The initial idea and high-level specification of “Shop with points”

Proposed by:

Alireza Samadifardheris 1961823
Ashkan Ansarifard 1970082

- **General Description of the suggested application:**

The application we would like to implement is a mobile application called **shop with points** and the idea is that the more users contribute to the environment of the application to review businesses (restaurants, quality of cloths, behaviour of hotel staff or any other business), the more they will get points. Later they can use their collected points to get benefits from discounts from any other business registered in the application.

In this application, there are two main types of users:

1. The first type of users are owners of businesses. The owner of the business requests to create a business on the application, saying a restaurant, and then the email address of that user would be registered as the business owner's email address.

2. The second type of users are normal users: these users have the capability of rating businesses (for instance: restaurants) and posting reviews for them. The more they contribute by giving feedback and reviews and improving the community, the more they will have points in their wallet. These points can be used, to benefit from some discounts from any shop. (one point per review)

   Note that, it is the business owner who decides how much discount he would like to give to a specific amount of points by putting this information on their page when they request to create an account. Some examples could be:
   - ✓ 10% discount on pizza by 10 coins redeemed from your wallet
   - ✓ 1 Cola if you spend at least 10 euros by redeeming 7 credits from your wallet
   - ✓ Buy jeans and get up to 60% discount (1 credit = 1%)

   You may ask why business owners should care about some virtual credits in their wallet? Though this part is not intended to be implemented yet , probably they can use sum of credits redeemed from their customers to get some benefits provided by the app (payment from an income source of the app to owners, advertising their business through this app, or any other app, etc)

- **Authentication:**

The users' log-in and account creation will be securely authenticated by authentication-providing services (such as Auth0) that provide user management like associating roles to users. It is only based on login information, where the user is going to be directed (either owner page or normal user page)

- **The frontend:**

1. **Business owner users:** These users have the ability to redeem from normal users' credits upon credit owners' request. Owner of business is connected to normal user's wallet by scanning with the camera a QR code generated on normal users' shopwithPoints Mobile application, then they can redeem the credits from their wallet.

2. **Normal users:** logging in as a normal user will allow users to see businesses nearby on the map by pins as well as the list of businesses. Users should be able to go to the businesses' page, see the name, description, how much discount would they earn by a specified amount of credits as well as comments by other users and the average score of the restaurant. The users are also allowed to write review and score the restaurant (maximum one review per user per business is the easiest way to avoid spam and we aim to have it in this way). The should have ability to go to a section , say wallet, to check how many credits they gathered so far and also generate a QR code valid only for a short period to be scanned by a business owner. (The idea of using QR code is to exchange the information of a JWT which could contain information about account information and the number of points user has.

- **The Backend**

In each view, there will be handled some CRUD operations by the use of REST APIs for interacting with a database application. the information is retrieved by API calls asynchronously to avoid UI becoming unresponsive meanwhile since API calls are usually considered slow procedures. when calling an API, a completion handler must be specified which is a defined function and will be executed after data are retrieved and a new thread will be created to wait for the result so the performance of the application which is running on the main thread won't be affected. Some obvious examples of these API calls that they are highly probable to be needed and implemented are:

asynchronous call of the API retrieves (GET)the information for each element on the floating panel where user can see the lists of the businesses

asynchronous call of the API retrieves(GET) the information for each pin on the map where user can see the location of the businesses

asynchronous call of the API retrieves (GET)the information of businessID for the tapped business including image, description, average score, as well as comments, the username of the commenter, and the score to be shown on the new view

asynchronous call of the API posts (POST) the comment and score of user with UserID for a business with businessID. when submit review is tapped if the text is null, then the application must request for the text of the review, else a review will be created containing user-id, review text, review score, the user id is saved in a data holder shared all among the views and the app. Then when the review is created the API call should post this and return ok if successful, else it will return an error. After submitting the review, our review should be seen in the reviews, and the average score is should be also updated. This can be also done for any other business

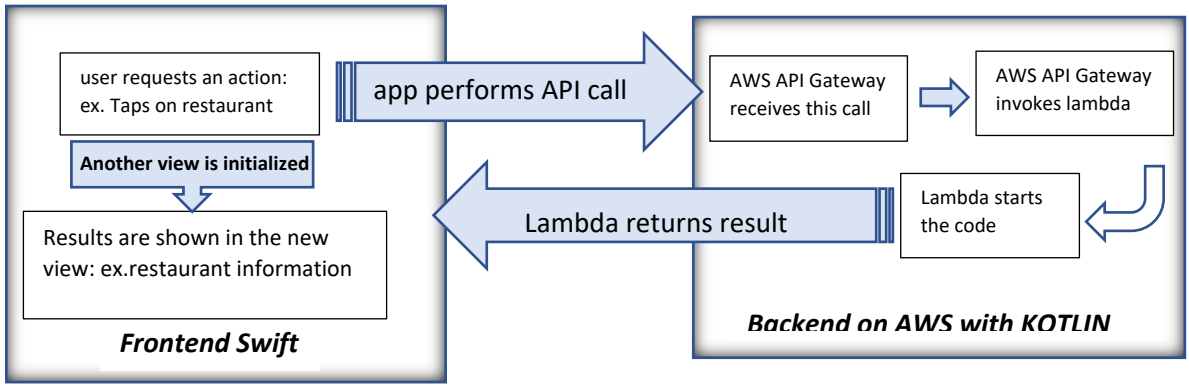
asynchronous call of the API deletes (DELETE) the comment and score of user with UserID saved in a data holder on the restaurant with businessID

asynchronous call of the API retrieves (GET) wallet credits of the user saved in the database to be shown in the wallet section

asynchronous call of the API retrieves (GET) wallet credits of the user saved in the database to be shown in redeem page of business owner

asynchronous call of the API updates (UPDATE) wallet credits of the user saved in the database decreased proportionally as much as requested. business owner from BusinessOwnerView can simply open the camera through the app, point the camera to the QR code generated by the user, if the QR code is valid, it will be directed to another view saying partner redeem view controller that asks how many points of the user want to be used, when the points redeems successfully it will be sent through API call to decrease from user points and return a successful redeemed done message to partner. On the user side also, the points should proportionally be decreased

etc.



The backend could be deployed on AWS lambda because it is powerful and free for 1 million invocations providing it the jar of backend code and that the code would be exposed to API calls. This approach is a serverless approach meaning that my code is not running on a specific server, but when it is exposed to a call, lambda will start our code, give the response to the user, and stop it.

Obviously a database service (such as MongoDB) for storing data including comments, reviews, user ID, the image of the restaurant, location, credits of the user, etc.) is fundamental and we may need to retrieve filtered information over the data stored in the database.