-12-21T20:18:26.3530832-08:00"  
}
```

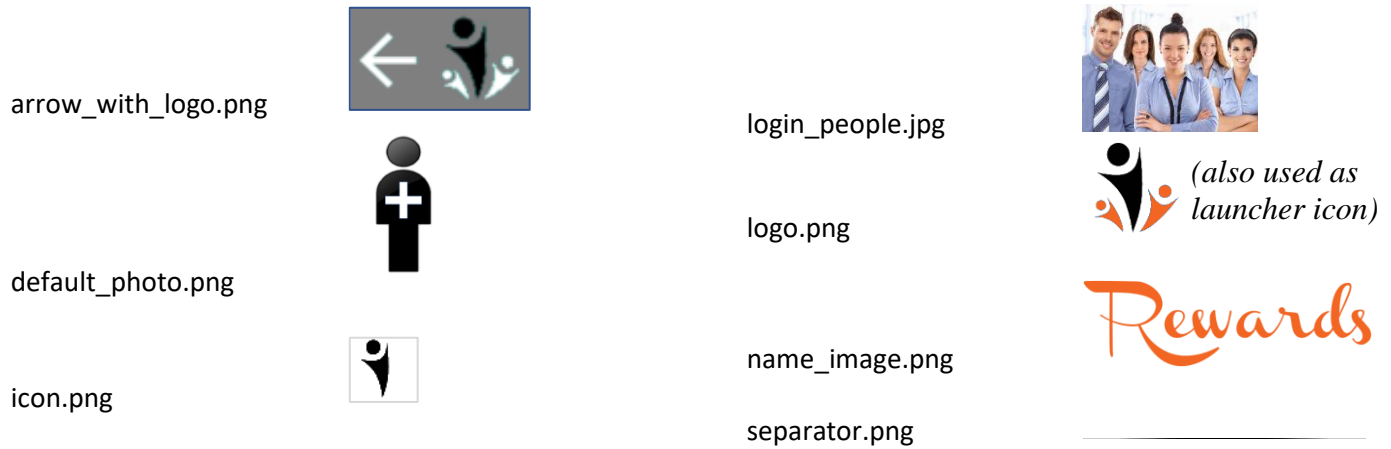
Note – there are 2 “HealthCheck” endpoints that return the (String) startup time of the Profile and Rewards controllers:

`http://christopherhield.org/api/Profile/HealthCheck`
Profile Controller Up since Sun, 27 Dec 2020 19:18:10 GMT

`http://christopherhield.org/api/Rewards/HealthCheck`
Rewards Controller Up since Sun, 27 Dec 2020 19:18:10 GMT

G. Provided Images

The following image assets are provided for you to use with this project:



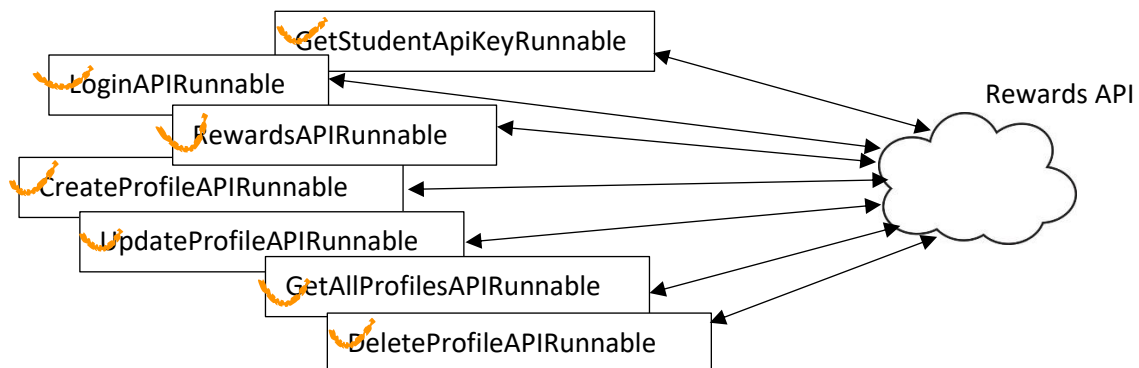
H. Menu Icons

For the required menu icons, you can use the build-in android menu icons:

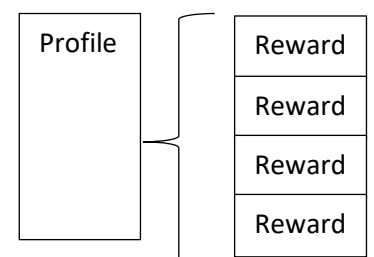
- ic_menu_edit
- ic_menu_save
- ic_menu_sort_by_size
- ic_menu_preferences
- ic_menu_delete

I. Other Classes

- You will need Runnables to execute the API calls described earlier and to receive their returned data.



- You should consider creating a class to represent the user profiles
- You should consider creating a class to represent the rewards a user receives.



Assignment Assistance

If you are stuck on an assignment problem that you have exhaustively researched and/or debugged yourself, you can put a ZIP file of your project on a share **location** (i.e., Google Drive, etc.) – make sure the link is public - and send me that link so that I can examine the problem. *All assistance requests must include a detailed description of the problem, and the details of what steps you have already taken in trying to determine the source of the problem.*

Note: To make your submission zip file smaller, before zipping your project file you must remove the “.gradle” (dot-gradle) folder (found in your project’s root directory), and remove the “build” folder (found in the “app” folder in your project’s root directory). NOTE: One or both of these folders may not be present – that is ok.

Submissions & Grading

- 1) Submissions must consist of your zipped project folder. For submission - before zipping your project file you *must* remove the “.gradle” (dot-gradle) folder (found in your project’s root directory), and remove the “build” folder (found in the “app” folder in your project’s root directory). NOTE: One or both of these folders may not be present – that is ok. Submissions not following these requirements will be penalized.
- 2) Submissions must reflect the concepts and practices we cover in class, and the requirements specified in this document.
- 3) Please review the Academic Integrity and Plagiarism section of the syllabus before, during, and after working on this assignment.
- 4) Late submissions will be penalized by 10% per week late. (i.e., from one second late to 1 week late: 10% penalty, from one week late to 2 weeks late: 20% penalty, etc.). NO SUBMISSIONS CAN BE MADE BEYOND 2 WEEKS LATE.
- 5) Grading will be based upon the presence and proper functionality of all features and behaviors described in this document.
- 6) Grading will be performed with the following SDK details:
 - Project Compile & Target SDK Version: 30
 - Project Minimum SDK Version: 25
 - Use AndroidX libraries where applicable.
- 7) Grading will be performed on emulator devices with the following characteristics:

Resolution	Details	Example Emulators
1080 x 1920	With Playstore	Pixel, Pixel 2, Nexus 5, Nexus 5X
1080 x 2220 or 2280	With Playstore	Pixel 3a, Pixel 4

If you do not understand anything in this handout, please ask.

Otherwise the assumption is that you understand the content.

Unsure? Ask!