Final Report for Dynamic Pricing - Bideal

Lyujia Zhang(bz2269), Guitang Lan(gl2510), Baochan Zheng(lz2467), Peiran Zhou(pz2210)

1. Motivation

What is Dynamic Pricing?

Dynamic Pricing is when prices change based on supply and demand. In contrast to static pricing, where a retailer will for example print a price tag and never change that price, dynamic pricing adjusts based on many factors in order to capture the most value - and deliver the most value - to customers.

What is the advantage?

Dynamic pricing enables companies to capture more value - and deliver more value. Frequently upon implementing dynamic pricing, overall prices only rise 1-2% but revenue and profitability increase as much as 10%. This is because dynamic pricing both is a form of price discrimination, charging some people more (early adopters, business travelers, etc.) and also improving the utilization of underutilized assets.

2. Overview

We designed a bid system from the customer side. With our mobile application, customers can scan the barcode of the item they want, check online price as preference and then bid online. There are three bidding chances for an item per person per day. If bid successes, we will generate a QR code which denotes the bidding information, including item, price and time, to process the deal. In this way, both customers and sellers can save time, since they don't need to argue a lot on the price.
Our design has a bright future. This kind of application is not popular in US, but it's indeed useful, especially nowadays, when people don't really have so much time to waste. Similar application has been proved to be a success in India. We have a good reason to believe ours is a meaningful and useful one.

3. App Description

Bideal is an IOS App to help the business implement dynamic pricing strategy. It is an application for all potential customers. The first functionality of the App is provide discount information. Then second and the most important functionality is host the dynamic pricing bidding. By using Bideal, the user can scan the barcode of the product through cell phone, then the app will tell users whether the product is under dynamic pricing. For all products under dynamic pricing, user has three chances to bid the product. As long as the bid price is higher than the minimal acceptable price, the user succeeds to win the deal. To help users to bid price, the app will also provide some guidance of price information by searching through AMAZON and EBAY. If user is unfortunate to lose the deal, he/she can try that

again after 24 hours. On the other hand, if user win the deal, he/she will receive coupons which will be presented to Cashier. These coupons are QR barcode that can be scanned by the cashier.

4. Architecture

The frontend is built with object C while the backend is built with node.js. The backend server is hosted in AWS Elastic Beanstalk and the database is hosted in AWS RDS. The frontend communicate with the backend through http endpoints provided by backend server.

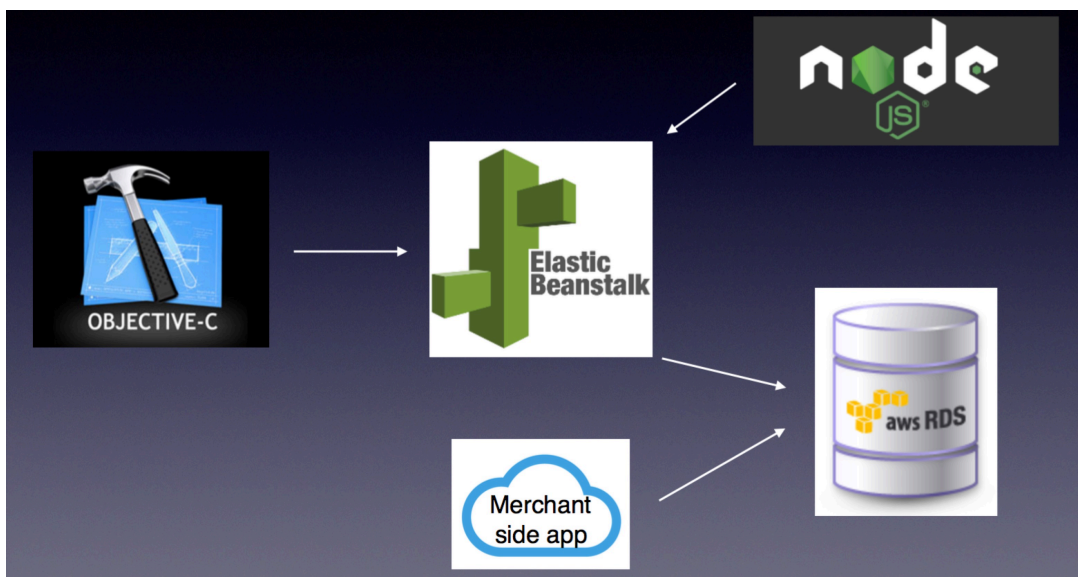


Figure 4.1 System Structure

4.1 Frontend

It contains five main components : login system, recent discount module , barcode scanner, qr barcode generator and bidding module. Login module include components for new user sign up, login, forget password, and reset password; discount module mainly show the recent discount information from dealmoon; barcode scanner focus on scan barcode from product and query the server whether the product is under dynamic pricing; bidding module is the primary module. It first checks the the eligibility of the user for bidding the product (if the user has bidden before and in 24 hour, he/she may be ineligible), then provide three chance for the user to bid the price. At the same time it offer user help to consult the price from AMAZON and EBAY. Finally after winning deals, the qr barcode generator will generate QR barcode coupon.

Figure 4.2 iOS Development Environment & Xcode Development Tool

4.2 Backend

It plays the role as a bridge to connect the frontend and database. It provides various endpoints to support the frontend functionality. It will process the task delegated by the frontend and may query the database or log transaction history to the database. We also use the database to communicate with merchant side application. It is a better way than directly call the API of merchant side application, because it decouples our customer side app from the merchant side app.

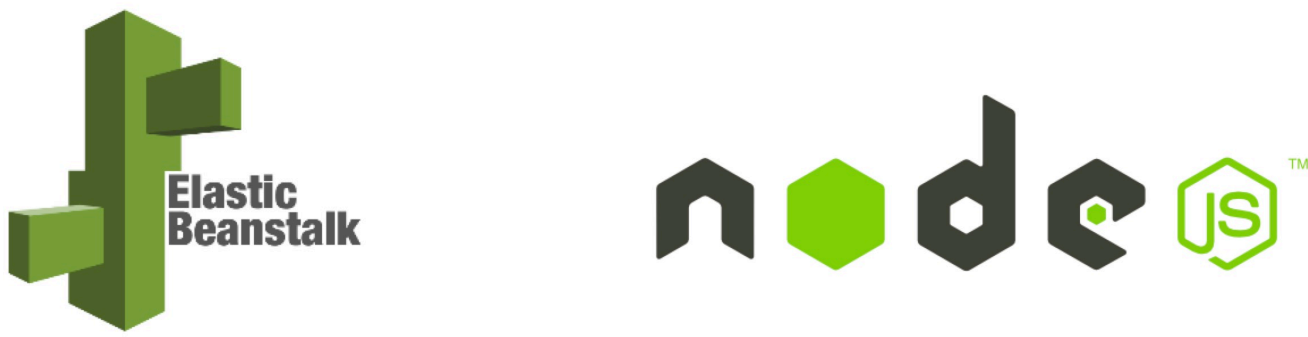


Figure 4.3 AWS Elastic Beanstalk & Node.js

4.3 Database Design

AWS RDS MySQL database was utilized to store our data. We have five tables defined in our schema, which are User, Business, Item, Buy and History. The attributes of each table are defined as the figure shows, with primary keys underlined. In the Buy table, we defined bid as business id, itemid as item id and price as the lowest price business offers to its customers for the certain item. When a deal is made, a record would be added to History table, with information such as buyer's username, bid price, item id, business id and timestamp. Through this schema, we can successfully compare the bid price with the lowest price to make a deal, and also manage a user's transaction history in account page.

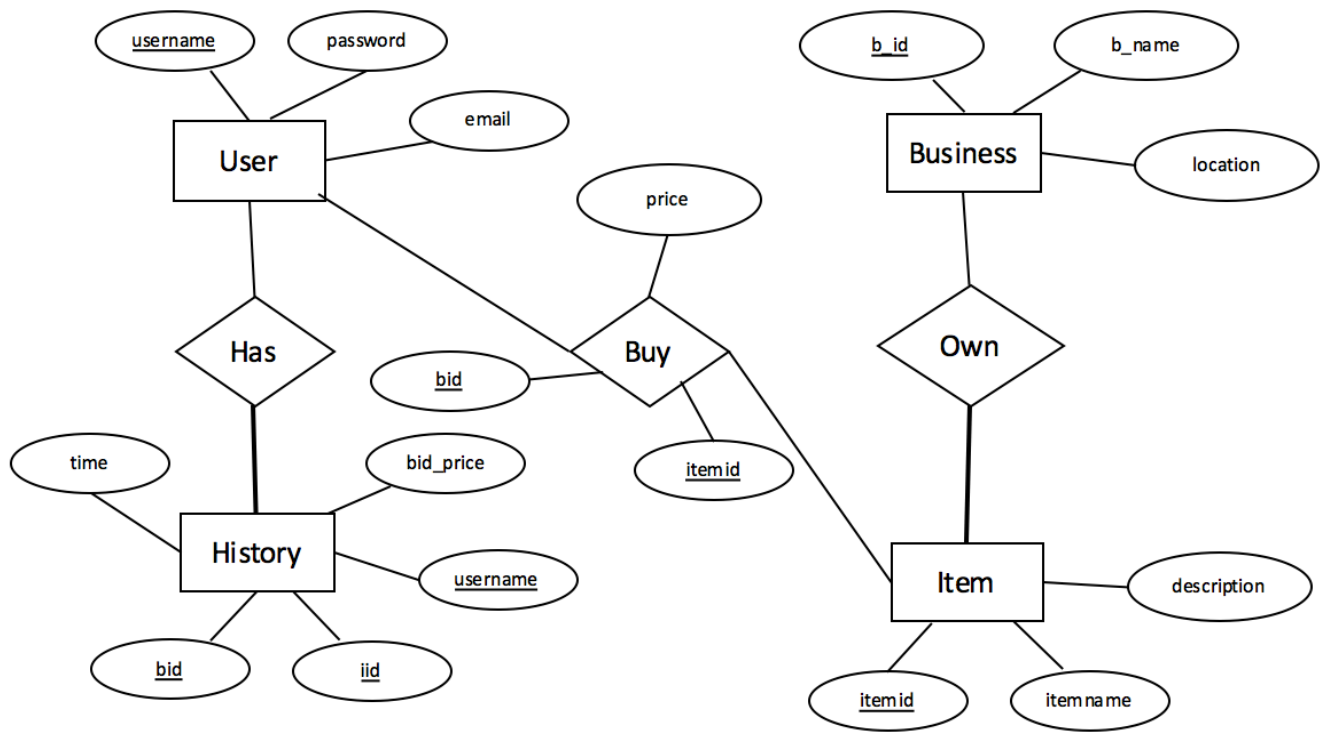


Figure 4.1 Relational Database System - SQL Schema

5. Code illustration

5.1 Login Module

This module concludes three view controllers - Login Viewcontroller, Forget Password View Controller and Register View Controller. The old user can directly log in the application by typing into the username and password. For the new users, they have to register firstly. During the registering process, the username can just register the username that has not been occupied by other users, the password should be matched in twice typing and the email should be based on the traditional email format like: pz2210@columbia.edu, xxxx@yahoo.com. After finishing the register, the user may forget his password, so he can click the "Don't remember your password ?" and entering the username, email address to get back the password.

Core Code:

1. Sending HTTP request, matching the data with database.

```
NSString *urlStr = [NSString stringWithFormat:@"http://dynamicpricing-env.us-east-1.elasticbeanstalk.com/login?username=%@&password=%@", self.
    usernameTextField.text, self.passwordTextField.text];
NSURL *url = [NSURL URLWithString:urlStr];
NSURLRequest *request = [NSURLRequest requestWithURL:url];
NSURLResponse *response;
NSError *error;
NSData *responseData = [NSURLConnection sendSynchronousRequest:request returningResponse:&response error:&error];
NSString *responseStr = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];
```

Figure 5.1.1 HTTP Request

2. Setting Global Value (Like: username, password).

```
AppDelegate *appDelegate = [[UIApplication sharedApplication] delegate];
appDelegate.usernameGlobal = self.usernameTextField.text;
```

Figure 5.1.2 Setting Global Value

3. Setting the Alert View, when the condition is satisfied, the alert view will automatically showed out.

```
UIAlertView *loginAlert = [[UIAlertView alloc] initWithTitle:@"Fail to Login"
                                                     message:@"Your Username or Password is wrong!"
                                                    delegate:self
                                           cancelButtonTitle:@"OK"
                                           otherButtonTitles:nil];
```

Figure 5.1.3 Alert View

4. Check the Email Format

```
-(BOOL) NSStringIsValidEmail:(NSString *)checkString
{
    BOOL stricterFilter = NO;
    NSString *stricterFilterString = @"^[A-Z0-9a-z\\._%+-]+@([A-Za-z0-9-]+\\.)+[A-Za-z]{2,4}$";
    NSString *laxString = @"^.+@([A-Za-z0-9-]+\\.)+[A-Za-z]{2}[A-Za-z]*$";
    NSString *emailRegex = stricterFilter ? stricterFilterString : laxString;
    NSPredicate *emailTest = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", emailRegex];
    return [emailTest evaluateWithObject:checkString];
}
```

Figure 5.1.4 Judging email format

5. Change View Page

```
[self performSegueWithIdentifier:@"login" sender:self];
```

Figure 5.1.5 Transformation between different Viewcontrollers

5.2 Main Page Module

This Module contains two part: Latest Discount Activities, Scan;

2.1 Scan

For the Scan module, people can just directly click the scan button and then choose the market that they are locating. And begin to scan the product. For example, a person may want to buy a bottle of water in macy, so he can just choose macy icon and begin to scan. If the application returns “Scan Failed” which means the product doesn’t have dynamic pricing. If the application returns “Scan Successfully”, which means the item has dynamic pricing. Then the user can begin to bid!

Core Code:

1. Choose which business the user is locating

```
- (IBAction)macy:(id)sender {
    AppDelegate *appDelegate11 = [[UIApplication sharedApplication] delegate];    // Set global value
    appDelegate11.businessnameGlobal = @"macys";
    [self performSegueWithIdentifier:@"WhichBusinessToScan" sender:self];
}
```

Figure 5.2.1 Business Locating

2. Scan Function (AV Foundation Framework)

```
CGRect highlightViewRect = CGRectZero;
AVMetadataMachineReadableCodeObject *barCodeObject;
NSString *detectionString = nil;
NSArray *barCodeTypes = @[AVMetadataObjectTypeUPCECode, AVMetadataObjectTypeCode39Code, AVMetadataObjectTypeCode39Mod43Co
                          AVMetadataObjectTypeEAN13Code, AVMetadataObjectTypeEAN8Code, AVMetadataObjectTypeCode93Code,
                              AVMetadataObjectTypeCode128Code,
                          AVMetadataObjectTypePDF417Code, AVMetadataObjectTypeQRCode, AVMetadataObjectTypeAztecCode];

for (AVMetadataObject *metadata in metadataObjects) {
    for (NSString *type in barCodeTypes) {
        if ([metadata.type isEqualToString:type])
        {
            barCodeObject = (AVMetadataMachineReadableCodeObject *)[_prevLayer transformedMetadataObjectForMetadataObject
                (AVMetadataMachineReadableCodeObject *)metadata];
            highlightViewRect = barCodeObject.bounds;
            detectionString = [(AVMetadataMachineReadableCodeObject *)metadata stringValue];
            break;
        }
    }

    if (detectionString != nil)
    {
        _label.text = detectionString;
        break;
    }
    else
        _label.text = @"(none)";
}
```

Figure 5.2.2 AV Foundation Implementation

2.2 Discount Activity Price Reference

For this part, it can be just used as a price reference module. Dealmoon is a website, aiming to provide the latest item discount activity.

Core Code:

1. Using Master - Detail templates to build the Discount Activities View Structure;

2. Using Table View to Visualize the Data

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];
```

Figure 5.2.3 Data Visualization - Table View

2.3 My Account & Logout

By clicking “My Account” button, user can check out his history deal records. And if the user wants to logout the system, he can directly click logout.

Core Code: HTTP Request / Visualize Global Value (Similar with above)

5.3 Bid Module

Entering into the bid view, the item information will show out at the middle part of the view, and also the user can check the online price through Amazon and Ebay. After making sure the item information and checking the online price, he can begin to bid.

The rule of bid: for each user, they just have three times to bid a product, only if one price is higher than the lowest price provided by business, the customer bids successfully. And the page will automatically transferred to the QR Code view page. If the user fails to bid three times, then the username will be blocked for the item 24 hours.

Core Code:

1. Generating QR Code whenever a deal is successfully made

```
UIGraphicsBeginImageContext(CGSizeMake(imgSize,imgSize));

CGContextRef ctx = UIGraphicsGetCurrentContext();

// init all pixels to 'light' colour
CGRect bounds = CGRectMake(0,0,imgSize,imgSize);
CGContextSetFillColorWithColor(ctx,iLightColour.CGColor);
CGContextFillRect(ctx,bounds);

// set any 'dark' colour pixels
{
    int x,y;

    CGContextSetFillColorWithColor(ctx,iDarkColour.CGColor);
    for ( y=0 ; y<logQRSize ; y++ )
        for ( x=0 ; x<logQRSize ; x++ )
            if ( qr->data[(y*logQRSize)+x] & 1 )
                CGContextFillRect(ctx,CGRectMake((iQuietZone+x)*scale,(iQuietZone+y)*scale,scale,scale));
}

// generate the UIImage
CGImageRef imgRef = CGBitmapContextCreateImage(ctx);
ret = [UIImage imageWithCGImage:imgRef];
CGImageRelease(imgRef);

UIGraphicsEndImageContext();
```

Figure 5.3.1 QR Code

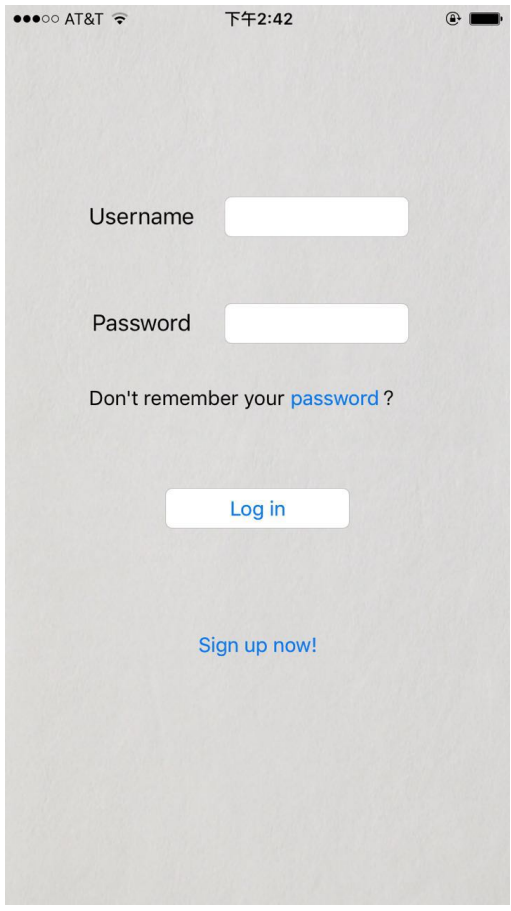
2. Automatically Change View Page when the timer is running over

```
[NSTimer scheduledTimerWithTimeInterval:1.0 target:self
                                selector:@selector(automaticTransfer) userInfo:nil repeats:NO];
```

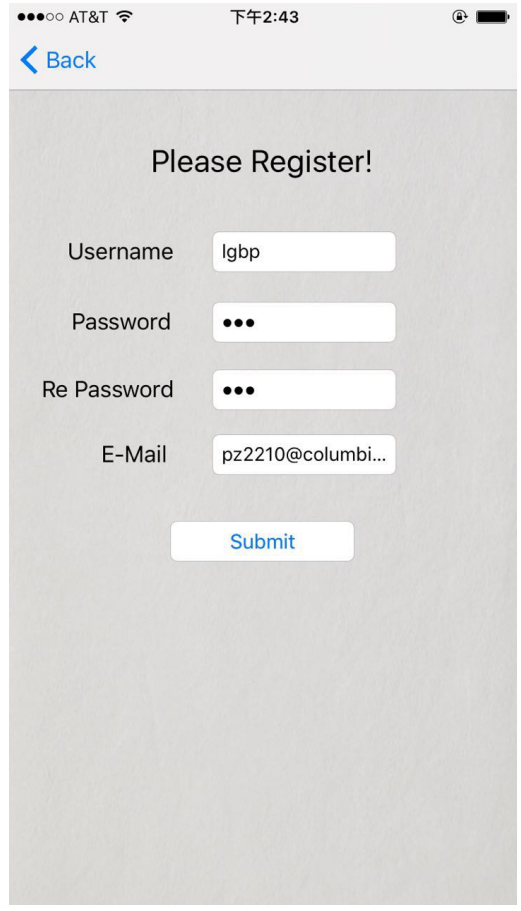
Figure 5.3.2 Timer

3. HTTP request and count three times timer (code is similar with the former part)

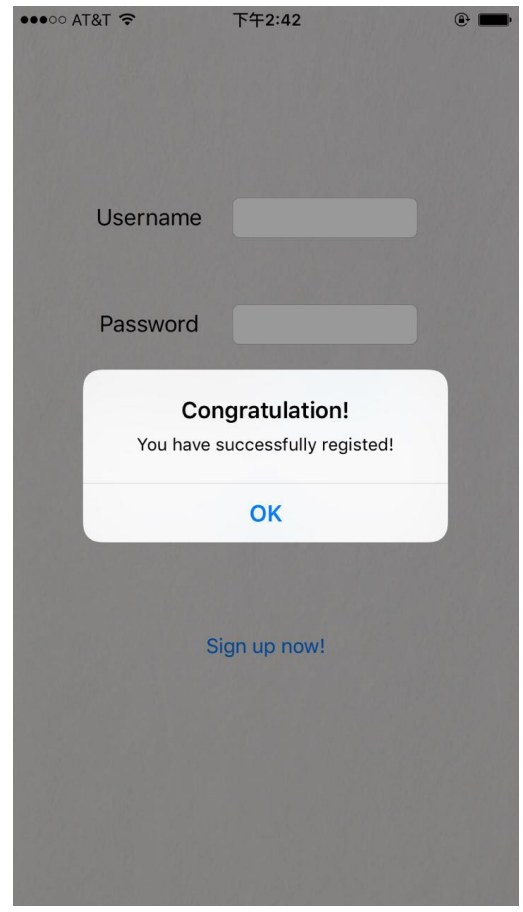
6. Demo Screen



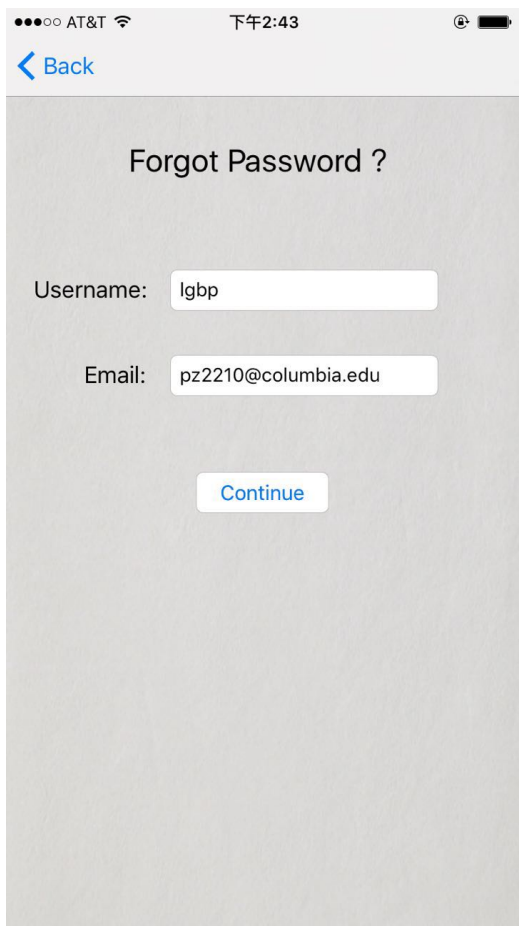
Login



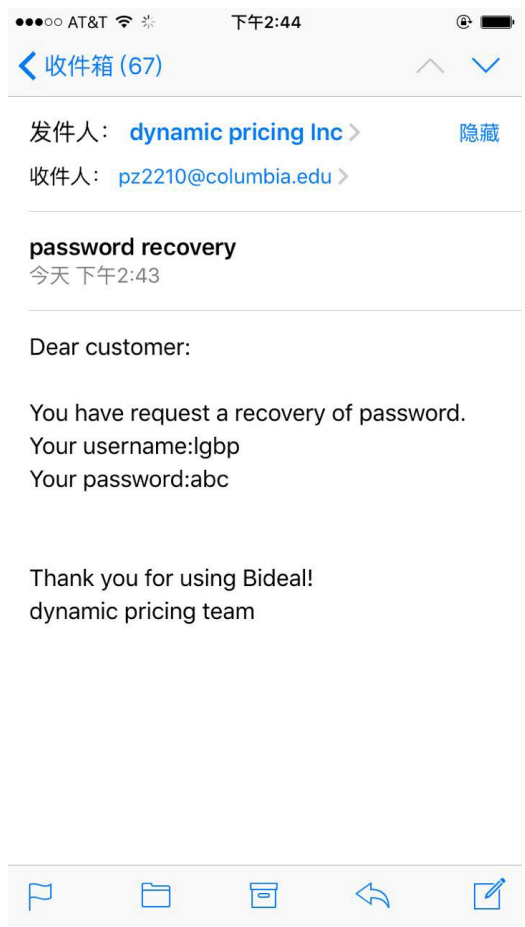
Register



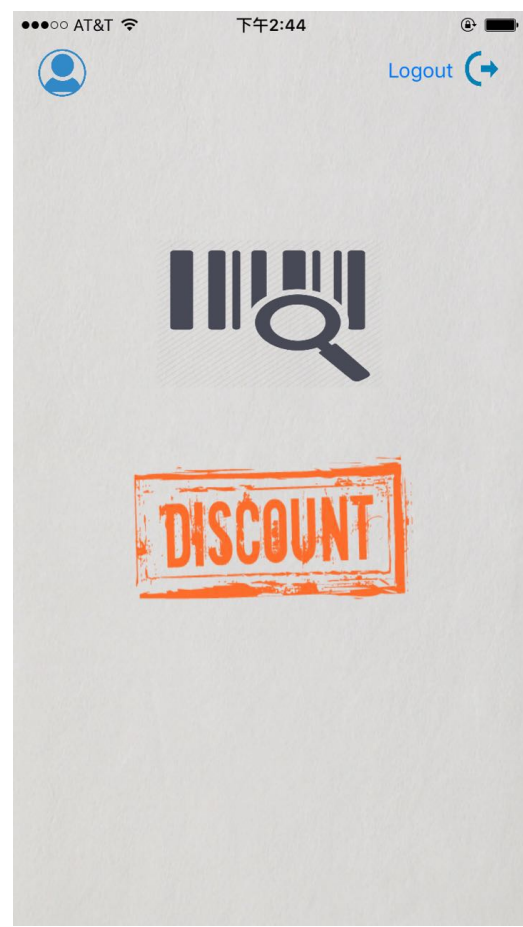
Register - Successfully



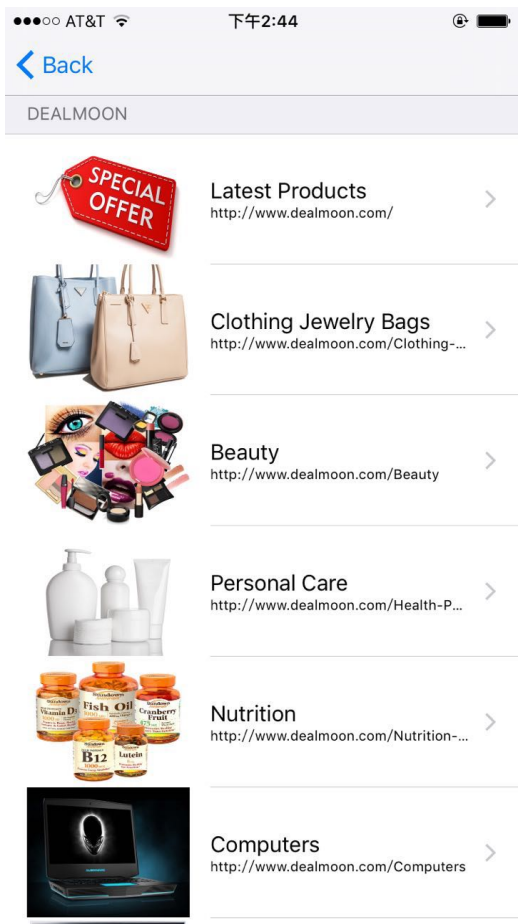
Forgot Password



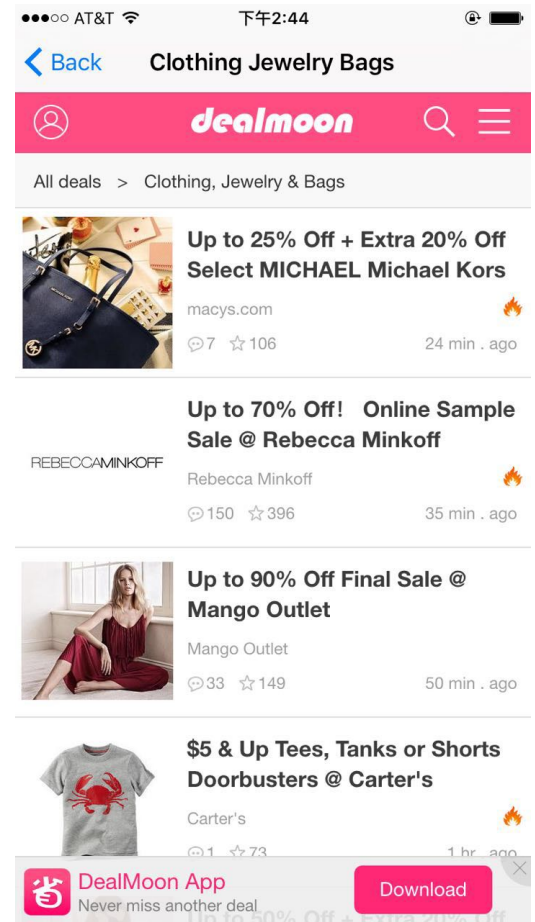
Receive Password



Main Page



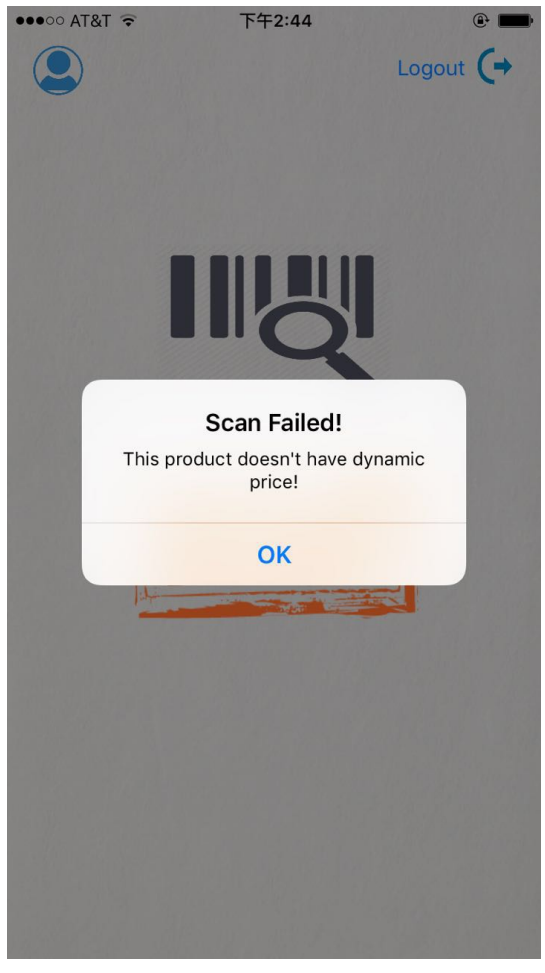
Discount Activity



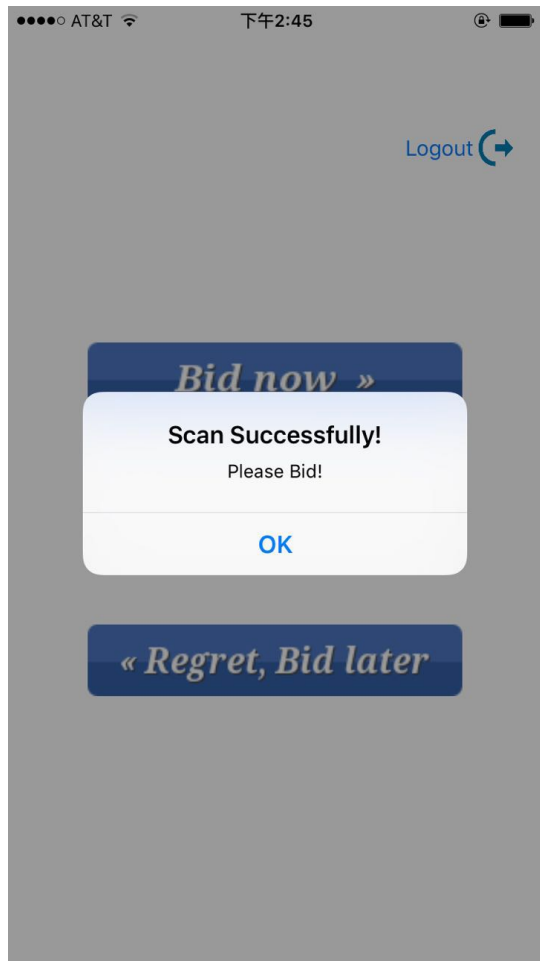
Discount - Details

Scan - Specific Business

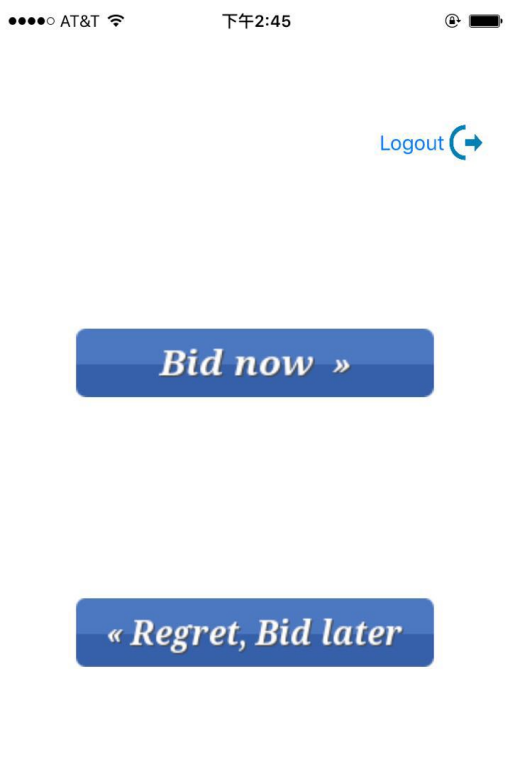
Scan Item



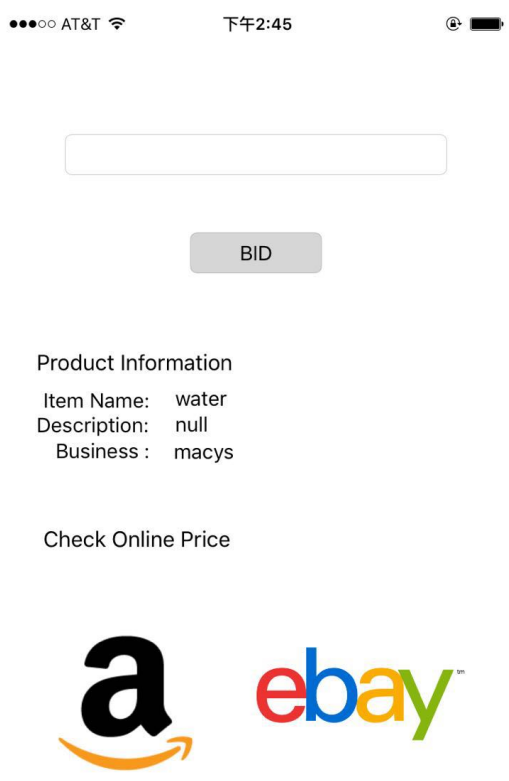
Scan Failed



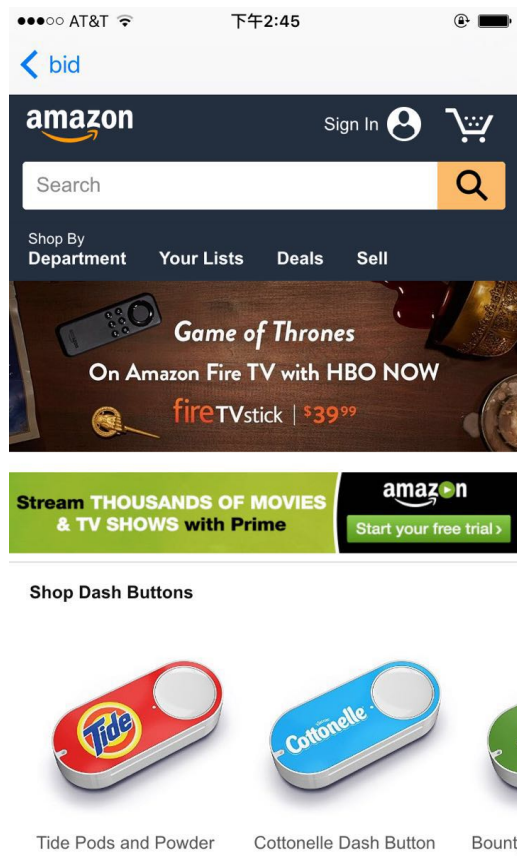
Scan Successfully



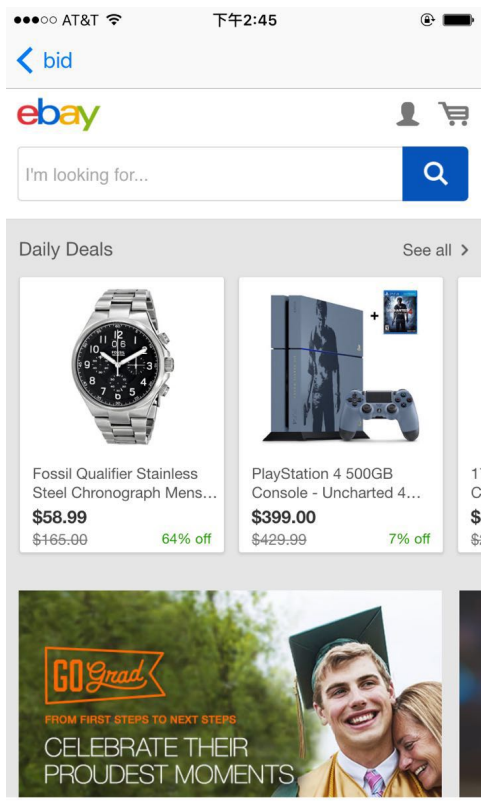
Decide Bid or Later



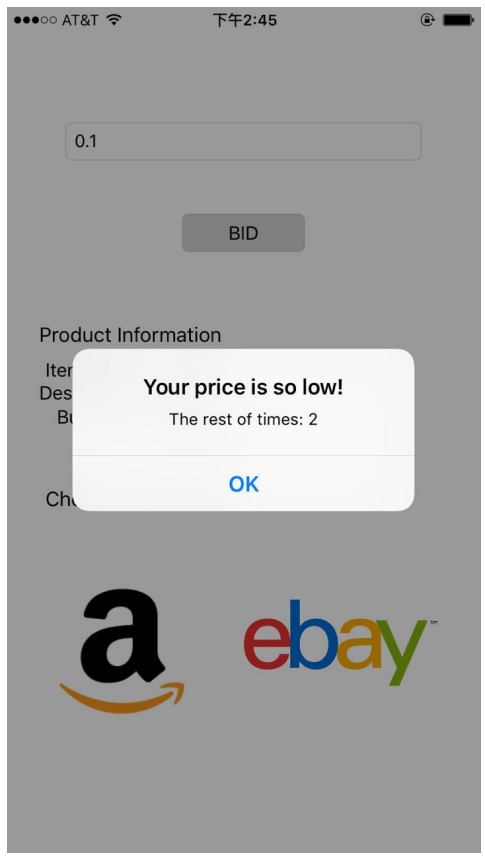
Bid Page



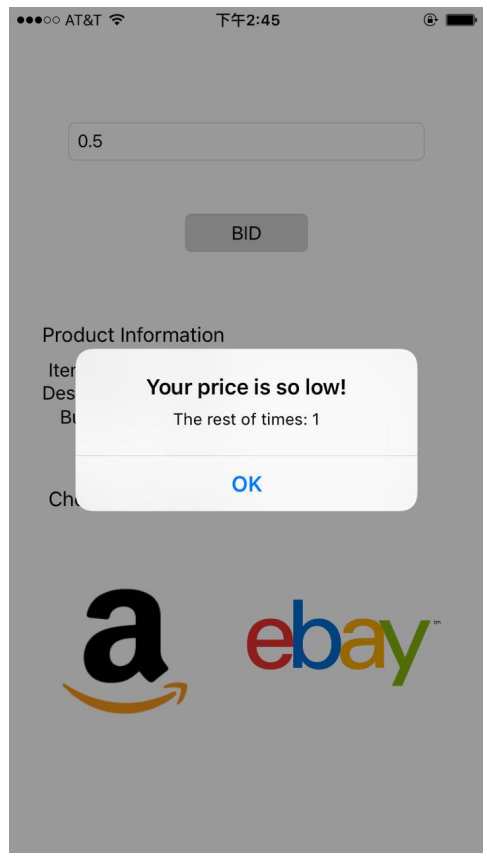
Amazon Online Price



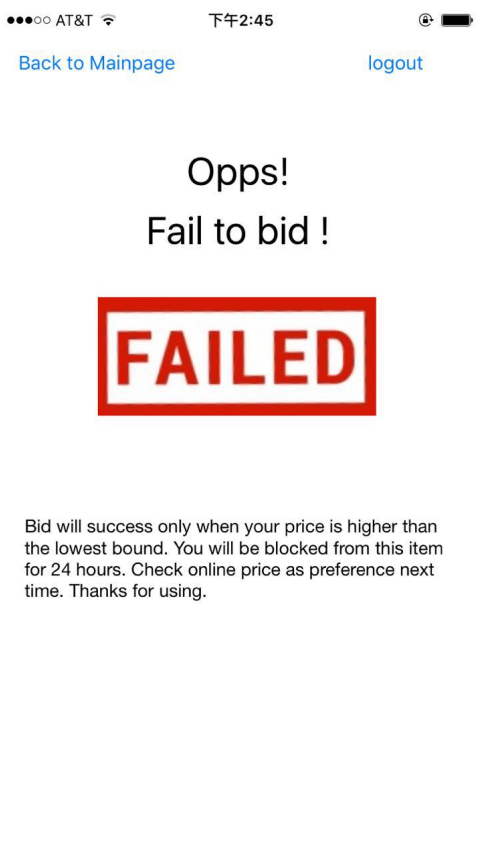
Ebay Online Price



Bid Price - Low

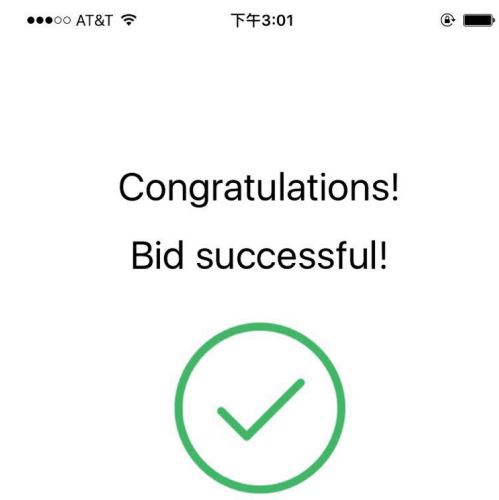


Bid Price - Low 2



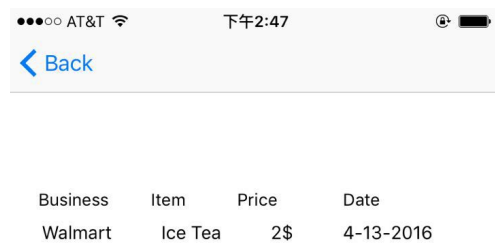
Bid Failed

Bid Another Item



Bid Successfully

Generate Coupon Barcode



Bid History

7. YouTuBe URL

https://youtu.be/i0RiBmTUwQc