

## PART A: Final Project Description and Documentation

**Project:** BestPrice

**Individual Report for:** **James Valles**

### **BestPrice Team Members:**

James Valles

Haonan Chen

William Schroeder

**Video Presentation Link:** <https://www.youtube.com/watch?v=cGWImCWEX6U> also uploaded to D2L

### BestPrice Overview

BestPrice is a shopping app for iOS that ensures you're always finding the *best price* on the hottest items. Whether you're searching for groceries or looking to reward yourself for a job well done, BestPrice guarantees that you never pay more than you need to at brick-and-mortar stores. Simply sign into the app using Facebook or email, then scan items from your local retailer to your heart's content. BestPrice will show you the cost of each item from stores around the web and even let you place an order from within the app. In a rush? Best price lets you save the items to a Favorites list for later viewing. With BestPrice, you'll never again need to open Amazon, Walmart, Target, or any other online retailer to be sure you're getting a good deal!

### Loading Instructions

The application used several features from open source iOS developer community called "[Cocoapods](#)". All the dependencies are already installed in the project folder. **To open project, simply Unzip the BestPrice folder and open .xcworkspace directly, you should be able to compile and run the project. Do not open .xcproject".**

- **Please note:** A request for microphone permission may pop when launching on the iOS simulator. This is a known interaction between macOS 10 and Xcode 10 when using the AVFoundation framework. Our app does not need access to the microphone.

**If pod dependencies for some reason don't show, please follow third party packages installation:**

- In your terminal, type "sudo gem install cocoapods".
- In your terminal, type "pod setup".
- In your terminal, "cd" in the directory where your project located at.

- In your terminal, type “pod install”
- Now, all the dependencies are installed in the project folder. Open the project by click on “.xcworkspace” rather than “.xcodeproj”.

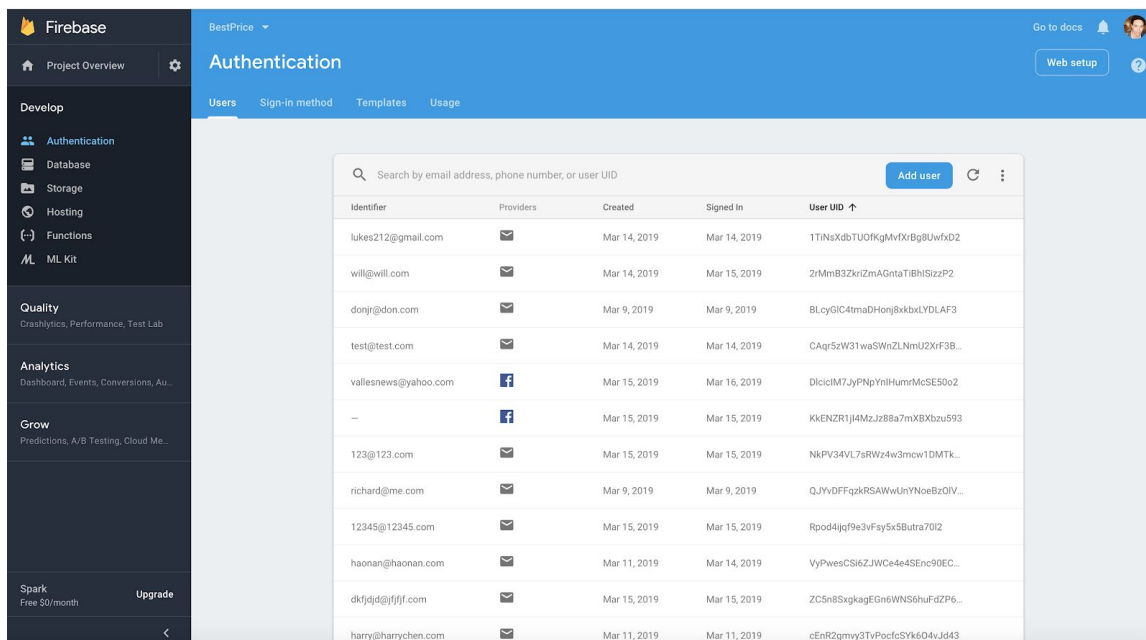
## **Frameworks & APIs Used**

The BestPrice application uses a number of APIs. These include (images below):

- Facebook Authentication
- Email/Passsword Authentication using Firebase
- Firebase database to persist user’s Wish List
- UPCItemdb API to retrieve product data
- SwiftyJSON to parse JSON from UPCItemdb
- AVFoundation to play opening video
- AVFoundation for camera/barcode scanning
- Alamofire to handle HTTP request
- WBLoaingViewIndicator.

## **Facebook Authentication with Firebase integration**

To log into the BestPrice application users can choose to user their Facebook account. Facebook authentication is used following the Facebook Login flow to manually pass the resulting OAuth access token to Firebase. Firebase then verifies those credentials and returns a response. Using theFBSDKLoginButton object, we set a delegate to receive login and logout events.



The screenshot shows the Firebase Authentication console for a project named 'BestPrice'. The left sidebar contains navigation links for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality, Analytics, Grow, and Spark. The main content area is titled 'Authentication' and includes tabs for Users, Sign-in method, Templates, and Usage. A search bar at the top of the Users tab allows searching by email address, phone number, or user UID. Below the search bar is a table listing users with columns for Identifier, Providers, Created, Signed in, and User UID. The table contains 13 rows of user data.

Identifier	Providers	Created	Signed in	User UID
lukes212@gmail.com	Google	Mar 14, 2019	Mar 14, 2019	1TINxXdbTUOfKgmVfXrBg8UwfxD2
will@will.com	Google	Mar 14, 2019	Mar 15, 2019	2rMm83ZkriZmAGntaTiBhISizzP2
donjr@don.com	Google	Mar 9, 2019	Mar 9, 2019	BLcyGIC4tmaDHonj8kxLxYDLAF3
test@test.com	Google	Mar 14, 2019	Mar 14, 2019	CAqr5zW31waSWnZLNmU2Xf3B...
vallesnews@yahoo.com	Facebook	Mar 15, 2019	Mar 16, 2019	DlciolM7JyPnpYnHhUmMcSE5o02
—	Facebook	Mar 15, 2019	Mar 15, 2019	KkENZR1j4MzJz88a7mx8Xbz593
123@123.com	Google	Mar 15, 2019	Mar 15, 2019	NkPV34VL7sRWz4w3mcw1DMTk...
richard@me.com	Google	Mar 9, 2019	Mar 9, 2019	QJYvOFFqzRSaWwUyYNoeBzOIV...
12345@12345.com	Google	Mar 15, 2019	Mar 15, 2019	Rpod4jq9e3vFsy5x5Butra70I2
haonan@haonan.com	Google	Mar 11, 2019	Mar 14, 2019	VyPwesCSi6ZJWce4e4SEnc90EC...
dkfjdjd@jiffj.com	Google	Mar 15, 2019	Mar 15, 2019	ZC5n8SxgkagEGn6WNS6huFdzP6...
harry@harrychen.com	Google	Mar 11, 2019	Mar 11, 2019	cEnR2gmvy3TvPocfcSYk6O4vJd43

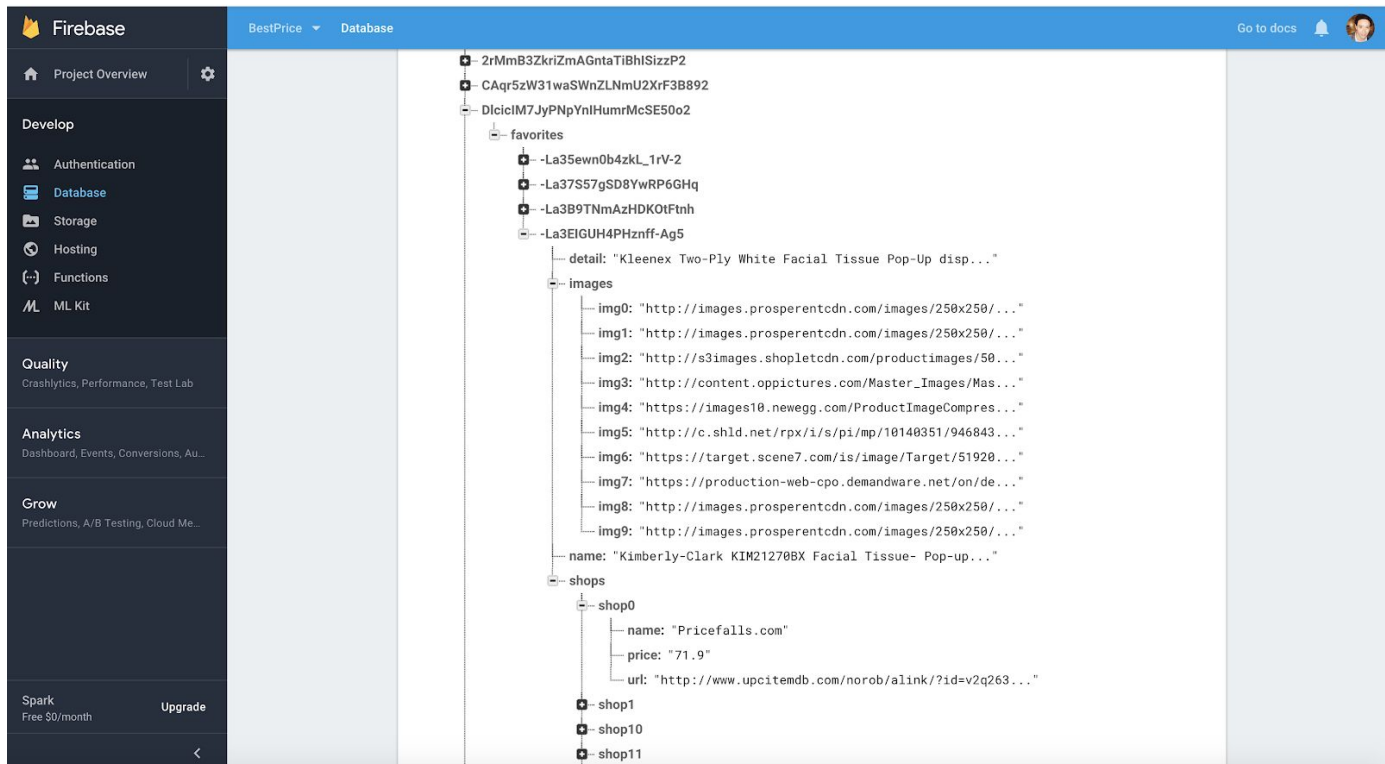
## **Email/Passsword Authentication Firebase**

The BestPrice application also allows users to create and log into their account using a custom email/password that is authenticated and stored in Firebase. Firebase Authentication allows users to authenticate with Firebase using their email addresses and passwords, and to

manage the apps password-based accounts. After a user signs in for the first time, a new user account is created and linked to the credentials—that is, the username and password. This new account is stored as part of BestPrice Firebase project, and is used to identify a user.

## Firestore database to persist users Wish List

We use Firebases Realtime Database to persist the user's Wish List data, and custom Display name for users logged in with their email and password. All Firebase Realtime Database data is stored as JSON objects. When a user clicks on the “Favorite” heart, the selected item's name, detail, images, retailers that carry the item, price, and web links are stored in the database (see image below).



## UPCItemdb API

Lookup by HTTP GET

GET

REQUEST URL

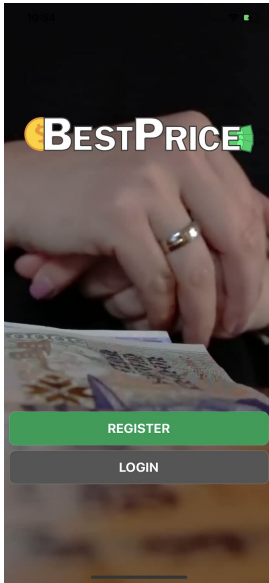
```
https://api.upcitemdb.com/prod/trial/lookup?upc=808124121041
```

RESPONSE BODY

```
{
  "code": "OK",
  "total": 1,
  "offset": 0,
  "items": [
    {
      "ean": "0808124121041",
      "title": "Mrs. Meyer's Clean Day 12.5-fl oz Lemon Verbena Hand Soap",
      "description": "Contains aloe vera gel, olive oil, and natural essential oils; Paraben-free; Ingredients are at least 98% naturally derived; Paraben free",
      "upc": "808124121041",
      "brand": "MRS MEYERS CLEAN DAY",
      "model": "808124121041",
    }
  ]
}
```

The BestPrice application makes several API calls to UPCItemdb. UPCItemdb is a UPC database where you can validate or lookup a UPC along with product information. Once we make a call to the API, we use SwiftyJSON to parse JSON from UPCItemdb. This is used each time a user either scans or manually enters a UPC barcode.

## **AV Foundation used to play opening video**

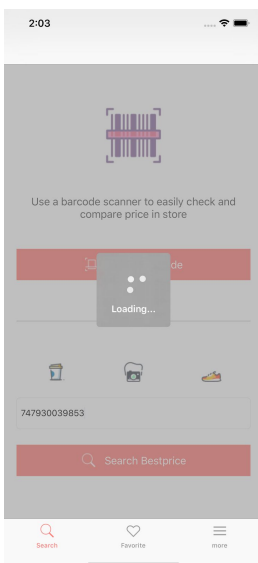


AVFoundation is used to play the opening video in the application initial scene. The video playing was taken from a royalty free stock video provider - <https://www.videvo.net/>. The video file is stored in .mov format. `playerLayer.zPosition = -1` allows us to place our application's logo over the video.

## **Alamofire framework to handle HTTP request**

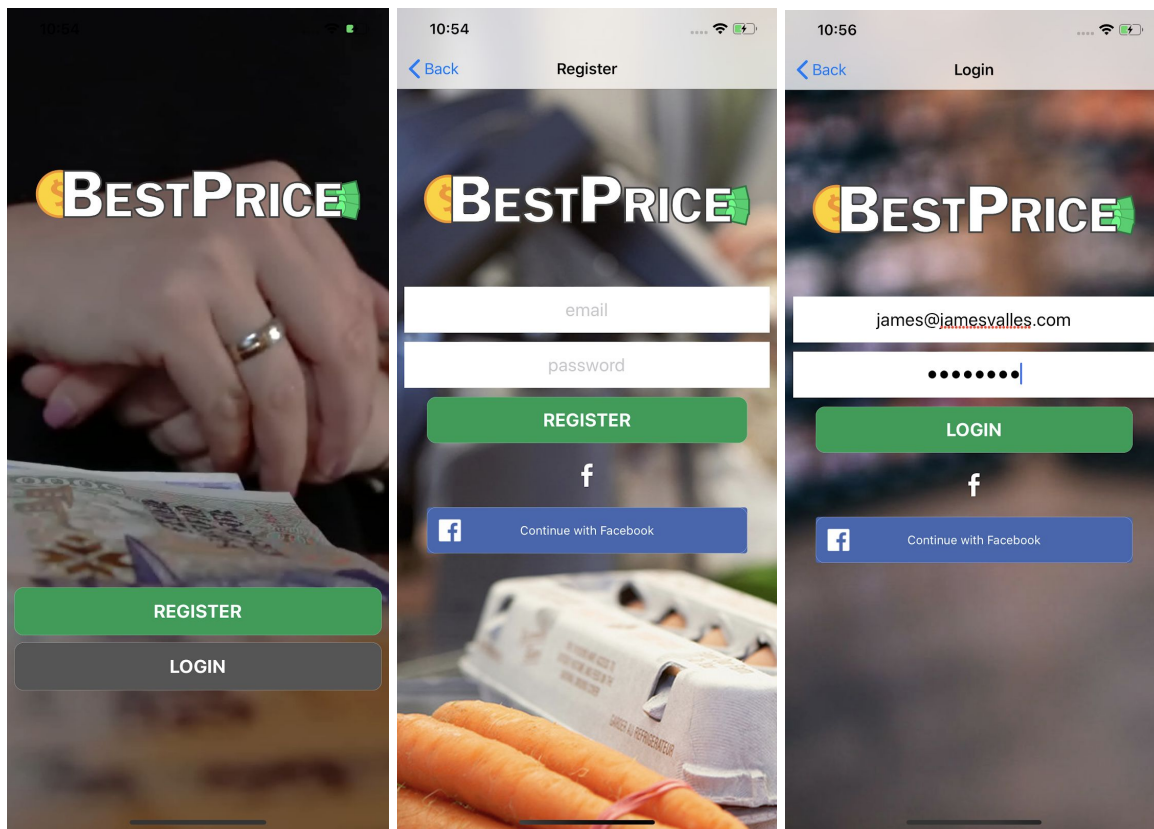
Alamofire is an elegant networking package written in swift which handling HTTP request and response by incorporate response validation and response chaining together so that you don't need to write your own closure to handle different HTTP response. Functions that have been used in BestPrice application includes HTTP methods, parameter encoding, HTTP headers, Authentication. For more information, please go to [Alamofire](#) to review.

## **WBLadingIndicatorView used to display loading animation**



WBLadingIndicatorView contains a set of loading animation indicator implemented based on CALayer written in objective-c. It provides interfaces to initialize indicator, choosing animation style, size and color of indicator, and when are how to stop animation. Features that have been used in BestPrice application are `WBLadingAnimationBallPulseType`. For more information please go to [WBLadingIndicatorView](#) to review

## Scene Descriptions

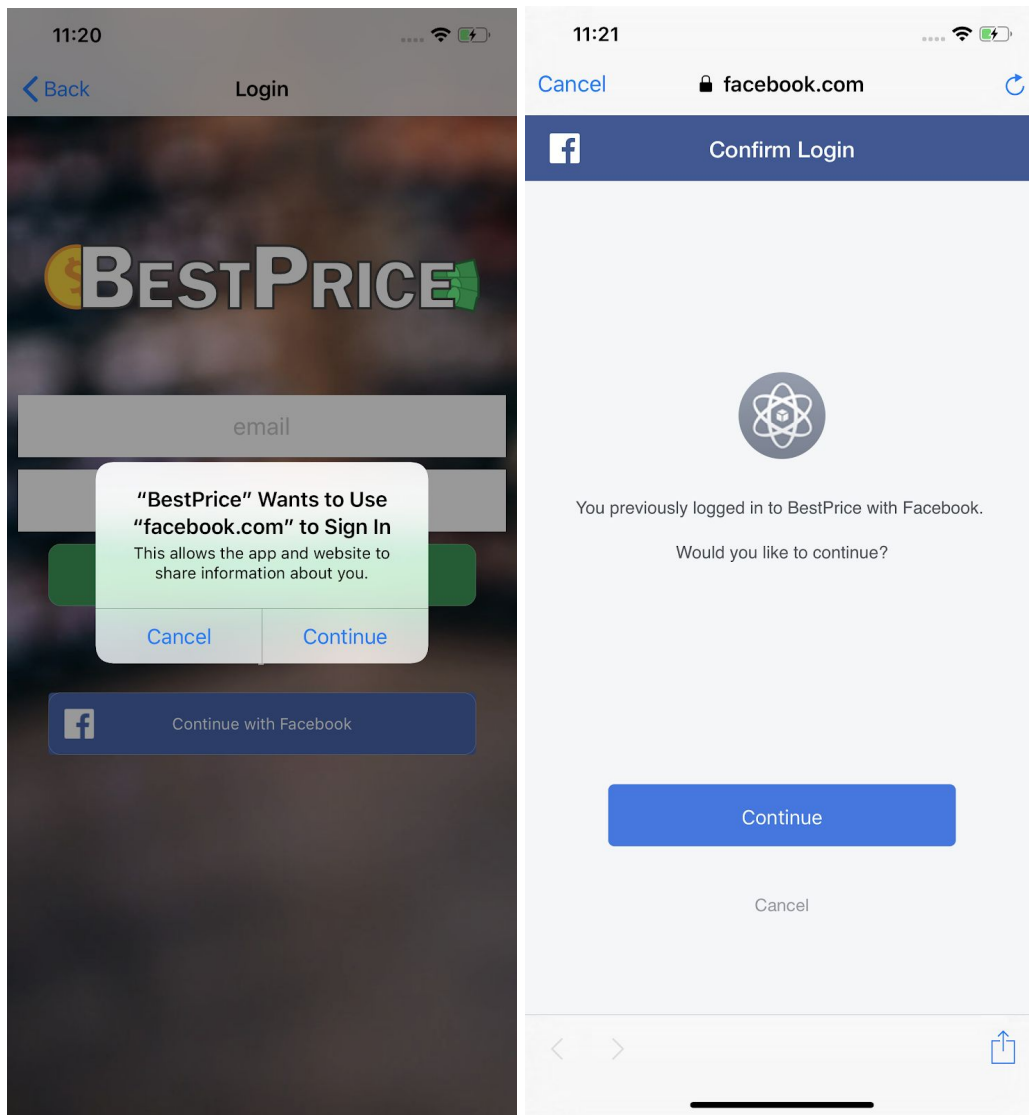


### App's Home, Registration, login scene

When you first launch the app, you will be given the option to Register or Login. The scenes above are (from left to right): startup screen, registration screen, login screen.

- If it is your first time using the app, choose the Register button. You will be redirected to a registration screen where you can set up your account. Make sure to choose a strong password! After you've entered your account information, the app will sign you in and you'll be able to scan.
- Returning users will want to choose the Login button. The login page will prompt you for your previously entered account information. Once you successfully login, your saved searches and account details will show up just as you left them.
- Alternatively, you can opt to sign in with Facebook. Both the Register and Login screens allow you to link a Facebook account to save your information.

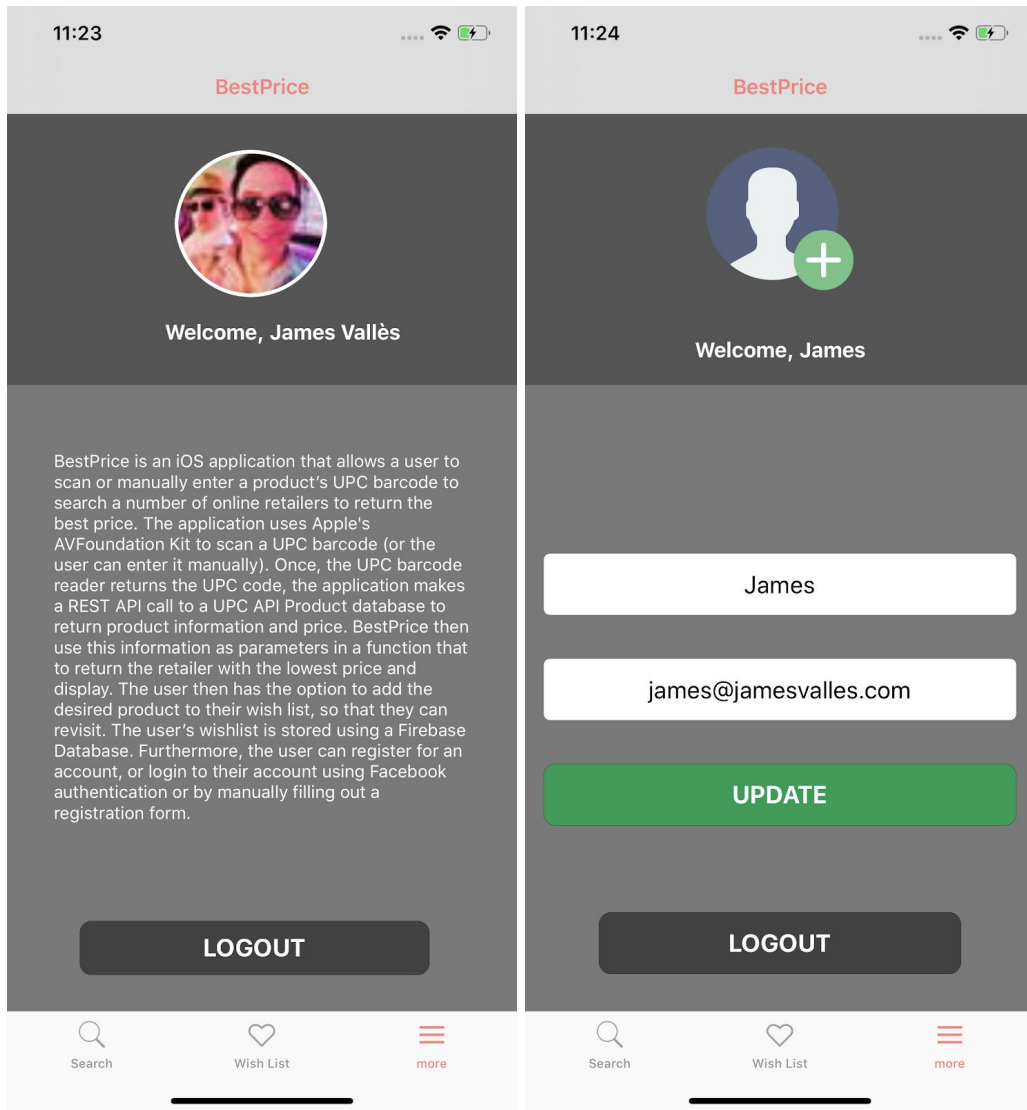
## Facebook Authentication



The BestPrice app allows users to login using Facebook with just a few quick taps. Simply select the Continue with Facebook button on either the Login or Register screen, then give BestPrice permission to use your Facebook information. From there, you need only follow the prompts on the screen to link your account. Once you login using Facebook, the option to continue with your previously signed-in account will be available.

**Please note:** the app will collect your name and profile picture from Facebook.

## User Profiles

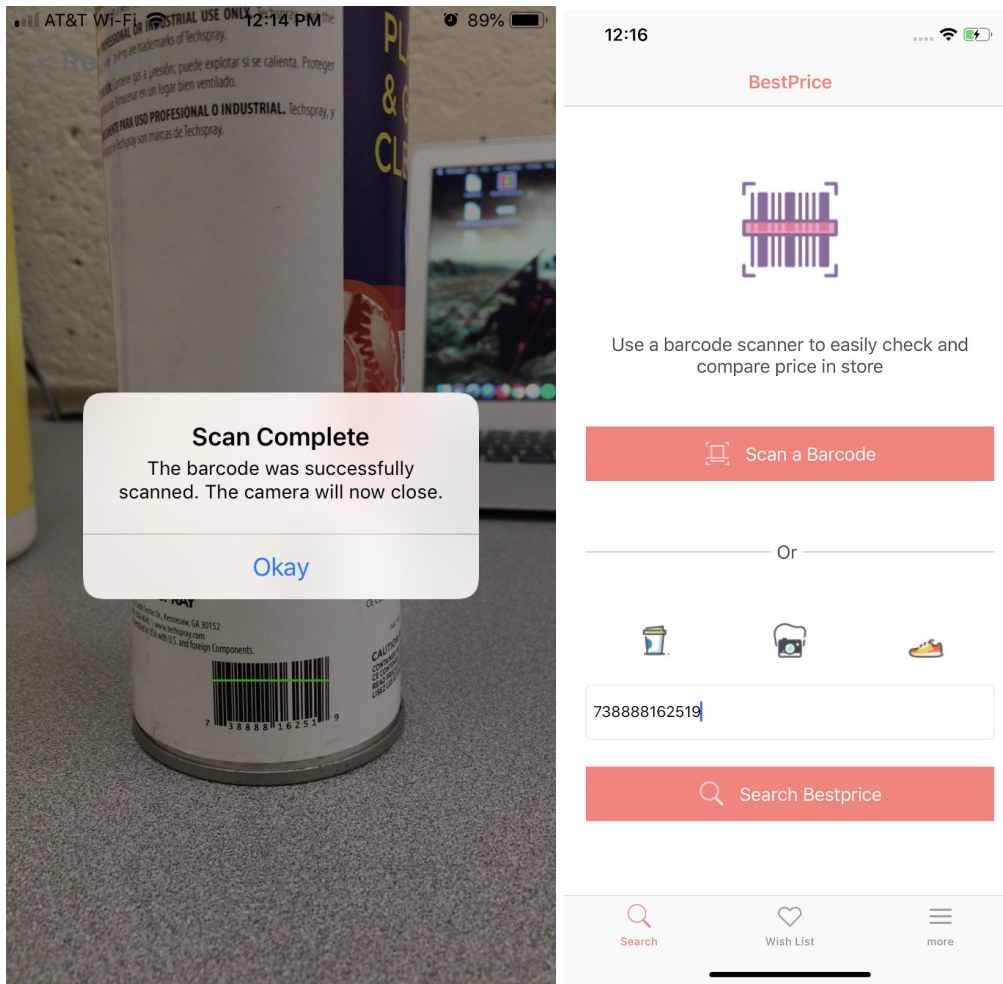


You will be presented with one of two profiles screens in the More tab, depending on how your profile was created. Both options allow the user's data to persist in the database for quick loading of the More screen.

- If you used Facebook to link an account, your user data will be automatically loaded into the profile page. The About Us information will be presented to you since you do not need the option to change your email address or name.
- If you created a new account using an email address, you will instead be given the option to add a profile name and update your email address.



## BarCode Scanner



Upon opening the app, you will be given the option between scanning a product or manually searching the database via text input. If your device does not support camera use, you will need to use the manual entry method.


1. The scanner can be opened by pressing the Scan a Barcode button. The button will launch a camera which can be closed using the return button in the upper-lefthand corner of the screen.
2. When you scan an item, a green bar will appear over the barcode. The app will alert you once the scan is complete and confirm that you are being returned to the scan menu.
3. In the scan menu, you can edit the text field to make any necessary changes to the barcode. If for any reason the code does not match the one on the item, change it now.
4. After confirming that the barcode scanned is correct, or after entering the code manually, press search to locate a list of shops carrying the item. You will be redirected to the results screen.



## Wish List


11:18

BestPrice




Go Jo 3275515 10 oz  
Purell Advanced  
Hand Sanitizer Refr...

Best Price:  
\$2.99



Expo 8oz Dry Erase  
Surface Cleaner  
Spray Bottle

Best Price:  
\$2.67



Mrs. Meyer's Clean  
Day 12.5-fl oz Lemon  
Verbena Hand Soap

Best Price:  
\$0.0

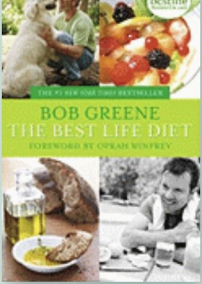
Search

Wish List

more

11:30

BestPrice




BEST PRICE  
\$0.99

Alibris

\$0.99

Detail

Favorite



Bett...

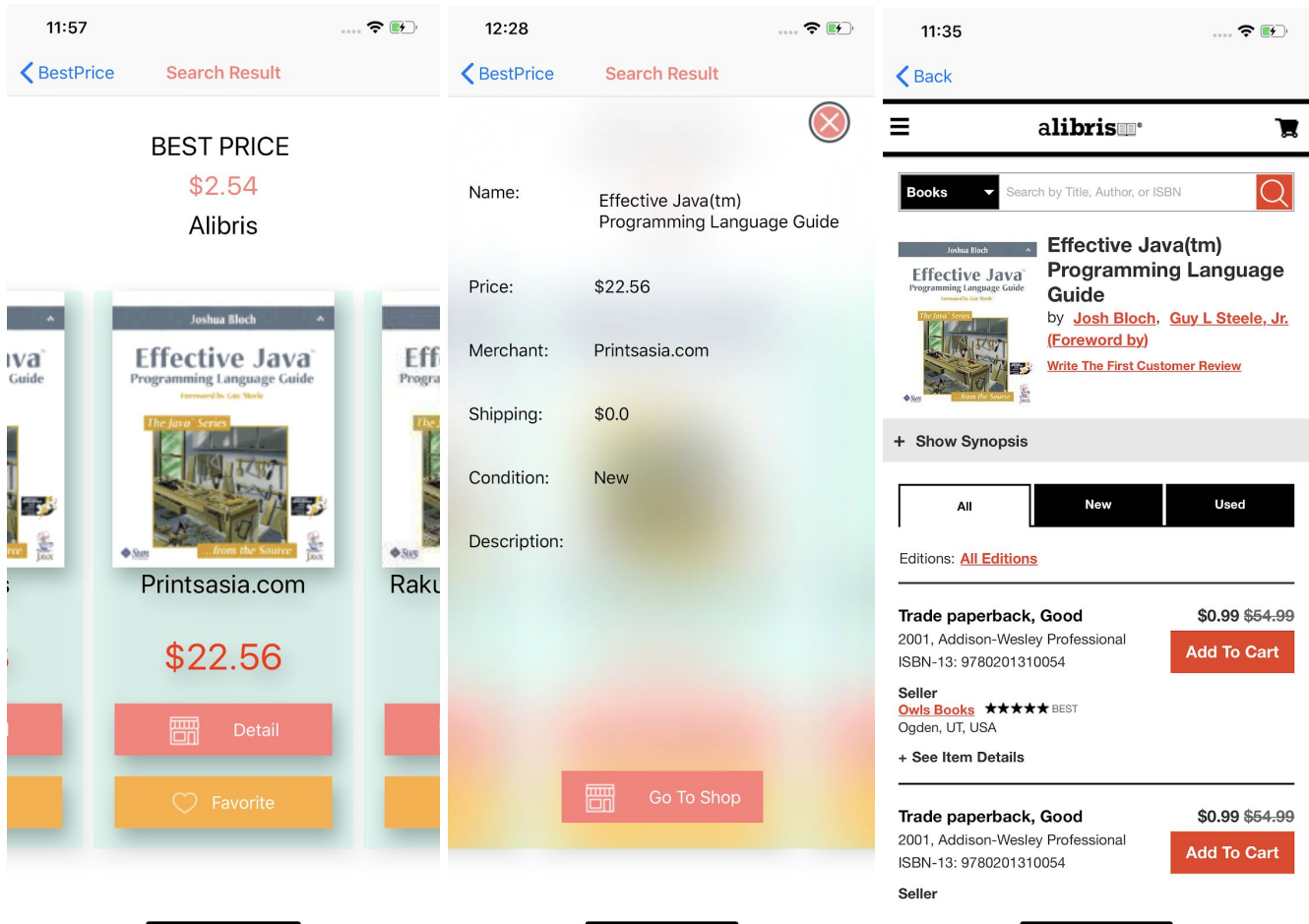
Search

Wish List

more

The wish list retrieves all favorites saved by the user. All favorited items are saved in a database, so they will never be lost due to switching apps or even changing phones. When you enter the favorites tab, the information is loaded into a table view, and you can then select a cell to show a carousel of vendors who carry that item.

## Item Search, Details & Go To Shop

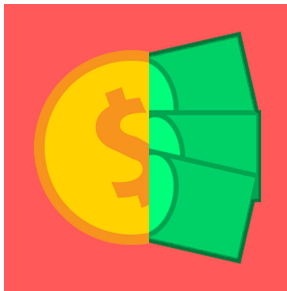


After searching for a product, you will be able to scroll through the vendors who carry the item. Each vendor cell lists the name of the vendor and the price of the item. If you would like to view more information on a specific product, simply press the Detail button. You can also visit the vendor's webpage from the Details screen.

Additionally, you can add an item to your favorites list from the item results. Tap the Favorite button to add that specific item to the app database. You will now be able to access that item via the favorites tab from any device you sign into in the future.

## Logo

The BestPrice application features an original logo that was designed to reflect the overall purpose of the app: to ensure our user's are always finding the *best price* on the hottest items.



## Features used in implementation

1. Multiple view controllers such as TabBarController, NavigationController, TableViewController, and CollectionViewController are used and connected by segues.
2. Data can be passed by using segue flow and protocol delegate implementation, which includes but not limited to UIApplicationDelegate, FBSDKLoginButtonDelegate, UITextFieldDelegate, etc.
3. Carousel slider was created by customizing UICollectionViewCell using 2D graphics and animation toolkit
4. Blurred background and popup transition when click "View detail" was implemented using visual effect view and scrollView delegate.
5. Loading indicator and background effect shown when data is getting back from server in asynchronous function by using pod "WBLoadingIndicator"
6. Asynchronous API call was implemented using pod "Alamofire" networking packages.
7. Data that returned from asynchronous function was handle by dispatchqueue and multithread function provided by swift languages
8. Views includes TableView and CollectionView are populated using data returned back from API
9. WebView are used for directing user to shop website.
10. Asynchronous API call was implemented to persist/retrieve user's Wish List product data in Firebase
11. Favorite items information under each user rendered in tableView
12. Used FBSDKLoginKit/Firebase to make an Asynchronous API call to authenticate users who choose to login via their Facebook accounts
13. Used Firebase authentication to make API call to allow users to create or login to a BestPrice user account
14. Used segue flow to check if user is logged in and to allow them to bypass login process
15. Used AVFoundation library to play video on application's initial scene background
16. Used StackViews and other views (text fields, image views, labels, button, etc. ) along with constraints to present information that is auto layout compliant on devices of different sizes
17. Used AVFoundation to user cellphone's camera to scan barcode to retrieve product's UPC which is then passed into the search text field

## **PART B: Final Project Discussion**

Working on the BestPrice iOS application has truly been one of the most rewarding projects I have worked on at DePaul. It has allowed me to solidify topics learned in other classes along with new topics I have explored in this class to work as a team and develop a useful and fully functional application. Moreover, it has given me the opportunity to learn and implement new features not discussed class.

For this project, not only did I fully design the UI for the application's Home, Registration, Login, Facebook Profile, and Username Profile scene, I also helped Haonan develop the Wish List, where I used Firebase's realtime database to not only store a user's Wish List items, but retrieve them, so they could be injected into the FavoriteTableViewCell and be displayed in the FavoriteViewController which implements UITableView.

Retrieving and storing data was definitely a challenge for me at first, since I knew absolutely nothing about a NoSQL database and had learn everything from scratch. I had to learn how to post data in a JSON tree structured format, how to retrieve data in a JSON tree structured format, and then parse data without using an external library from multiple formats including Arrays, Dictionaries, Strings, Doubles, to create the Merchandize and Retailer objects so they could be injected into the tableview cells. This work gave me some practical experience I will for sure use on the job. Here is how the storing and retrieval process using the Firebase API works. In the ShopCardCollectionViewCell which implements UICollectionViewCell, on line 35 there is an IBAction that is connection to the Add to Favorites button. Once the button is pressed, the function I wrote addToFavorites() is invoked. This function uses the userID variable which is the current user logged in and create a new node under the child favorites see line 44. To add images, you add another branch and call the .setValue() method from Firebases API: `new!.child("images").setValue(array)`. This will store the user's favorites under their user id, and each favorite has a unique key, which is generated involving childByAutoID() see line 43. To retrieve data, on line 93, of the FavoriteViewController, I have written the function loadfavorites(), which retrieves a user's favorite items, which is invoked from within the getFavorite() function, which is called whenever the viewillappear.

Aside from making API calls to Firebase, I also use Firebase's OAuth authentication and Facebook's FBSDKLoginKit to create a mechanism to register and authenticate the app's users. The process involves retrieving an OAuth token from Facebook and passing it to Firebase for authentication. Again, it took a lot of patience to figure out how all this worked, including setting up an App on Facebook's Developers site, first, and then working with the UIApplicationDelegate, FBSDKLoginButtonDelegate, UITextFieldDelegate and both Facebook and Firebase OAuth API to achieve this registration and login functionality. My work in the end allows a BestPrice user to create a BestPrice account using Facebook or to manually enter their email address and custom password. They are then able to login using these credentials or via Facebook. Not only did I implement the authentication process, but I also had to figure out the logic to prepare a segue once login is successful. Moreover, if a user is already logged in via Facebook, even if they close the app out entirely, the user automatically bypasses the login process and is redirect to the UPCInputViewController. Once a user is logged, Firebases auth API allows you to retrieve the user's provider data by typing: `Auth.auth().currentUser?.providerData`. You can do a for loop on provider data and then invoke `.providerID()` to retrieve how the user was logged in. For example, if the user is logged in via "facebook.com", you are able to then able to run specific code that pertains to that user.

Since, I used FBLoginKit in conjunction with Firebase to obtain a authentication token, I also used Facebook's login and logout button. Because their button was much different then the button's I had created for the RegisterViewController and LoginViewController, I had to figure out how to disable Facebook's button constraint so that I could use my login button. In the end I

created the class FBSDLLoginButton, which allows me to override Facebook's button constraints.

Aside from authentication, I worked with several iOS frameworks including the AV Foundation. I have noticed a number of trendy apps, which play a background video on the app's initial scene so I decide to explore this.

In the process, I was able to get a .mov video to play, in muted state using the AVFoundation framework, first hand. The UIViewController (also the app's initial scene) invokes the function playvideo() whenever viewWillAppear is invoked. The code for the function begins on line 58.

During this process, aside from the code, I learned the importance of choosing the right video by making sure the file was compressed as size does matter.

Lastly, I had the opportunity to explore and install some of the pods offered in the wonderful world of Cocoa Pods. I learned how to add pods to my podfile and how to install them using terminal. Pods I used include: 'Firebase', 'Firebase/Auth', 'Firebase/Database', 'FBSDKLoginKit'.

During the course of development, I also implemented several topics I learned in class including the following: how to resign a keyboard when using textfields, nesting text field in a collection view (the text field were implemented on the user's login, registration page, and user's profile page (if logging using an email)).

Additionally, on that profile page, I created a field called name which allows a user to upload their display name. Once the user press updates, refer to line 59 and the function updateInfo() on AltAccountViewController, the users display name appears in UILabelView and is also stored in the database. I wanted to include updating username and creating a new password using a similar technique, but ran out of time. On the profile page, I also worked with the scrollable textfield, used Stack Views and constraints to achieve auto layout. Furthermore, throughout, I used IBAction, Outlets, several different kinds of views, and Segues to navigate from multiple view controllers, and helped Harry setup the following controllers: TabBarController, NavigationController, TableViewController.

Among some of the biggest challenges I encountered during the development project include: learning new material when time is extremely limited, figuring out version control when collaborating with a team of three, and auto layout.

First, I definitely felt the pressure of a deadline. Along with working on class assignments, I decided to take on tasks, some of them which were not discussed in class. This required time outside of the classroom to learn these features. I am so proud I did, but time was definitely an issue as I work 40 hours a week and am attending DePaul full-time. While there were times I became frustrated, I recognize persistence is the key to success. No gain unless there is pain. So, in the end, it was well worth it.

My next biggest nightmare was version control, especially when working with a team of three. Resolving conflicts became an issue when a team member inadvertently deleted all of my changes rendering the master branch useless. We had to troubleshoot on how to roll back the merge (moving the HEAD to the correct merge) so that the master branch would be fully functional again. Additionally, I also lost a days worth of work, because I forgot to add files before committing and then made the huge error of discard my changes on local system thinking that I had properly merged my commits to the master branch. I learned some very painful lessons, but but I am glad I was able to resolve them in school than on the job.

Lastly, my biggest challenge was the auto layout! This was a struggle. I found myself pinning stack views over and over, then centering, and literally attempting every trick in the book, only to have a mess appear on the screen over and over again. After hours of practice, resetting constraints and restarting over, I finally realized that having a plan before starting to pin is always helpful. Dr. Jia you were so right about that.

I think there are many ways the application can be improved. For the next release, I will definitely improve the user profile section. Right now, while it is there, it doesn't offer the user

much functionality except for updating the profile name or showcasing the retrieving and displaying the user's facebook photo or name. By the way, that took some time to figure out along with cropping the image as a circle with a border.

I think I could improve the UI and make it appear more professional. I would also do the same for the registration, login pages, and maybe merge them into a single scene and add the ability to retrieve user's password via email. I think the app's detail view, and product view could also be improved. I specifically would improve the carousel. The images fall on a baby blue background, which makes the product's image look boxy and less smooth. I would also work on expanding the app to allow shoppers to seamlessly access the vendor making it easy to purchase the product instead of having to click through multiple scenes. Perhaps, the next release should feature a button directly under the favorite button. I would remove and make better use of the navigation top bar. Instead of displaying the text "Best Price," I would remove it all together. Also, Haonan added the photo indicator icon for photos and added a time delay to show this functionality. In the real world, I would have had this removed, so that the images could display faster without the indicator. Overall, I think the app idea has great potential. I know there are already many similar apps on the market. To make this one better, I would give the user the ability to enter and search a product's name instead of solely relying on a UPC barcode. Also, I would, perhaps, develop a module where we could display featured products (ads) to generate revenue. Last, I would spend additional time refactoring the code base specifically checking for code smells, including duplicate code, and areas where design patterns can be used. I would also spend more time testing. I discovered a number of bugs towards the end of development. But, I feel there are others. So, I would also like to continue testing to make the app less vulnerability.

Overall, I absolutely love developing in Xcode. I found it to be easier than Android Studio. However, there were some limitations I did experience over the course of development. I really wish there was a better way to read the log. It is packed with so much information making it hard to decipher what is going on. Additionally, I had problems sideloading my iPhone. It appears Apple was experiencing some sort of outage. It would be nice to receive notifications if changes are made instead of learning the hard way. Overall, I really enjoyed Xcode. I really found the code suggestions (prompted suggestion to fix my code) extremely helpful. Often time I would call an old method that were renamed, and Xcode would automatically tell me what was happening and how to fix it by clicking the fix button. I loved the flexibility and ease in designing the UI. Instead of using XML in Android, I felt I had greater control to visualize what I was creating and the ability to place views precisely where I would want them. Of course, I had to use constraints, but the flexibility was nicer than that of Android Studio. Moreover, I really appreciated the way Xcode is laid out and how I can easily minimize sections of the interface. I found the storyboard features to be incredibly helpful and the ability to change things using the attribute inspector much simpler.

I started off 10 weeks ago knowing absolutely nothing about iOS design and over the last few weeks I have worked extremely hard at learning and implementing the basics. At the end of this course, I feel with more practice and time, I would not only enjoy becoming a full-time iOS developer, but I feel I could be really good at it. I've always wanted to build an iOS app, and now I can say I've done 9 of them. Yes, there are small, but I have learned many concepts in each one. I cannot thank you enough Dr. Jia for a great class. From learning how to use Xcode, to learning about iOS architecture, the swift language, the powerful optional type, adding widgets, constraints, class inheritance, table views, customize prototype cells, you name it, and even some history about Steve Jobs, I found this class to be well worth it. Thank you.



## **PART C: James Valles' specific list of individual contributions to the team**

1. Designed the following scenes: Home, Registration, Login, Facebook Profile, and Username Profile scene (I added the logo Will Schroeder designed).
2. Developed and worked on multiple view controllers including: Account View Controller , AltAccountView Controller, RegisterViewController, LoginViewController, FBSDKLoginButton class, FavoriteViewController (Haonan Chen added some logic to this controller as well), FavoriteTableViewCell (worked with Haonan Chen to retrieve data to populate cells, and Shopcardcollectionview (added function to persist item to database in JSON tree format. These scenes are connected by segues: loggedIn, goIn, toTabController, bypassLogin, goUser.
3. Worked with Hoanan Chen to configure segue flow and protocol delegate implementation, which includes but not limited to UIApplicationDelegate, FBSDKLoginButtonDelegate, UITextFieldDelegate.
4. Asynchronous API call was implemented to persist/retrieve user's Wish List (favorite items) product data in Firebase to assist Haonan Chen populate cells in TableView and CollectionView
5. Used FBSDKLoginKit/Firebase to make an Asynchronous API call to authenticate users who choose to login via their Facebook accounts
6. Used Firebase authentication to make API call to allow users to create or login to a BestPrice user account
7. Used segue flow to check if user is logged in and to allow them to bypass login process
8. Used AVFoundation library to play video on application's initial scene background
9. Used StackViews and other views (text fields, image views, labels, button, etc. ) along with constraints to present information that is auto layout compliant on devices of different sizes
10. Assisted Will Schroeder and Haonan Chen in fixing bugs, developing documentation, reformatting code
11. Worked with Will Schroeder and Haonan Chen to plan project idea and used git for version control to streamline development process
12. Added app icon to app
13. Helped team with editing together final video

