CPEN 391 - Module 2 Individual Report

a) Individual Identification: Be sure to give your name, student number, group number, and group project title.

Name: Qian Rui Chow
Student number: 46351152
Group number: 11
Group project title: Quickshop

b) Individual Contribution: List all of the tasks you worked on personally. For each task, indicate whether you performed the task yourself or in conjunction with someone else. If any of these tasks required an especially large amount of effort, indicate this. If you believe any of the parts you did are of exceptionally high quality, provide details to justify this. This section should be as detailed as possible – vague or overly generous statements that are considered evasive, confusing, or overstating your work will negatively affect your mark.

Tasks I have worked on personally:

I worked on the UI design of user mobile application. I chosen a background colour for the app which is not only soothing to the eyes but also matches our shop logo. I ensured that important and often used buttons are easily located at convenient places. Below are the list of features that I have designed and implemented:

a) Registration

User can upload their profile picture from using front camera, back camera or from their own photo library. User can view their chosen profile picture at the registration page before uploading. The display of their chosen profile picture is of circular frame, prompting user to choose a picture with only 1 face in it. Besides that, user can verify if their picture met the requirement by clicking on a valid button. An alert message will pop out to notify user if their picture is valid or if the picture is not valid because it has x number of faces. Registration is only successful if new users submit a unique username, if their password matches the repeat password field and if their picture has only 1 face in it. If any of these criteria is not fulfilled, alert message will pop out telling users the exact reason and prompts them to register again. Any fields that are left empty will also generate an alert message to prompt user to register again.

b) Security - Login

In the login page, user entered username and password will be authenticated and authorized by the server. I implemented a function that sends a POST call to the server for this to happen. A unique token will be received if login is successful and user can access the features of the app.

c) Main menu

Main menu has 3 obvious icon buttons, which users can identify easily by just looking at the icon: My Cart, Transaction History, My Profile. User can also choose to Log Out if they want to exit the app.

d) My Cart

My Cart displays the current items that the customer intend to purchase in quickshop. With this display, customer can check if they are correctly charged before exiting the quickshop. It has a user friendly display as it shows a picture of the item in their cart next to the description of item, quantity and price. I implemented a function that loads the JSON from URL and displays it accordingly on the display. Besides that, the total price of all items the customer plan to purchase can be found at the right bottom of the display.

e) Transaction History

Transaction History displays a webview of the blockchain data of transaction ledger. A user can view their past transactions in store here.

f) My Profile

This display allows user to double confirm that the account that they are signed in is truly their own. It also allows user to check their current profile picture and username. Normal procedure, which is to download the user unique profile picture through URL takes a very long time. I have further improved the latency/performance issue by using a caching library.

c) Detailed Design: Describe your part of the project, including (where relevant): - Block Diagram of the hardware part of your embedded system - Architecture of your software: Modules, submodules, libraries, components etc. - List of functionality and features (from the user's perspective) that you implemented. Describe the level of completeness of each. - User interface/GUI description e.g. screen shots - Major data structures and where they are stored. - Functionality you implemented in hardware inside the Graphics accelerator, e.g. draw horizontal line, vertical line, line from [x1,y1] to [x2,y2], plus anything you did above and beyond this e.g. rectangles, triangles, text + fonts etc. - What have you included to make sure your design could be extended in the future? (all companies want to allow for subsequent versions and upgrades) - Any other Technical details regarding any interesting features you implemented.

I ensured that my code for each feature of the user app is in submodules. Here is a list of the submodules

a) Registration

Figure A is a screenshot of the registration page. For this page to be working, I used Vision for face detection, Alamofire for HTTP calls, MobileCoreServices and QuartzCore frameworks for Camera/Photo Library permission. Keyboard appears and display scrolls up when user desires to input text into text fields. This is implemented so that user can view their input while typing. I also implemented placeholder text in the text field so that the screen is not clumped up by

redundant labels. Because Alamofire does its network calls asynchronously, I created a registerSuccessCallback function such that it will then allow an alert to pop up to notify user that registration is successful or not. And if not, what is the reason of failed registration in order for them to make changes and register again. When using the camera, users can retake immediately if their recent snapshot was not satisfactory. The “I already have an account. Let me login.” is a button that directs users to log in if they have an existing account or if they want to login immediately after successful registration.

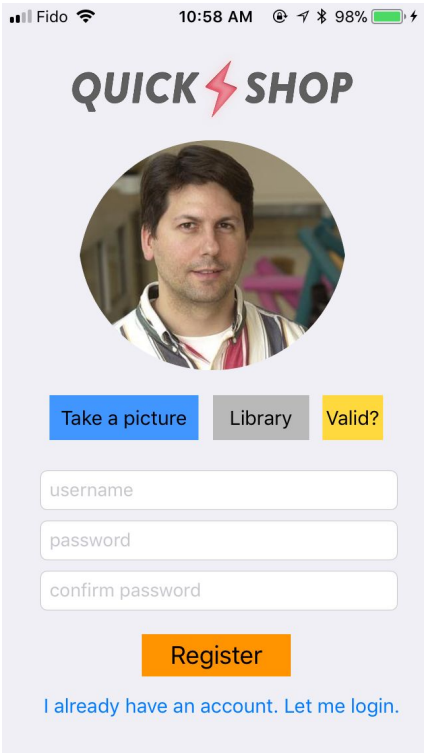

Figure A

b) Face Detection

Front camera has a problem detecting faces, while back camera and photos from photo library has no problem detecting faces. As many users tend to prefer selfies, I investigated into the problem and found out that the pictures from the front camera does not rotate images – it just sets an orientation EXIF-tag. Hence, I implemented a normalizeImageRotation function that hard rotates the photo before uploading it for face detection processing.

c) Camera Permission

It is a code written that detects the permission of the camera and handles all the cases properly. This allows user to access the camera immediately when button on registration page is tapped.

d) Security - Login

The app communicates with the server for login authentication and authorization in this submodule. It was not easy to use Alamofire as it is not a built in library. I needed to implement

a loginSuccessCallback function so that when server sends back a token for successful login, the user will be directed to the main menu page. If login failed, an alert will pop out showing the error occurred. Login page is shown in Figure B below.

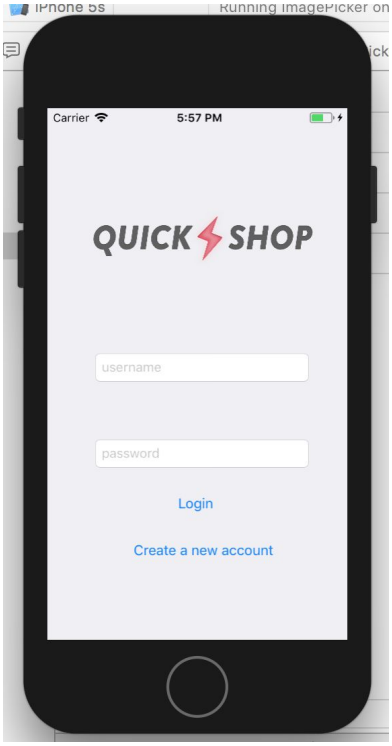

Figure B

e) Main menu

This display has three picture icons that leads to the three main functionalities for the user, as shown in Figure C. User can also log out in this view controller.

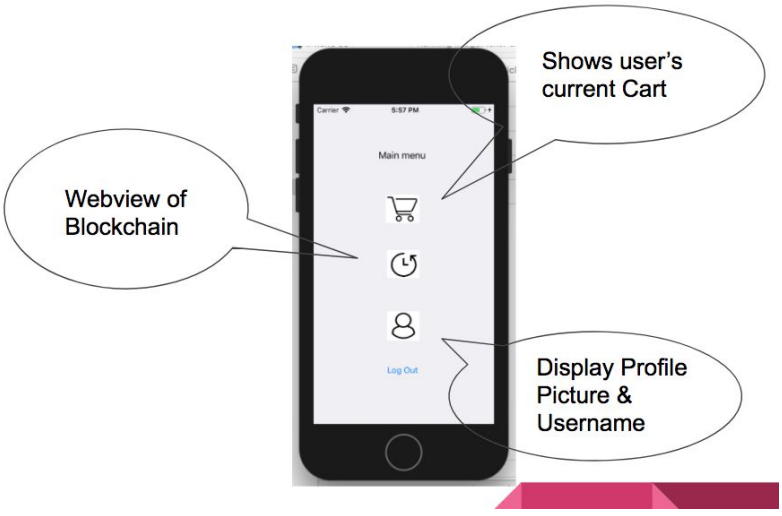

Figure C

f) My cart (CartViewController)

The server sends the app a JSON string, consisting of product name, quantity and price. I implemented a getJSONfromUrl function, parses the JSON into string/int array and displays it according to the columns in the display, as shown in Figure D. The display of product picture is done in the app, implemented by me.

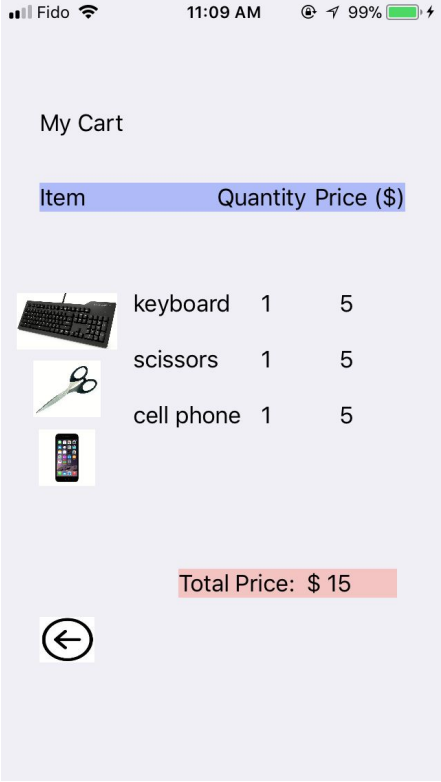

Figure D

g) AlamofireHTTPCalls

Alamofire is a HTTP networking library written in Swift. It supports HTTP response validation, Upload File / Data / Stream / MultipartFormData and does chainable requests. This submodule contains functions that I have implemented, makePostCallWithAlamofire and makeGetCallWithAlamofire with my own unique function parameters which include successCallback. By writing these functions in this file, I can code reuse in submodules that needs these functions.

h) Transaction History (BlockchainViewController)

I have implemented a webview that displays the website of blockchain transaction history in a portrait orientation. It has a back button that allows user to go back to the main menu. A screenshot of this display is shown in Figure E.

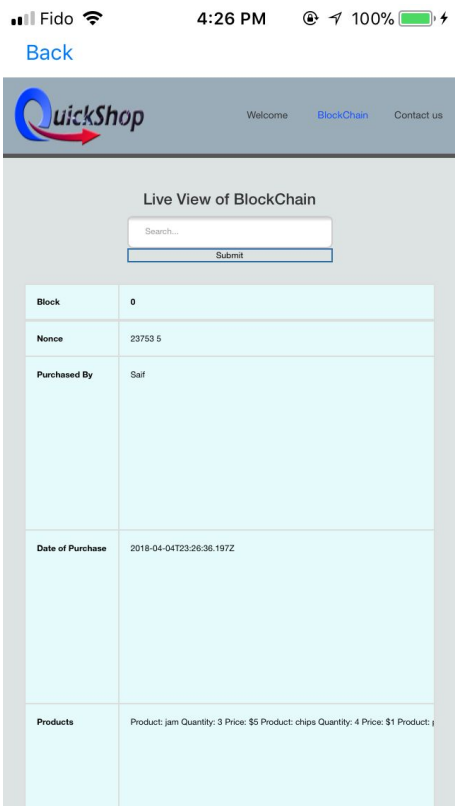

Figure E

i) My profile (MyAccountViewController)

When the user logged in, the username of the user is stored in the app. Hence, to display the username was straightforward, calling UserDefaults.standard. The other part was to display the user's profile pic. A getNameUrl function was implemented to get the unique url, for example, if username was Guy, then the URL would be http://store.saif.ms/users/Guy/picture. getProfilePicFromUrl auto loads when view is displayed. One latency/performance problem occurred here. The image downloads successfully from the URL, but it takes very long to display. Hence, I investigated the issue and the root cause was because of the asynchronous call. This issue was solved when I used an outsourced caching library, SDWebImage, to load the image and display it on the view controller.

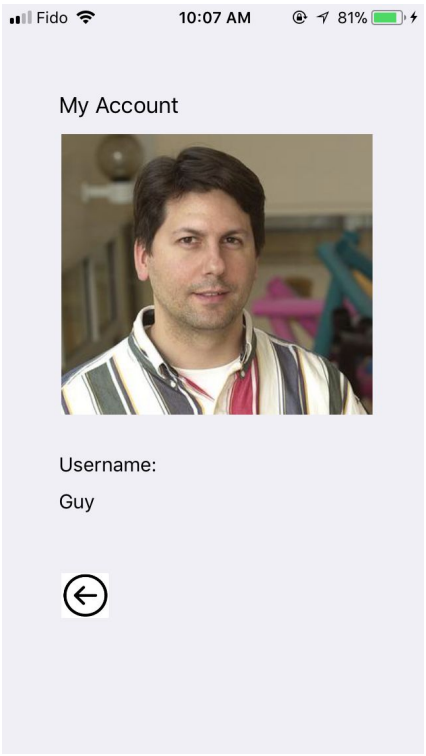

Figure F

d) Challenges: Explain the major challenges you faced in completing your project. Explain any interesting solutions to these challenges: did you use interesting algorithms, or techniques? Did you develop

The few challenges I have faced is listed below:

- Only Front Camera unable to detect faces
    - Image needs to be rotated straight before faceDetection processing. Was hard to debug as no error was shown and back camera, photo library pictures work on face detection processing.
    - Solved by hard rotating the front camera images for faceDetection processing
- Library that supports live Camera which detects faces only support >iPhone6
    - Solved by using webcam instead of phone camera
- UIImage from URL takes long time to load in Swift
    - Due to asynchronous call
    - Hard to detect error as picture does download successfully
    - Solved by using caching library SDWebImage
- Unable to upload picture in Register using Alamofire to multer in server
    - No error shown but picture was not uploaded
    - Solved by specifying the fileName & mimeType using Alamofire in order for multer to pick up the upload on the server end
- iPhone camera and photo library prompts for user's permission to access

- Need to write code that detects the permission prompt and handle the cases

e) Team Effectiveness: Evaluate the effectiveness of your team, and give suggestions for improvement

Our team worked very well. We were all committed to the success of the project. My groupmate, Justin suggested the webcam idea when using the phone camera for entrance and exit camera could not be implemented by me as I need an iPhone 6 and above. We worked together during weekends. Everyone replies promptly on group chat. Everyone was also clear of their own tasks and we set realistic goals at all times. We had many internal deadlines of what we want to achieve every week. We reserved the last week for enhancing the project, handling unforeseen cases, and presentation flow. The project did not feel rushed towards the end because of good planning and time management.

f) Other Comments: Include comments on items such as: - How did you ensure that your tasks would integrate with your group? - What hurdles did you have to overcome? - What did you do to ensure success, or at least improve the likelihood of success? - What did you learn? - Were there any team dynamic issues? - Anything else you spent your time on (related to the project :-)

I have no prior experience in mobile application development and front end development. So, everything was new to me. I have also never communicated with server. I went to the wrong direction for one week by researching how to implement a local SQLite database and implemented the foundation of it. Later, I realized that all I needed to do was to send GET and POST calls to the server implemented by my group member. The camera permission issue took me some time as well because I did not know what was the root cause. In order for the app to work, external libraries were needed as well. I wanted to enhance the UI for My Cart by making a table view, but was unable due to the time constraint. There was a lot of things I wanted to implement for the app, for example, user being able to update their profile pic from the app, add credit card details/payment processing, and enabling user to delete their account from the app.

g) Acknowledgments: Did you include any source code / IP / etc from other sources? Note that something done by yourself, but for a different course, is considered an "other source" UBC does not allow you to get academic credit multiple times for the same work.

Alamofire:
https://github.com/Alamofire/Alamofire

Face detection:
https://medium.com/@dragosholban/face-detection-with-apples-ios-11-vision-framework-a143a15e384d