BeachTrade

Final Report

Tyren Villanueva, William Luong

This report will cover a wide range of topics and detail various implementations that were used to develop a fully functional Android Application. The application, BeachTrade, was designed as a platform for Cal State Long Beach students to connect and trade their items. Students are no longer limited to unorganized Facebook groups, and can respond to sellers at a click of a button.

1. Introduction and Background

1.1 Statement of Problem Area (brief, nontechnical)

Textbook prices at the Cal State Long Beach Bookstore are too high and unreasonable for students. Students are always looking for new ways to save money and get their textbooks at a cheaper rate. Our application, BeachTrade will allow students to find textbook at better prices compared to the Bookstore. Not only will a financially struggling students have the ability to save money, they can earn money by selling their old items.

1.2 Previous and Current Work, Methods and Procedures (representative)

We previously constructed a social application that allowed users to stay fit and challenge friends. This previous work inspired BeachTrade, because we wanted to take our experience with social networking and help our fellow students.

1.3 Background

BeachTrade is an application designed to solve the problems we faced as students. The current method to trade and sell books at CSULB is inefficient and time wasting. Selling a $100 dollar book back to the bookstore can earn you a measly amount of $5 dollars back. Facebook groups are currently the most popular method for selling student items, but it is a difficult tasks finding the exact item you need. There are no categories or filters available to limit your search. The Facebook group is also populated by people who don't go to the university or have an interest in selling items. The CSULB book trade group feed is usually populated with pictures or posts that have no connection to what the group was designed for. These existing issues led to the creation of BeachTrade.

1.4 Brief Project Description (overview of new, extended or different functions, structure or operation)

Our application is designed to target the Cal State Long Beach audience by offering them a quick and easy place to sell their items online.

1.5 Purpose/Objectives/justification of Project (theoretical, practical, or educational impacts on users)

The purpose of BeachTrade is to allow less fortunate students to get their textbooks at a cheaper price and earn money by selling books. Most college students are in debt or getting by with a minimum way job, thus our app is a solution for students who cannot afford to get their required textbooks to be successful in their class.

1.6 Team work assignment and accomplished

As a team, we've accomplished most of our project goals. We each assigned our own goals with both AWS implementations and actual source code work. We worked closely together to make sure we were both on track and completed tasks to each others liking.

2. System Functional Specification

2.1 Functions Performed (itemize and describe)

SubmitUserItem()- This function takes the Cognito identity of the user and posts a product description to a user feed.

Login.registercallback()- this handles authorization and with a facebook token and ties it with AWS Cognito user identity.

User profile.getDetails()- This retrieves the information of the user profile and is used with the User profile activity to display a user's information

2.2 User Interface Design

Dynamic user interface with recyclerview and cardview. This is used to display user posts and the items that are up for sale.

2.2 User Input Preview

User can traverse through application by using the included Navigation Bar

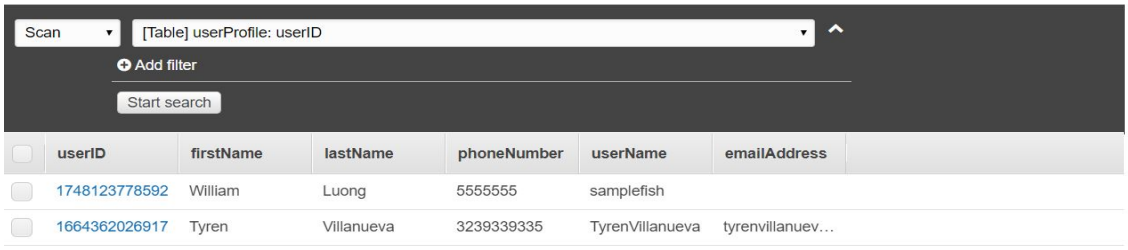
2.3 User Output Preview

-Successful registration through Facebook.

-User profile changes displayed for other users.

2.4 System Data Base/File Structure Preview

DynamDB- Users and user items



userID	firstName	lastName	phoneNumber	userName	emailAddress
1748123778592	William	Luong	5555555	samplefish	
1664362026917	Tyren	Villanueva	3239339335	TyrenVillanueva	tyrenvillanuev…

itemID	description	itemName	itemType	pictureLink	price	userID	username
1748123778592	for eng majors	grammar han…	Books	chicken.jpg	10.01	17481237785…	
1748123778592	good	potato	Other	chicken.jpg	1.26	17481237785…	samplefish
1748123778592	used iclicker	iClicker	Technology	chicken.jpg	14.11	17481237785…	
1664362026	used	toaster	Technology	chicken.jpg	10.99	16643620269…	
1664362026917	good condition	iphone7	Technology	chicken.jpg	300	16643620269…	
1748123778592	For Bio101 cl…	An Intro to Bi…	Books	chicken.jpg	99	17481237785…	

2.5 External and Internal Limitations and Restrictions

Limitations are caused by the capabilities of the mobile device being used. When the application was used on my Samsung Galaxy S7, the user profile information loaded quickly. When used on an older Galaxy S4 the profile page took a while to retrieve information from the database and had trouble rendering the profile picture.

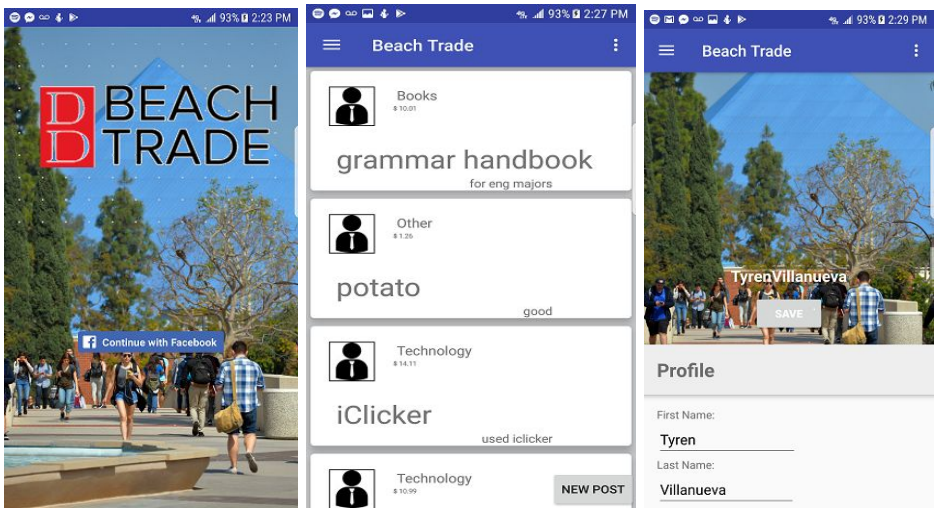
2.6 User Interface Specification

2.6.1 Interface Metaphor Model

-Standard implementation of user login and password entries.
-Standard implementation of user specific data entries (profiles).
-Standard implementation of user profile Photo upload; take a new photo or upload a photo from user's library.
-Navigation bar implementation for easy traversal through network activities

2.6.2 User Screens/Dialog

Home Screen Post Feed Profile Page



2.6.3 Report Formats/Sample Data

-Registered users/Facebook users

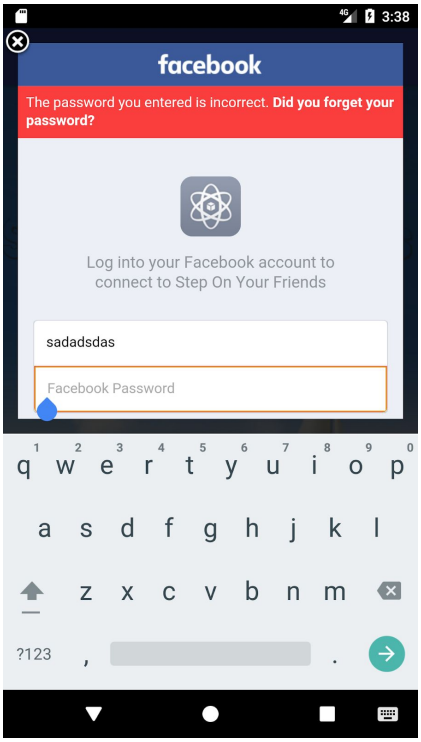
-User Posts

2.6.4 Online Help Material

Online help material will be posted to https://s3.amazonaws.com/aws-website-csulbtrade-x5cdc/index.html

2.6.5 Error Conditions and System Messages

-Invalid email/password combination - Facebook API.



2.6.6 Control Functions

-onCreate(Bundle)
-onSuccess(LoginResult)
-onCancel()

-onError(Exception)

3. System Performance Requirements

3.1 Efficiency- this application is efficient by organizing user posts and providing all the needed tools and functions in a navigation bar.

3.2 Reliability

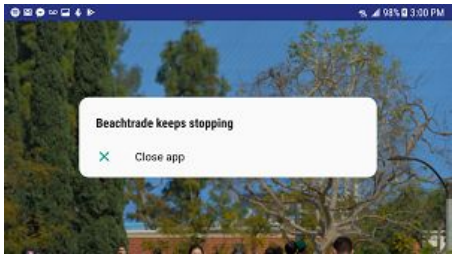
-The application is dependent on an existing internet connection. The application will ensure that any missed notifications will be retrieved once the user is connected back to the internet. Messaging takes place over email, therefore a user cannot miss a sent message.

3.2.1 Description of Reliability Measures (accuracy, precision, consistency, reproducibility, etc.) The application is guaranteed to perform exactly what it set out to do. In order to increase reliability and consistency, we set out to use third party tools like aws services, and the facebook API. The choice to handle communication through email removes the issues that comes with the inclusion of a chat service.

3.2.2 Error/Failure Detection and Recovery (failure modes, failure consequences, error logging and reporting, manual and automatic recovery procedures)

-Failure consequences - application data may be null/corrupted or not saved, restart required
-Error logging - IDE detects and reports warnings
-Recovery- When application is recovered, the state of the application is restarted to the login page. All of the user information is intact due to the pulling from our database.

3.2.3 Allowable/Acceptable Error/Failure Rate



This happens when a device is constantly rotated.

3.3 Security

3.3.1 Hardware Security

3.3.2 Software Security

If placed on the market, our device will be available on the google play store. Applications on the play store are verified and proven to be secure for commercial use.

3.3.3 Data Security

The data sent and retrieved from this application is handled by AWS DynamoDB. Unless the administrator alters the tables, all of the user's data and post is secure by Amazon cloud services.

3.3.4 Execution Security (user validation)

User validation and authentication uses dependencies provided by the Facebook API. The token that is provided to the user is represented by a unique identifier that is hard to replicate by adversaries. A man in the middle or dictionary attack is halted thanks to Facebook's protocols.

3.4 Maintainability

Control over issues and errors occurs through the synchronization of github. An error found by a team member can be quickly repaired with a push.

3.5 Modifiability

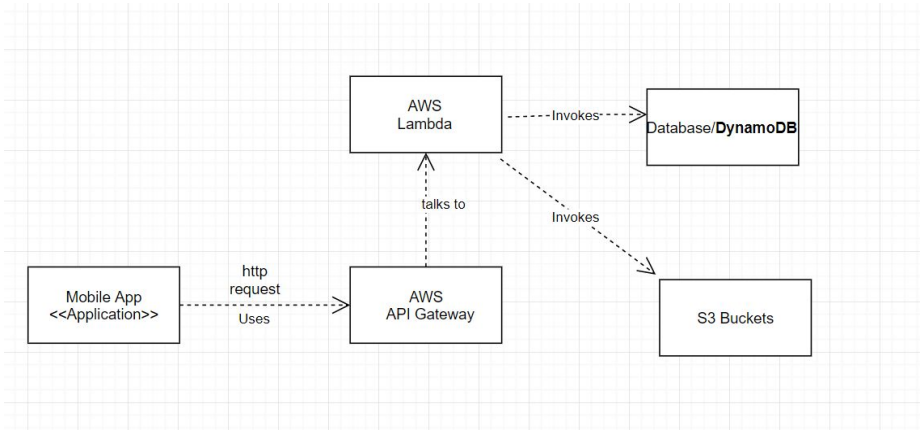
BeachTrade was developed for Android applications only. A shift to IOS would require a rework over the entire source code.

3.6 Portability

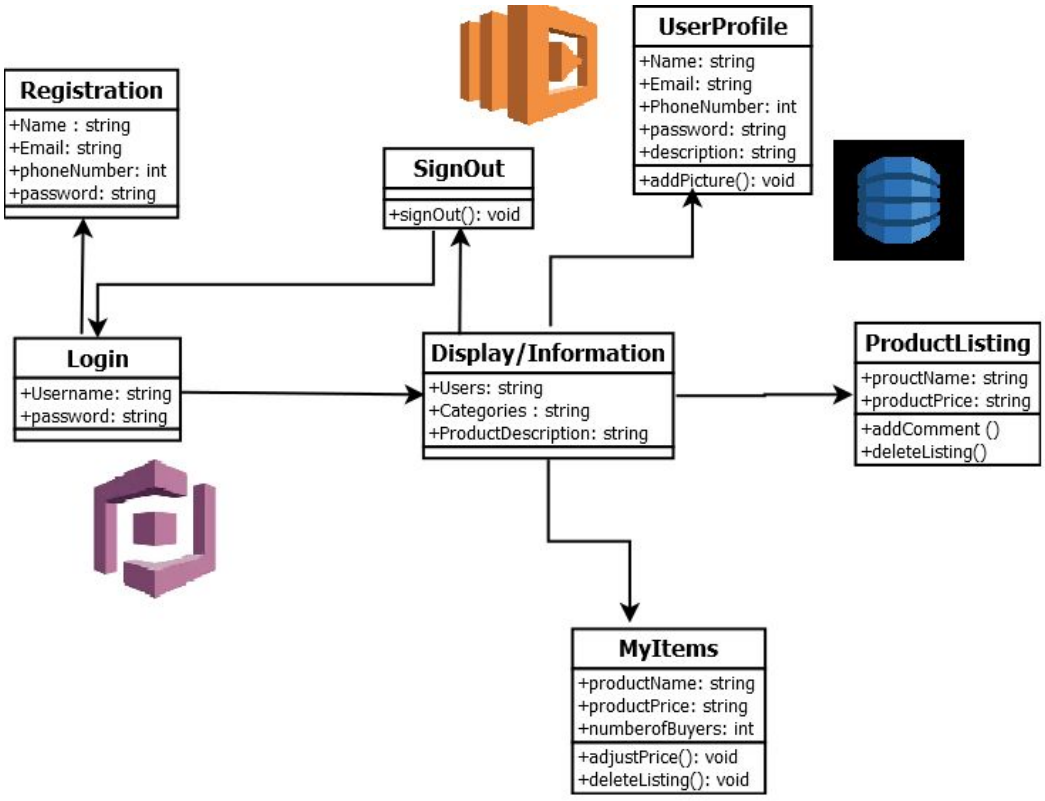
Portability is dependent on device's operating system. Earlier versions of Android may not be compatible.

4. System Design Overview

4.1 System Data Flow Diagrams



4.2 System Structure Charts



4.3 System Data Dictionary

Name- Registered user full name

Email- Registered user email addresss

productName- The price of the product that is listed for sale

adjustPrice()- function that allows user to change the price of

deleteListing()- delete the product on the social media feed that belongs to that user

4.4 System Internal Data Structure Preview

AWS DynamoDB

firstName-User entered first name

lastName- User entered last name

phoneNumber- User phone number

emailAddress- User email address

4.5 Description of System Operation (high level)

Combines social media share and user profiling to allow users to connect and communicate what they desire from other users

4.6 Equipment Configuration (diagram and description)

Android emulation- Samsung Galaxy S7

4.7 Implementation Languages (which and why)

Java was the chosen implementation language for portability and its compatibility with Android Studio.

4.8 Required Support Software (preexisting)

Software includes Android Studio IDE, Java JDK and SDK.

5. System Data Structure Specifications

5.1 Other User Input Specification

5.1.1 Identification of Input Data

-Data is identified through input and output in Java

5.1.2 Source of Input Data (NOT input device)

Input comes from the user, Facebook API, and AWS Cloud services

5.1.3 Input Medium and/or Device

Any device running the operating system Android 7.0 Nougat can be an input medium for the application.

5.1.4 Data Format/Syntax

Data is limited to Java Syntax

5.1.5 Legal Value Specification

-Validation is offered on the Registration and Login page due to the policies required by Facebook

5.1.6 Examples

A user who decides to input a password with less than eight characters will be prompted to retry.

5.2 Other User Output Specification

5.2.1 Identification of Output Data

-Data is identified through input and output in Java

5.2.2 Destination of Output Data (NOT output device)

User profile data is stored on both AWS Cognito services and Facebook servers. The user's entered credentials and their posted items will be stored on Amazon DynamoDB.

5.2.3 Output Medium and/or Device

The output device is any mobile device running Android 7.0 Nougat or higher.

5.2.4 Output Format/Syntax

Output is standard english formatting and Java syntax

5.2.5 Output Interpretation (meaning of output)

Output describes user's information and the product they are trying to sell

5.2.6 Examples

When a user inputs the information for their product, a card is created and placed in the card view that displays all the information of the the product to potential buyers.

5.3 System Data Base/File Structure Specification

5.3.1 Identification of Data Base/Files

Database information is identified through Amazon Cognito identities.

5.3.2 (Sub)systems Accessing the Data Base (creating, updating, using; frequency)

The implemented AWS SDK libraries and functions help carryout calls to DynamoDB

5.3.3 Logical File Structure (record formats, file organization, access methods, rationale, examples)

Files and records are organized and kept in Amazon S3 buckets.

5.3.4 Physical File Structure (storage device, blocking, organization, access, etc.)

5.3.5 Data Base Management Subsystems Used (internal or external)

AmazonDynamoDB was used for database management

5.3.6 DataBase Creation and Update Procedure (if NOT by system)

A function call embedded in the AWS SDK is used for updating.

5.4 System Internal Data Structure Specification

5.4.1 Identification of Data Structures

NoSQL database service

5.4.2 Modules Accessing Structures (creating, updating, using)

5.4.3 Logical Structure of Data (format, organization, access, rationale, examples)

Format is a list of strings and organized in database tables

6. **Module Design specifications (for each module)**

6.1 Module Functional specification

6.1.1 Functions Performed

Creating and updating a card to hold product information that can be displayed on the server

6.1.2 Module Interface Specifications (input/output arguments/global variables/files)

Input is a list of strings that hold the information for user items. Output is a card displayed on the main page. Variables for module are specified in section 4.4.

6.1.3 Module Limitations and Restrictions

This module is limited to text and cannot accept other objects that we intended.

6.2 Module operational Specification

6.2.1 Locally Declared Data Specifications (variable dictionary)

`AccessToken accessToken` -This requires the token received from Amazon Cognito

`CognitoCachingCredentialsProvider-`This offers the ability to persist the Cognito Identity in SharedPrefferences

`itemName-`Name of the product that is being sent to card.

6.2.2 Algorithm Specification (flowchart, pseudocode, decision table, etc)

6.2.3 Description of Module Operation

-Uses AWS sdk to generate feed with user items

7. System Verification

7.1 Items/Functions to be Tested

User Login Tested

-Valid/Invalid input

User Registration Tested

-Valid/Invalid input

Profile Activity Tested

-Valid/Invalid input

7.2 Description of Test Cases

User Login is tested ,but returns an error due to the automated services not having the ability to log into a facebook account.

7.3 Justification of Test Cases

Login must be tested because the entire application depends on it. Without a verified Cognito token, the application will not work.

7.4 Test Run Procedures and Results

AWS Device Farm was used for automated testing. When both login and the profile activity was tested, we found that the application crashed.

7.5 Discussion of Test Results

The virtual device used for testing could not get past the login page, which made the profile page inaccessible since a token was not retrieved.

8. Conclusions

8.1 Summary

The current method for selling and purchasing textbooks on CSULB's campus is inefficient and time wasting. With BeachTrade on the market, 49ers can quickly find what they are looking for at a cheap price. This social media platform will keep students connected and allow them to excel in their classes.

8.2 Problems Encountered and Solved

-Back button on device going back to login page when the user has already been authenticated

-App crashing in vertical view

- Application slowdown on certain devices when a large picture size is uploaded.

8.3 Suggestions for Better Approaches to Problem/Project

Unit testing and researching common problems with certain layout implementations.

8.4 Suggestions for Future Extensions to Project

Future extensions to this project include improving portability. Development on IOS will allow a greater audience to connect and interact on campus. With an app marketed to college students, IOS development is a must.Adding an end to end encrypted chat between users would make the app more user friendly and improve practicality for this to be used on campus by a large number of users.