Discount Shuttle

Project Overview:
Generally speaking,Discount Shuttle is designed to collect discount information for customer they are interested in, and to build a platform for seller to attract customer.

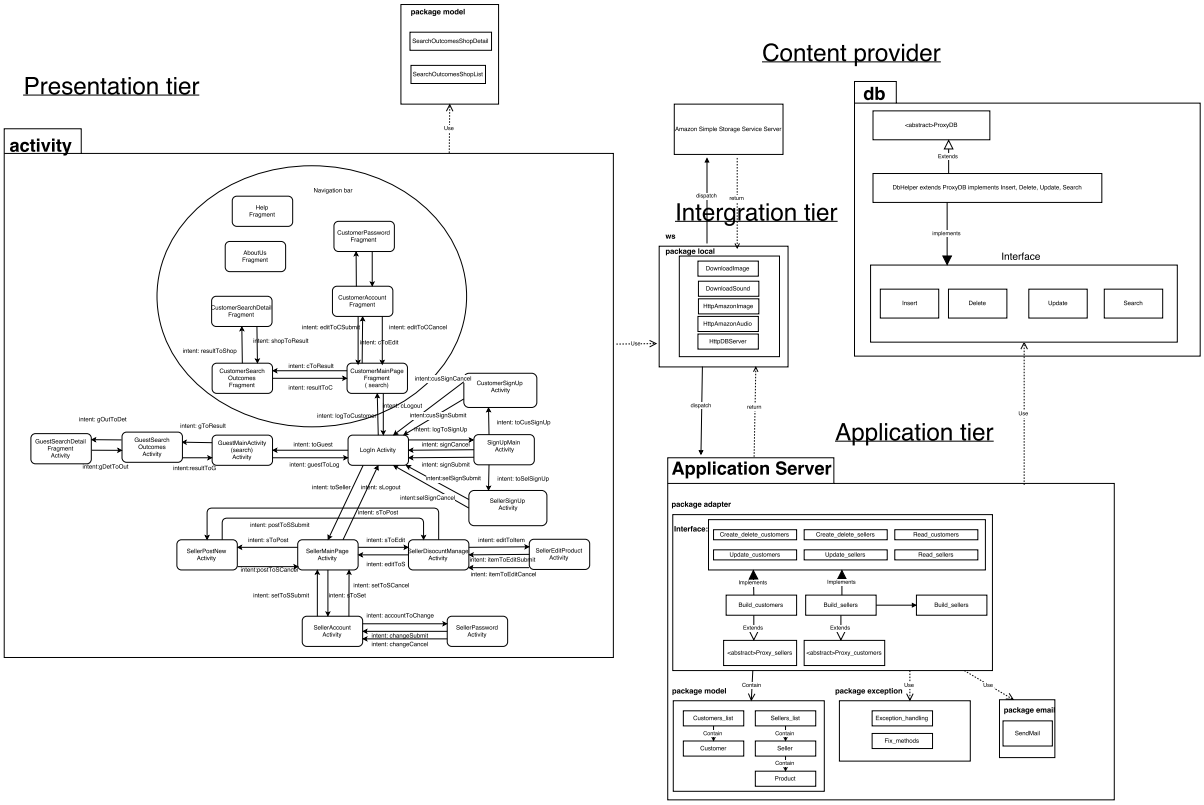
As a seller, one can simply post the discount information combined with their shop's location, those data will be send and stored in our server, which is a tomcat server running by a laptop.

On the other hand, Discount shuttle allows customer to find their interested discount information close to them.They can get details about the product, the shop, even the way lead to the shop.

This idea was generated one day I hang out for dinner with my friends in south side and find a supermarket named ALDI accidentally. I find most of the items there are in discount and nearly half price of those in Giant Eagle where I used to shop.

As a student without income, I consider that those information especially discount information can be very important to people like me. they can save lots of money and buy high quality products on the same time.

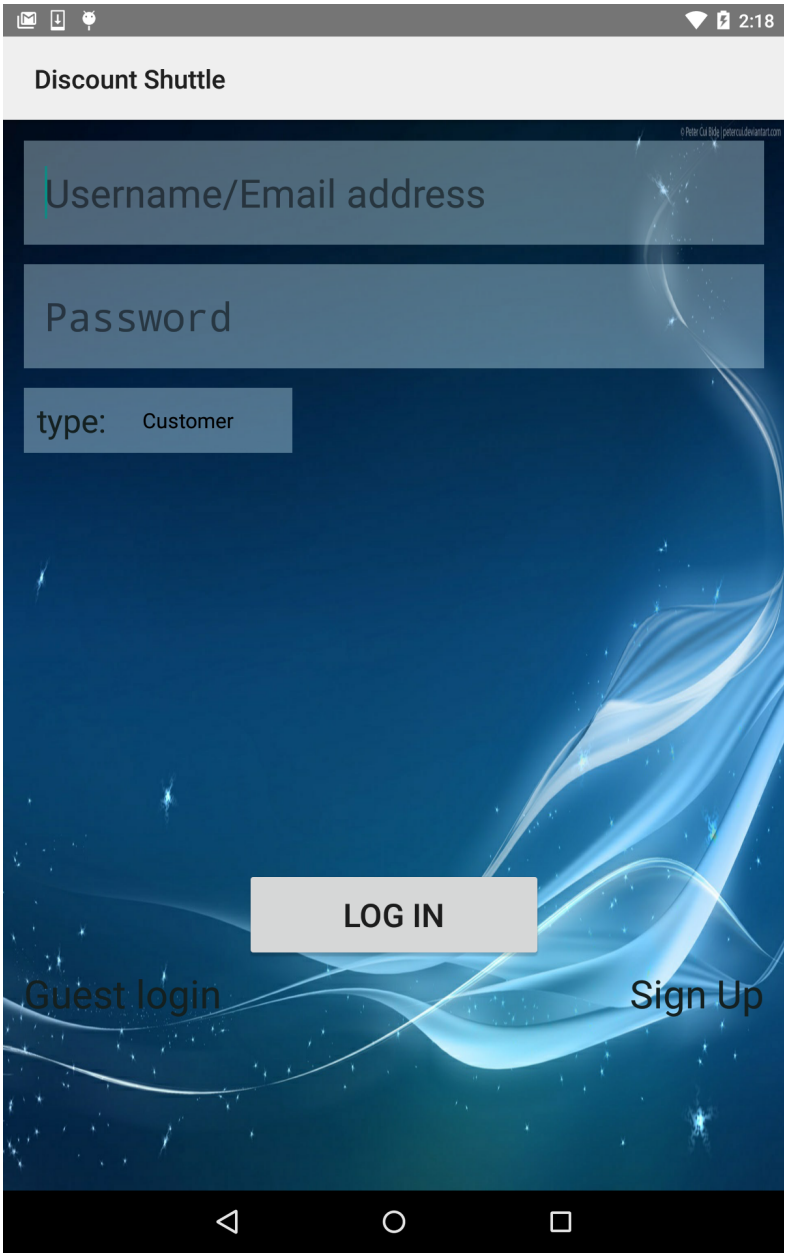
General class diagram:



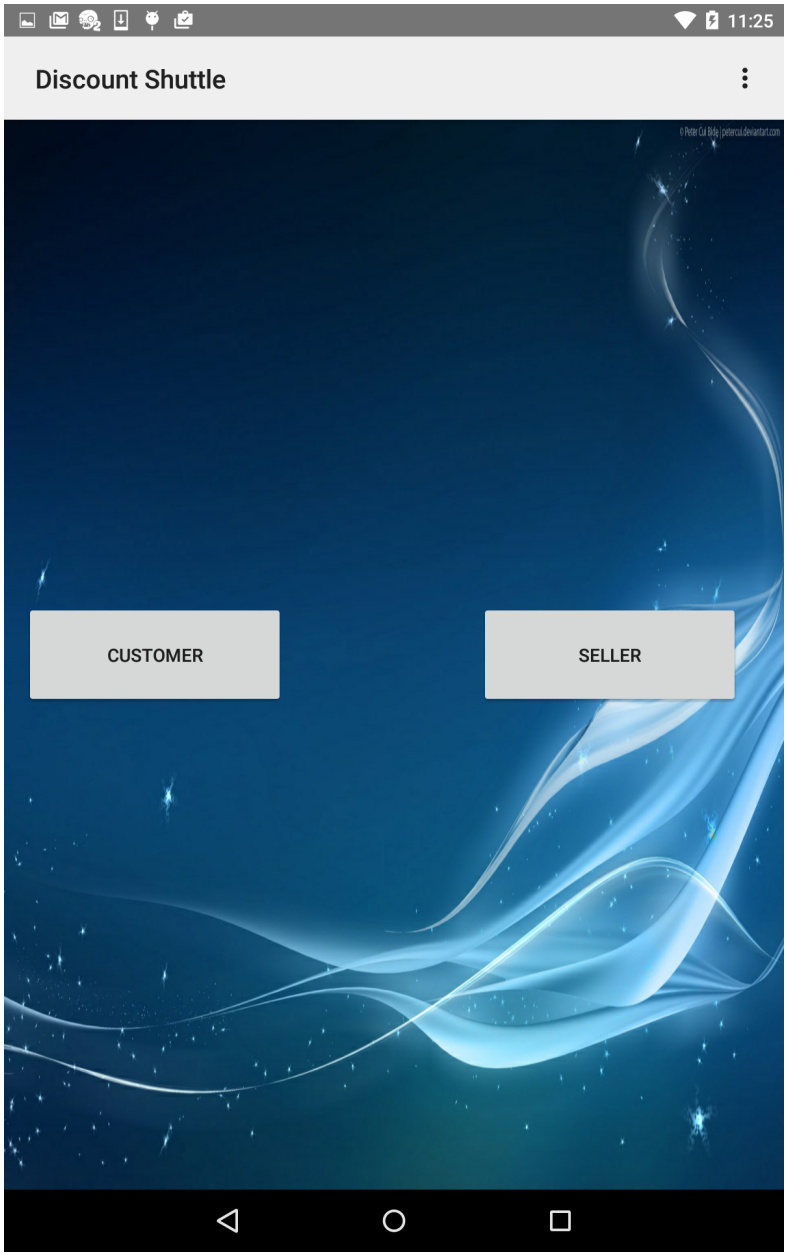
Step 1: Designing Screens

Resolution(768*1280 xhdpi) and Resolution(1200*1920 xhdpi):

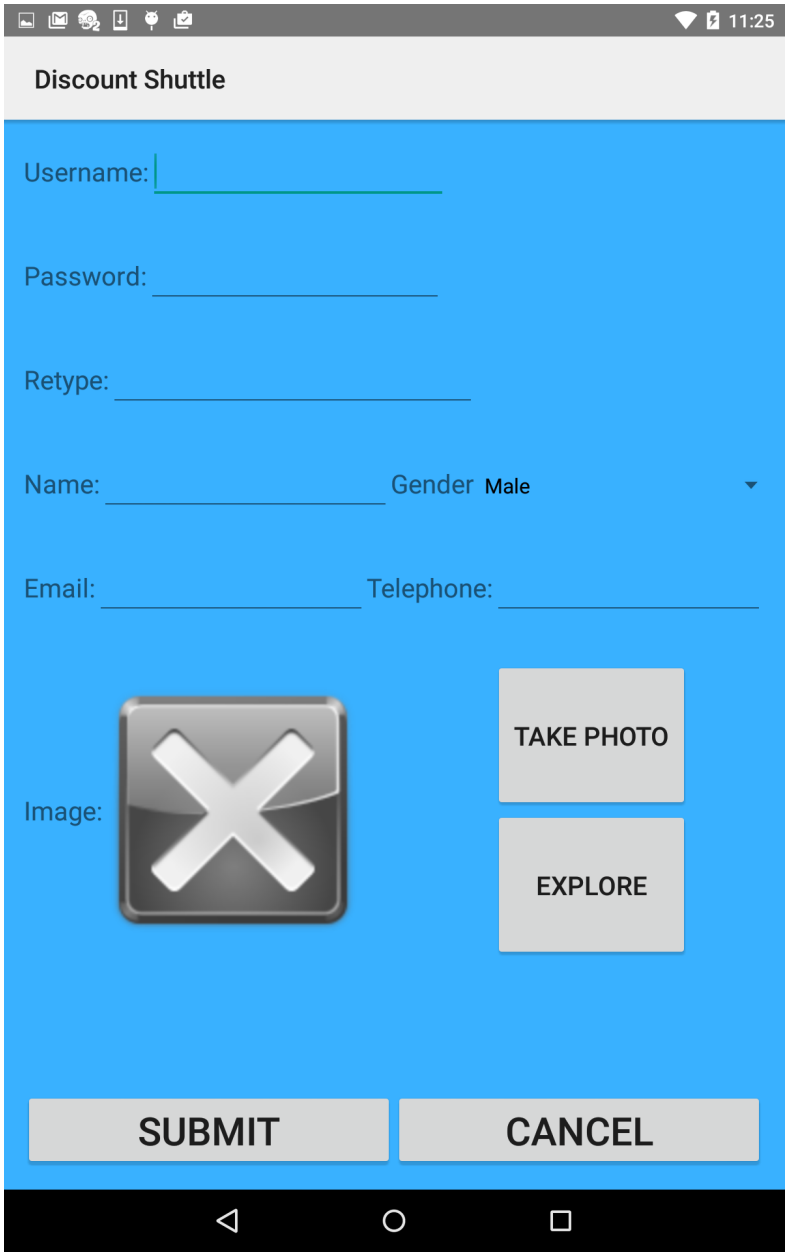
1.Main page:



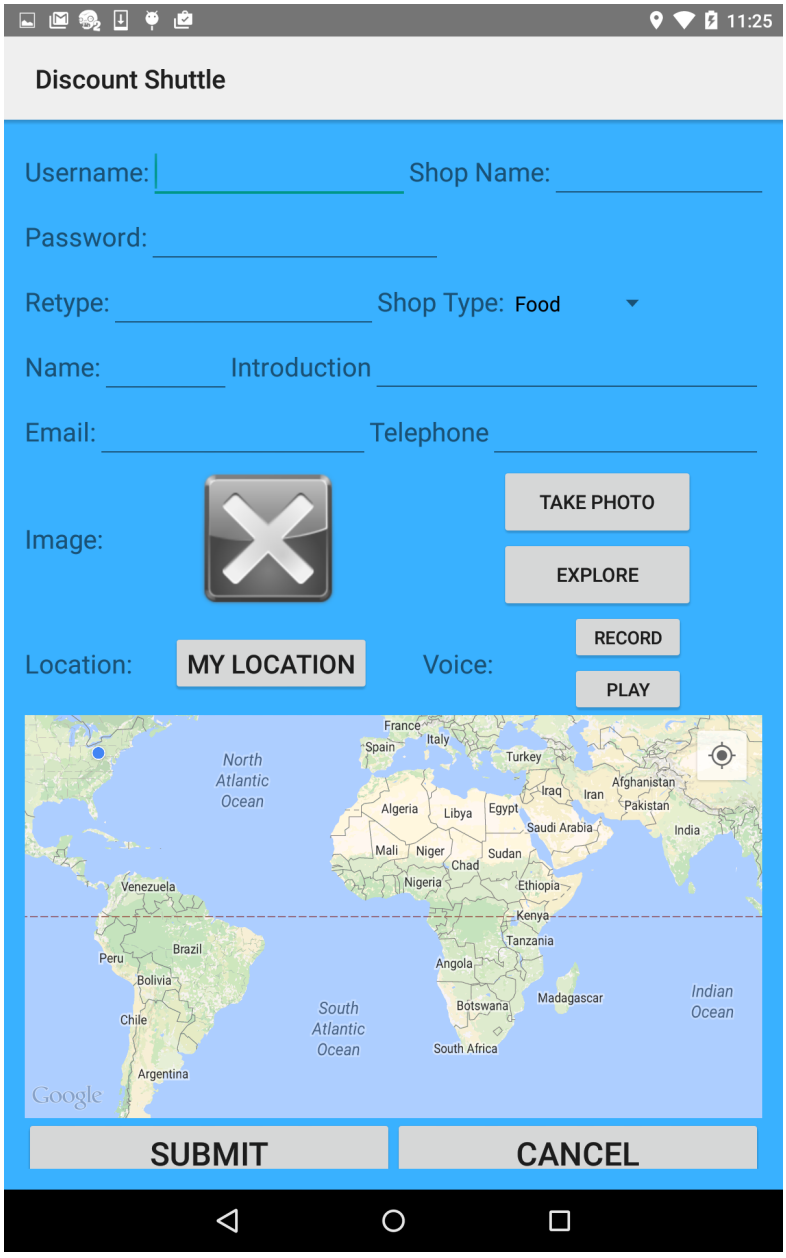
2.Main sign up page:



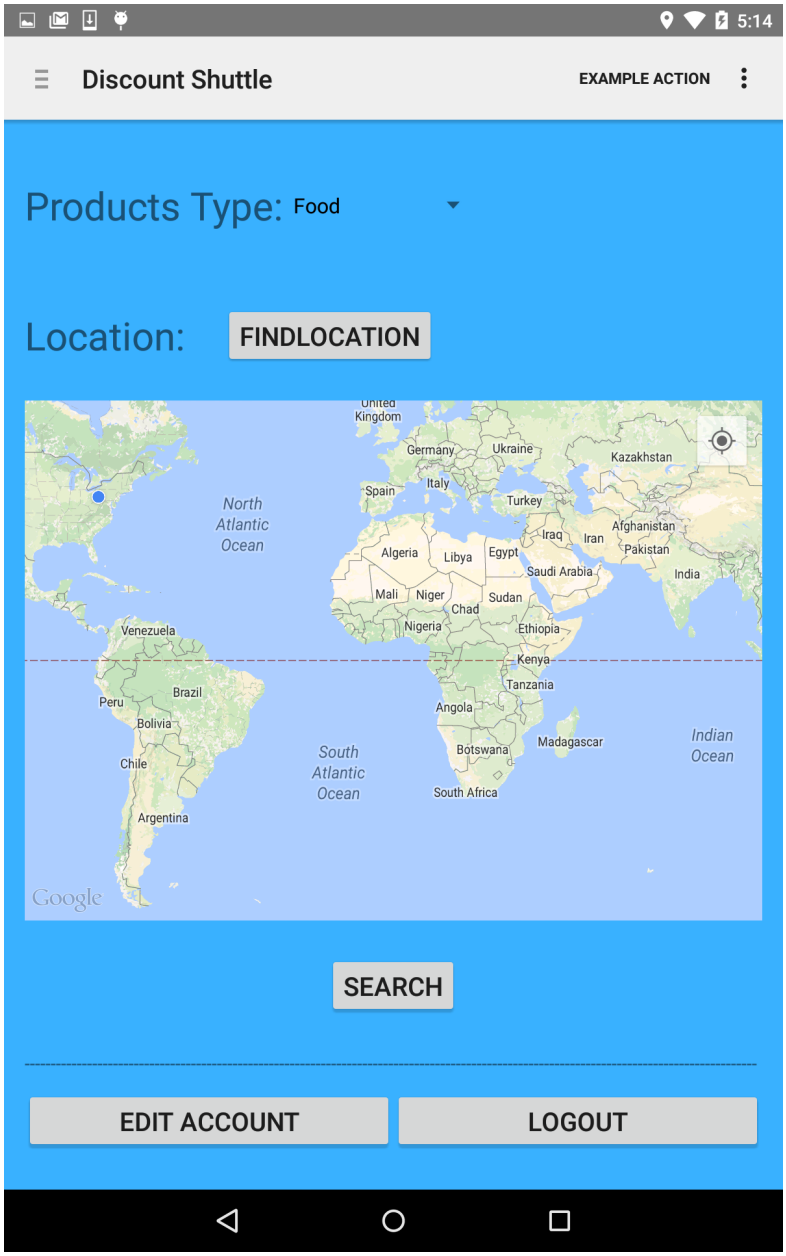
3.Customer sign up page:



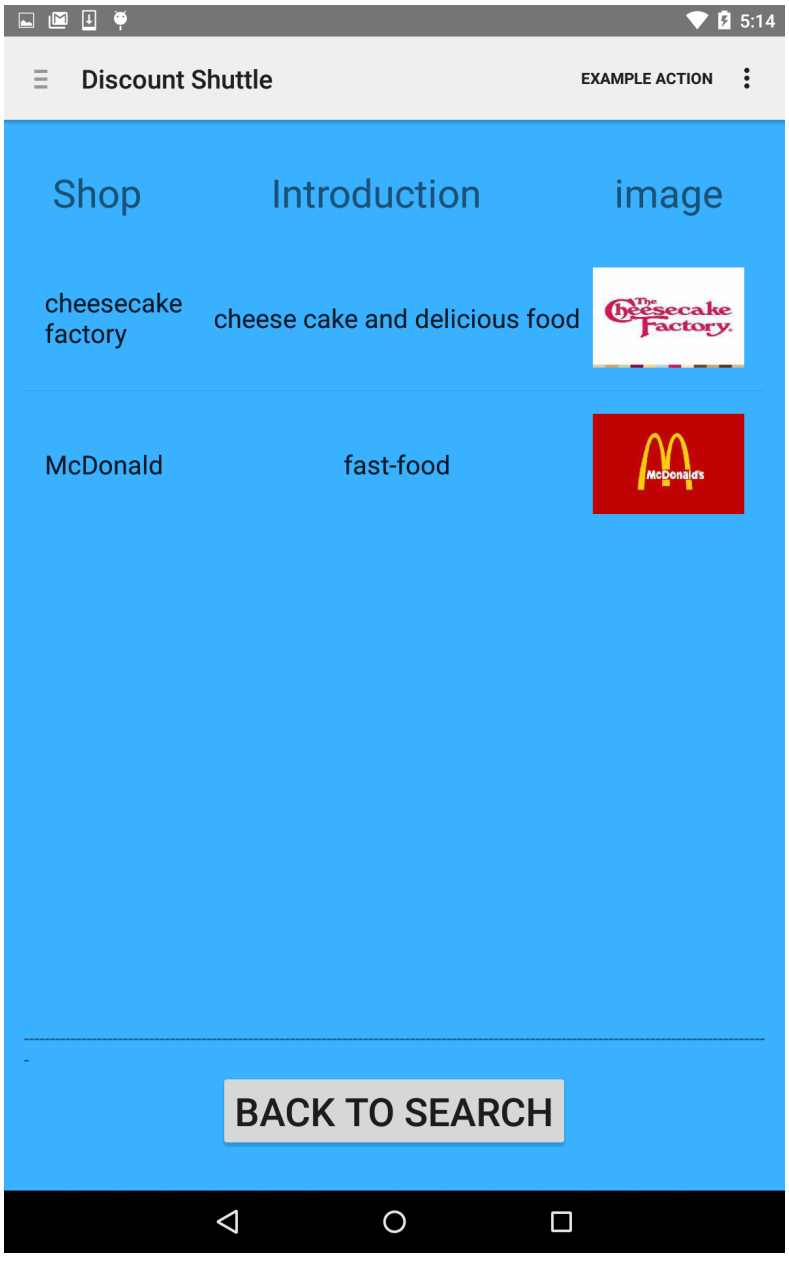
4.Seller sign up page:



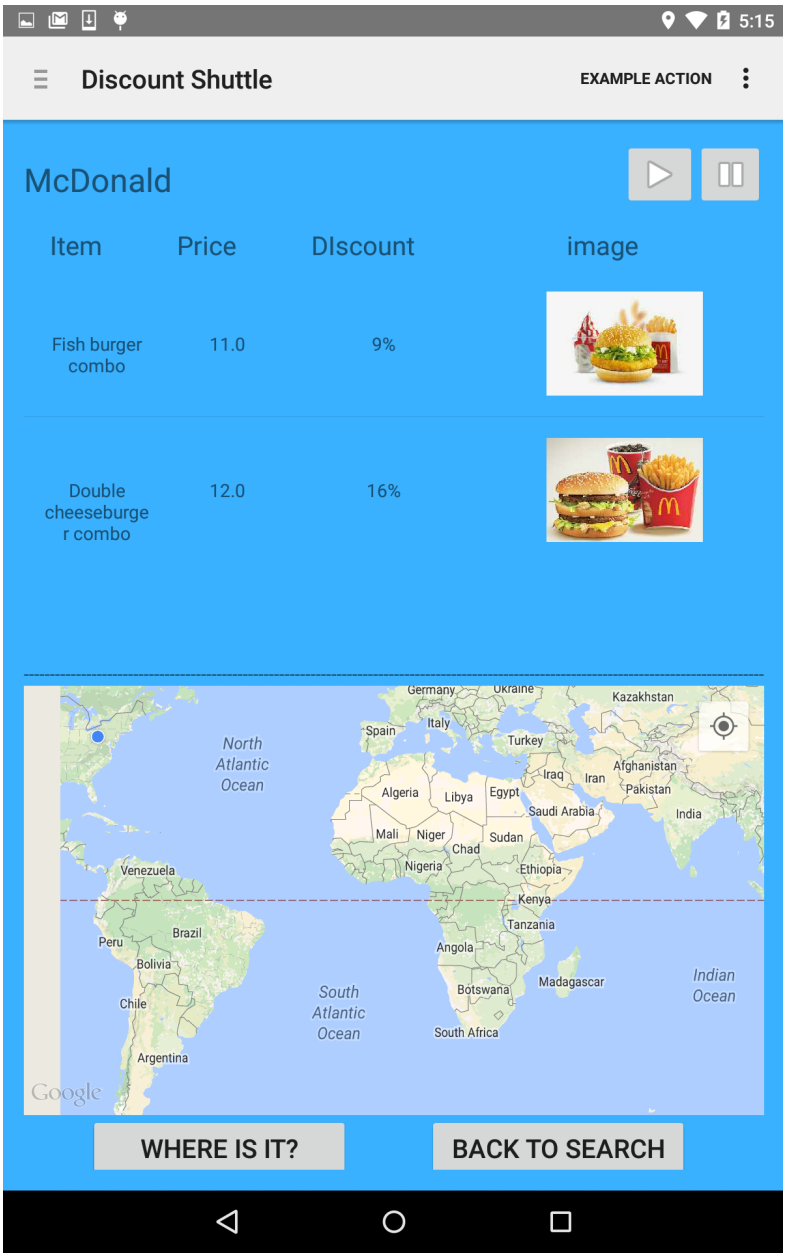
5.Customer main page:



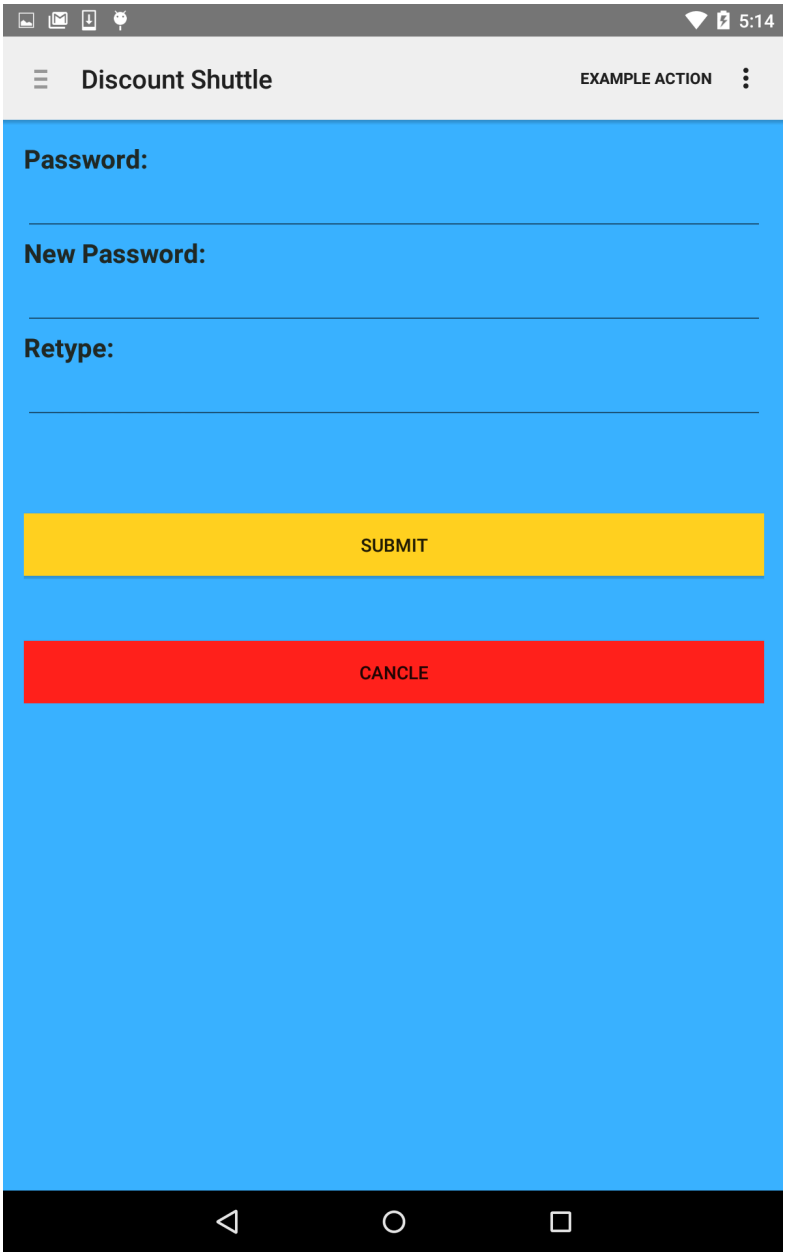
6.Customer search outcomes page:(guest)



7.Customer search detail page: (guest)

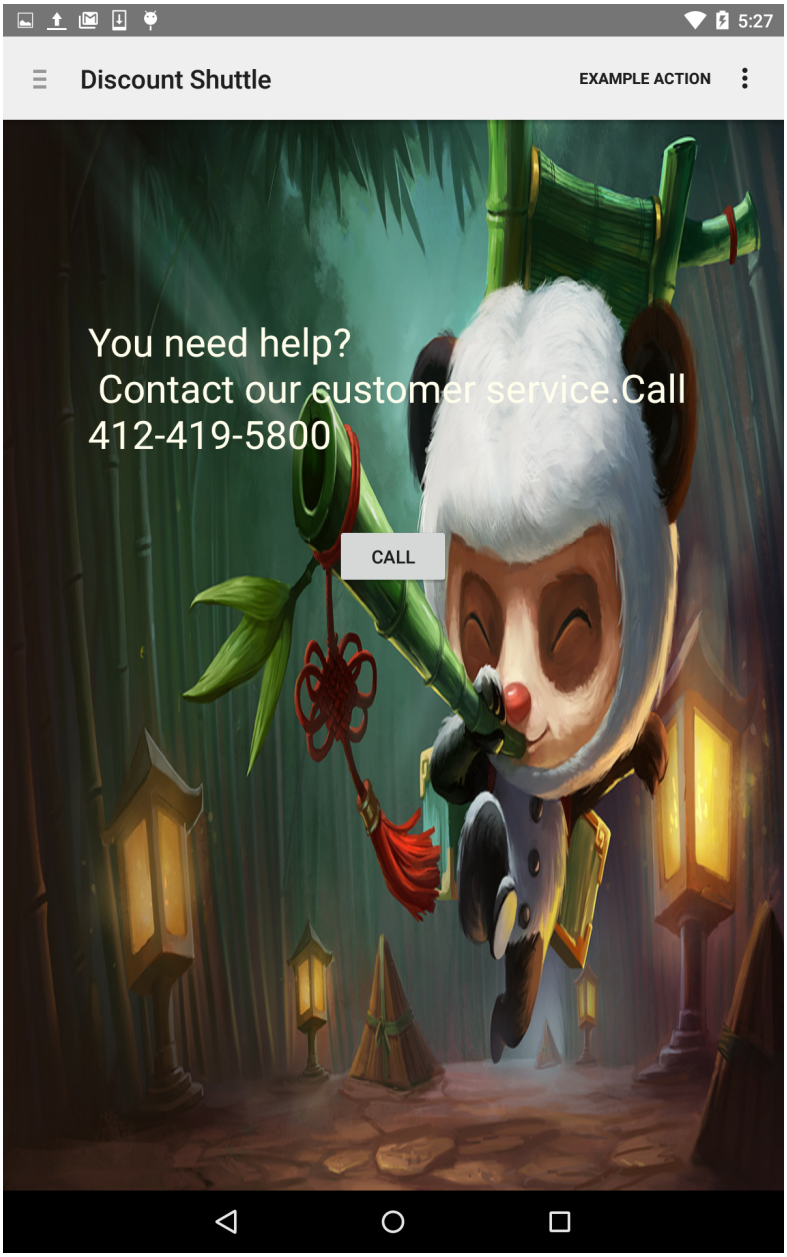


8.Customer change password page: (seller)

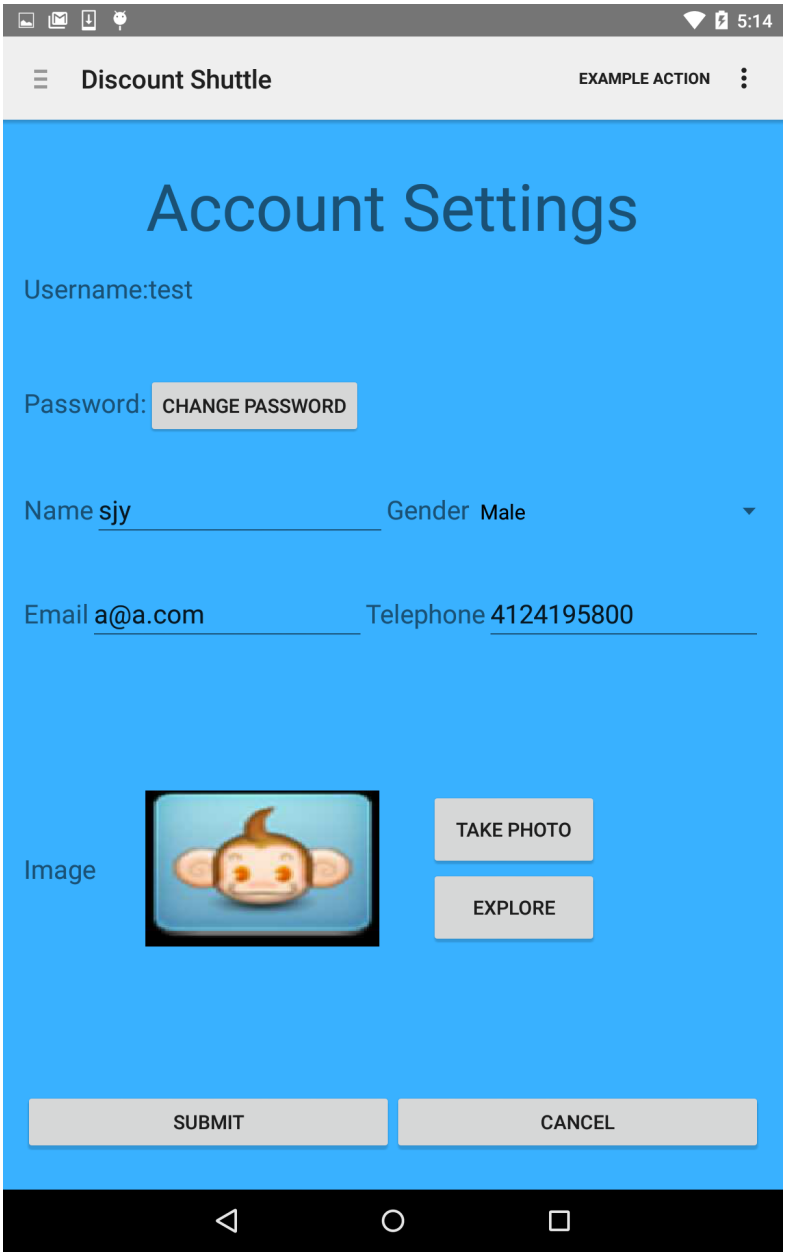


9.Customer About us page:

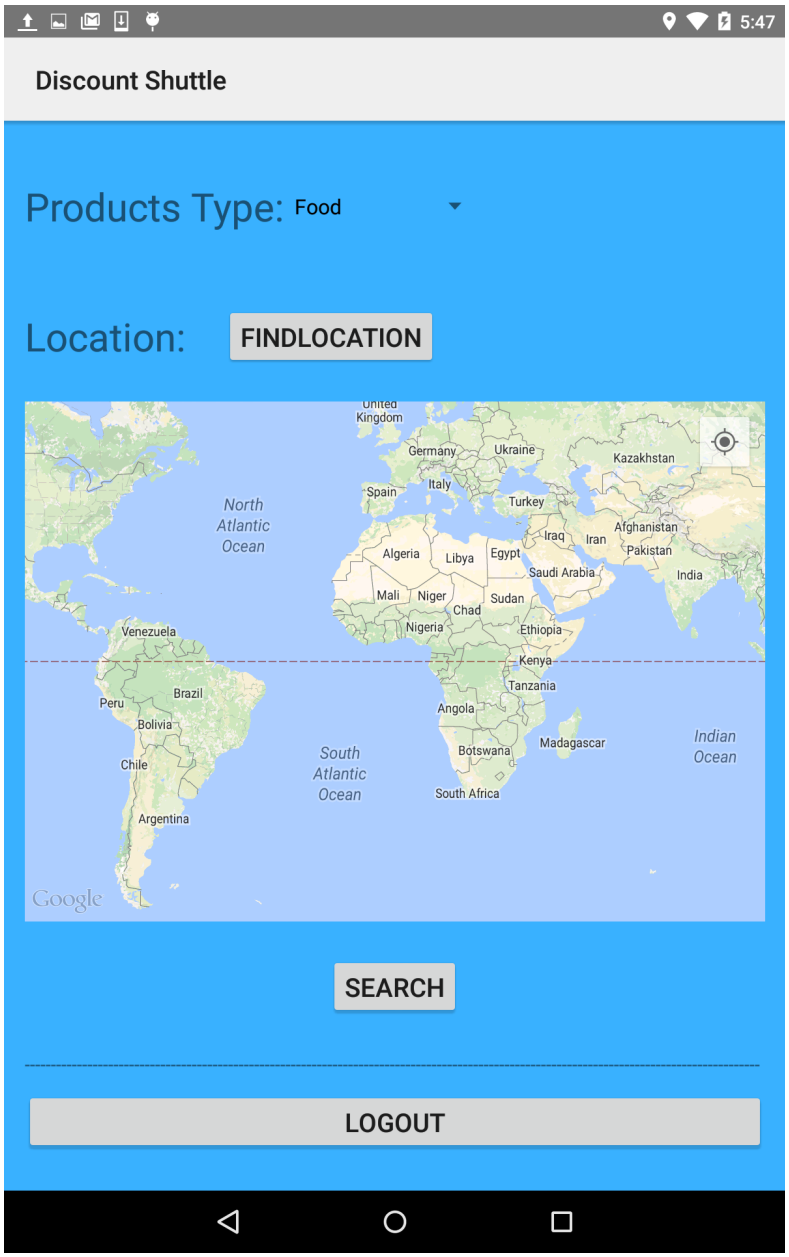
10.Customer help page:


5:27
Discount Shuttle
EXAMPLE ACTION
You need help?
Contact our customer service.Call
412-419-5800
CALL

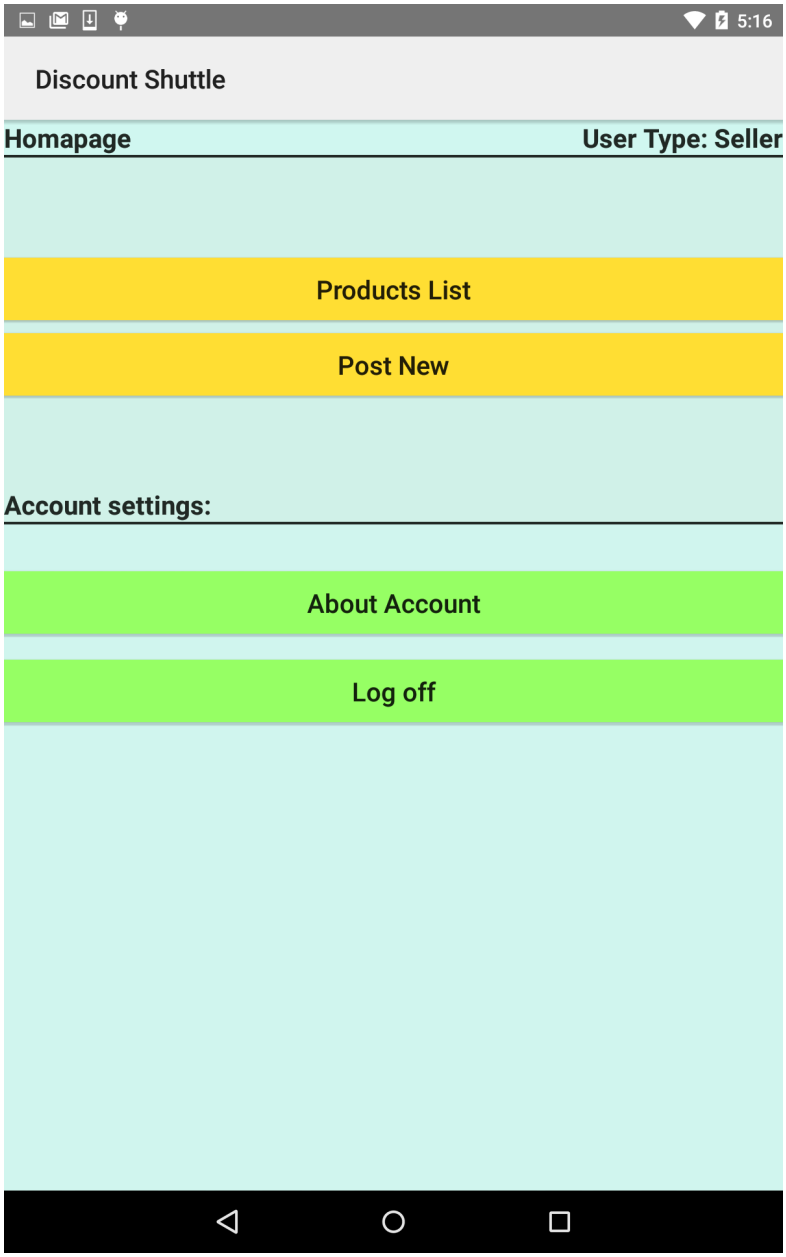
11.Customer Account page:



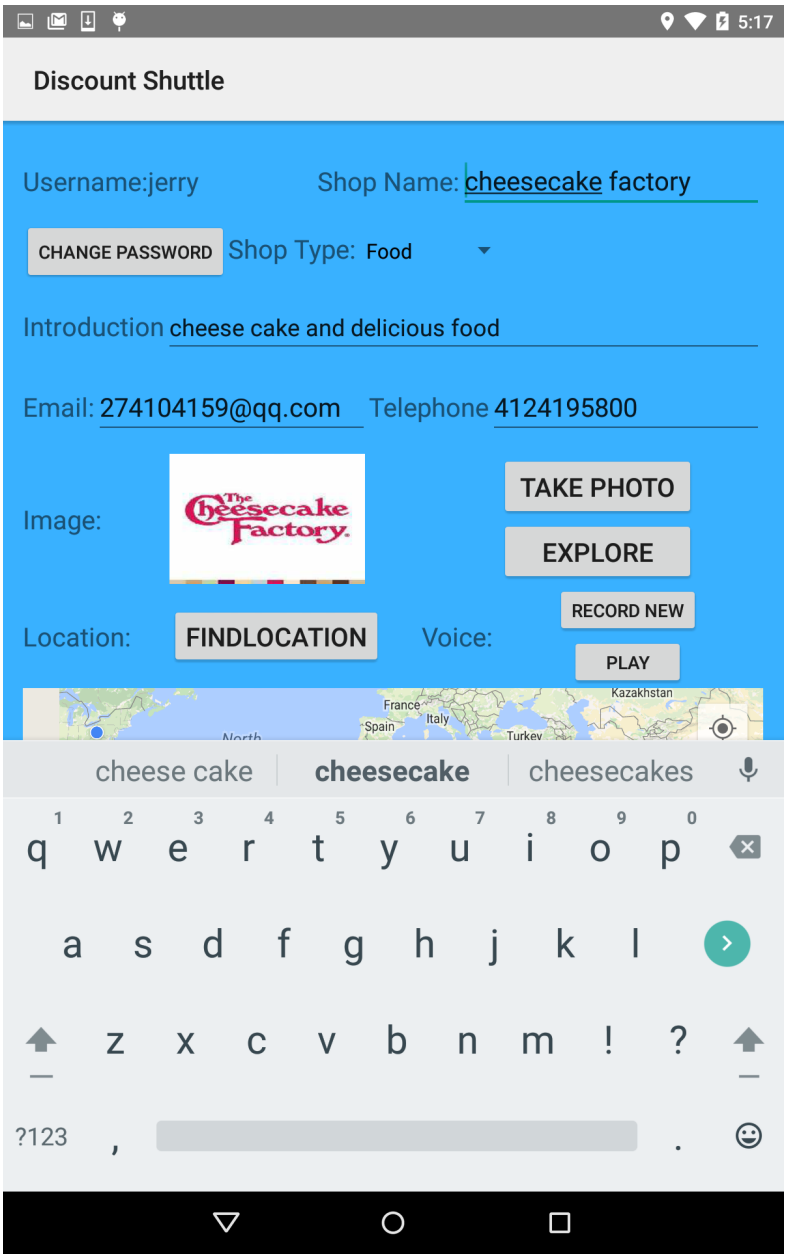
12.Guest Main page:



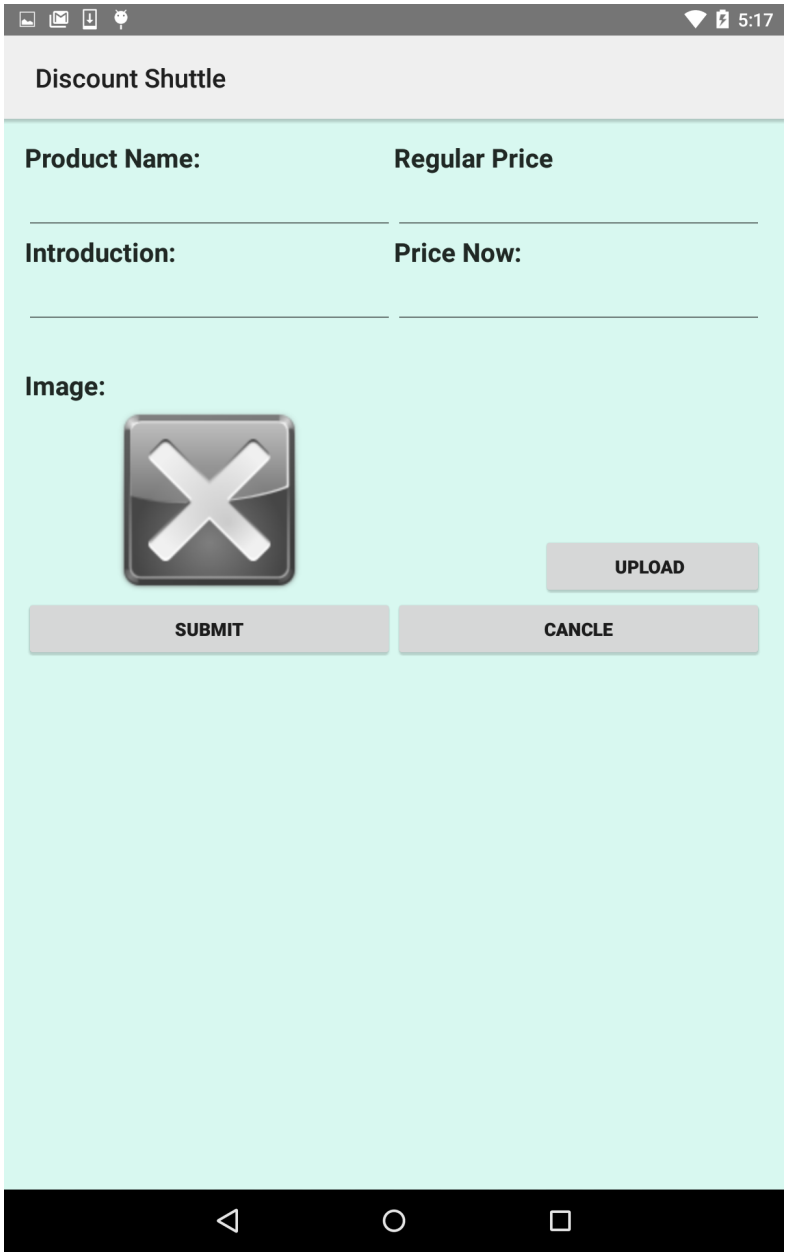
13.Seller Main page:



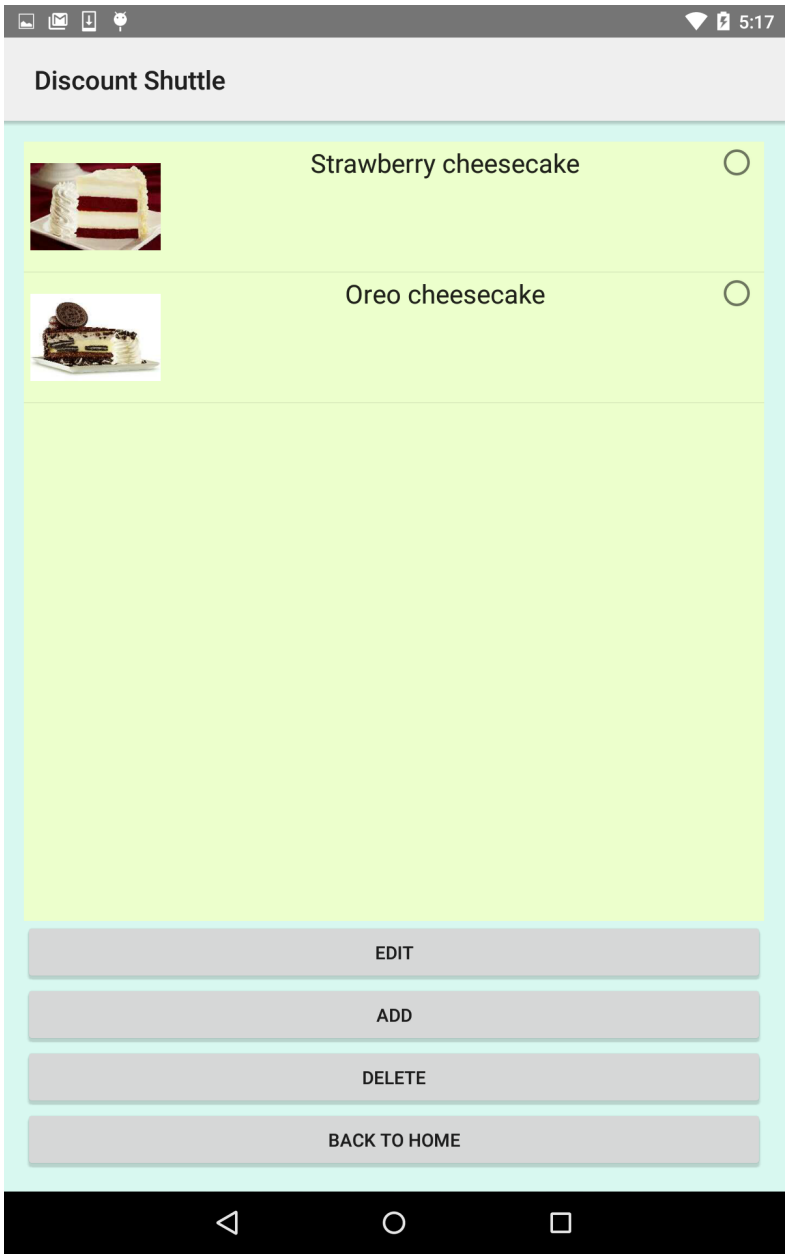
14.Seller Account page:



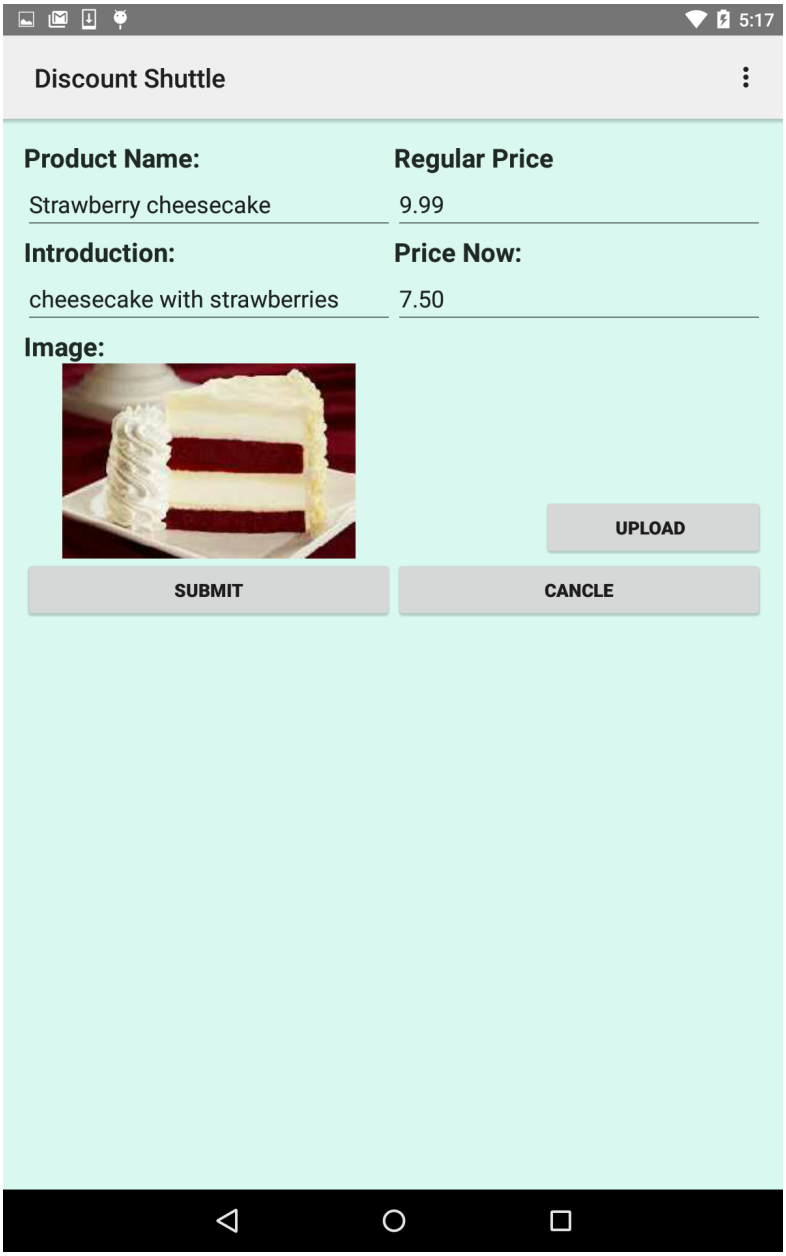
15.Seller Post new page:



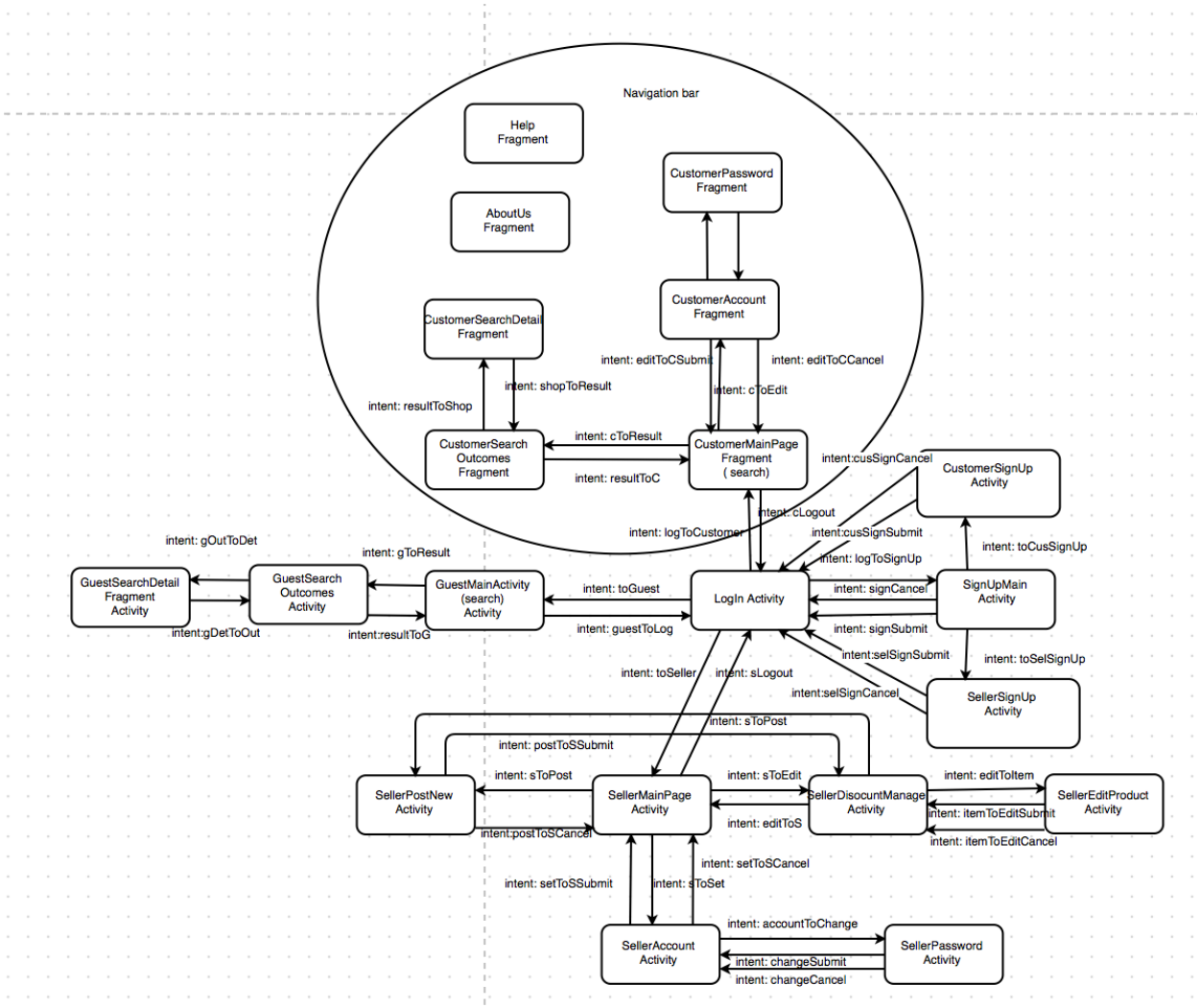
16.Seller Discount Manage page:



17.Seller Discount Manage page:



Step 2: Presentation Tier



Step 3. Designing Content Provider (For Storage)

1.Requirements

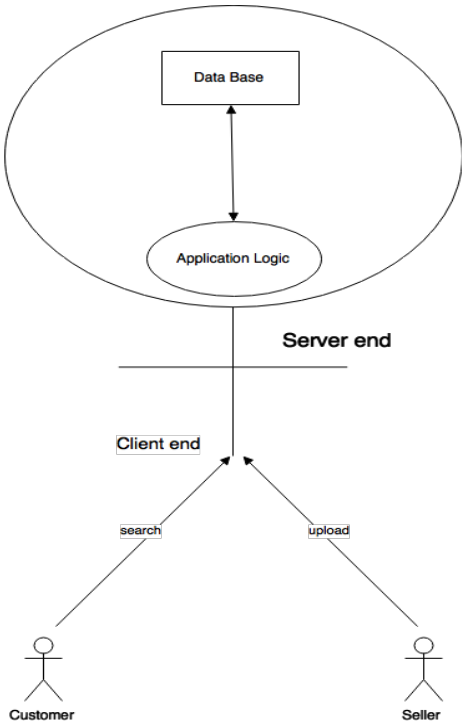


Fig. Discount Shuttle interaction model

Discount Shuttle is a kind of application that dealing information demand and supply. In our model, we have Seller and Customer. Their personal information was stored in our server when they registered. In this way, when they log in, they will be authenticated based on the database records.

Both Seller and Customer may modify their personal information at the application client end. The modified information will be updated in the database at server end.

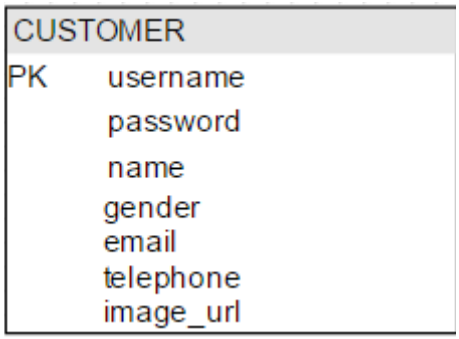
A Seller can upload a new discount information. The information primarily is about a Product. After the seller input required product information, the data will be sent to server to build and record the data information.

A customer can use the mobile device to search for their interested products by product type and location. The relative information will be responded from the remote server to the customer.

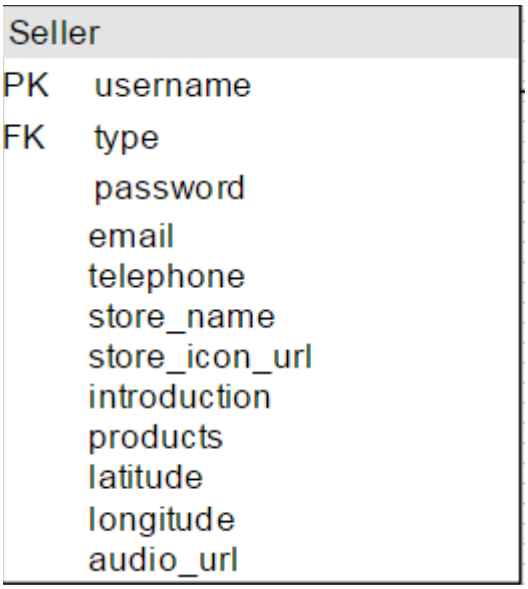
2.Logic data model

Based on the requirement, there are two actors and one product assigned to a certain product type. As a result, there should be 4 tables for customer, seller, product and product_type.

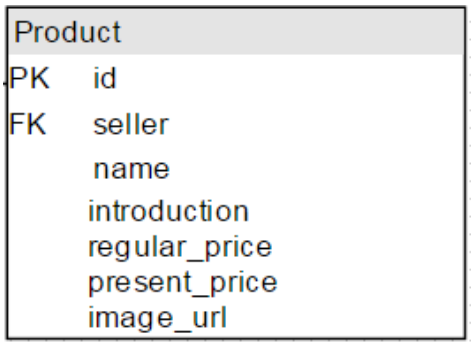
A user register an account as a customer should provide a unique username, so username serves as a primary key in the customer table. For account protection, a user of course should have a password. This password shall be recorded in the database so that a user could log in the account with different mobile device. Email is also a required field for customer, since it is the most common way to contact a user. Other basic information such as real name, gender, profile image and phone number are optional fields in the table.



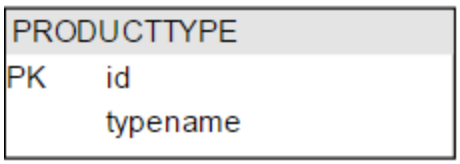
For the same reason, a seller will have a unique username, required password and email, optional real name, store name, store image url and introductions.



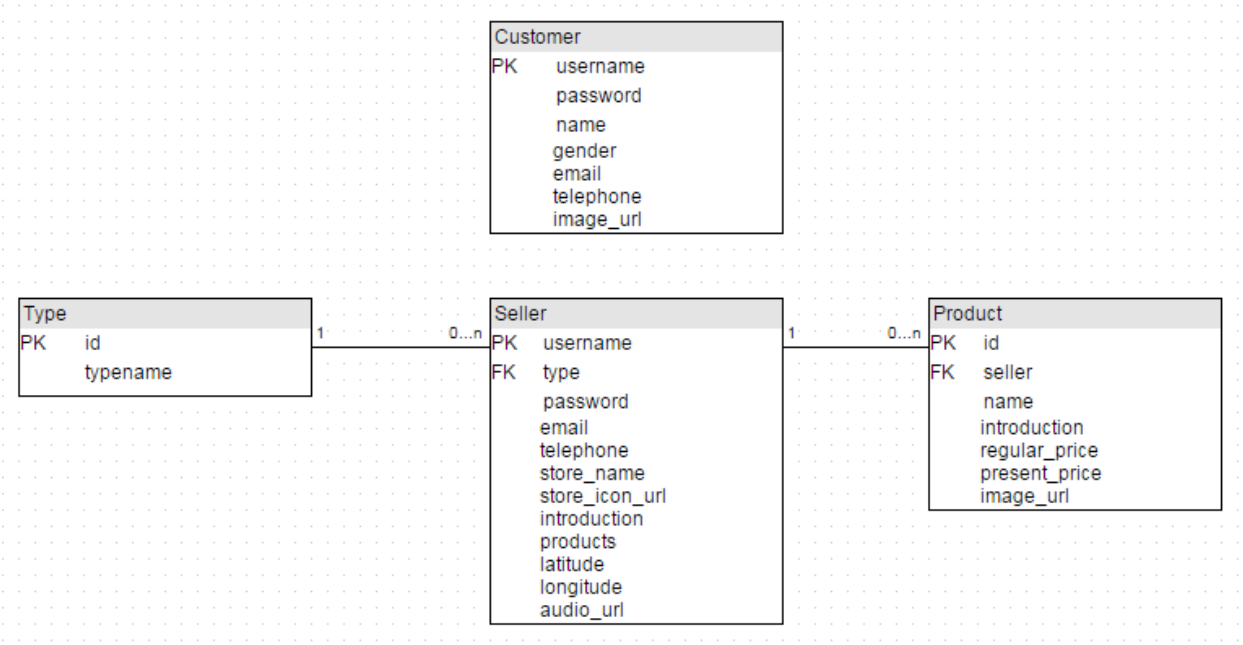
Product shall has its own table because a seller may have several products. All basic attributes may be duplicate with one another. As a consequence, an integer id shall be the primary key. One foreign key is seller's primary key to describe their relationship. Another foreign key is the product type id as a primary key in the product_type table, because a customer may search a product base on the product type and a seller may declare a new product type and add it in the product_type table.



As explained above, a product type table is needed for performance.

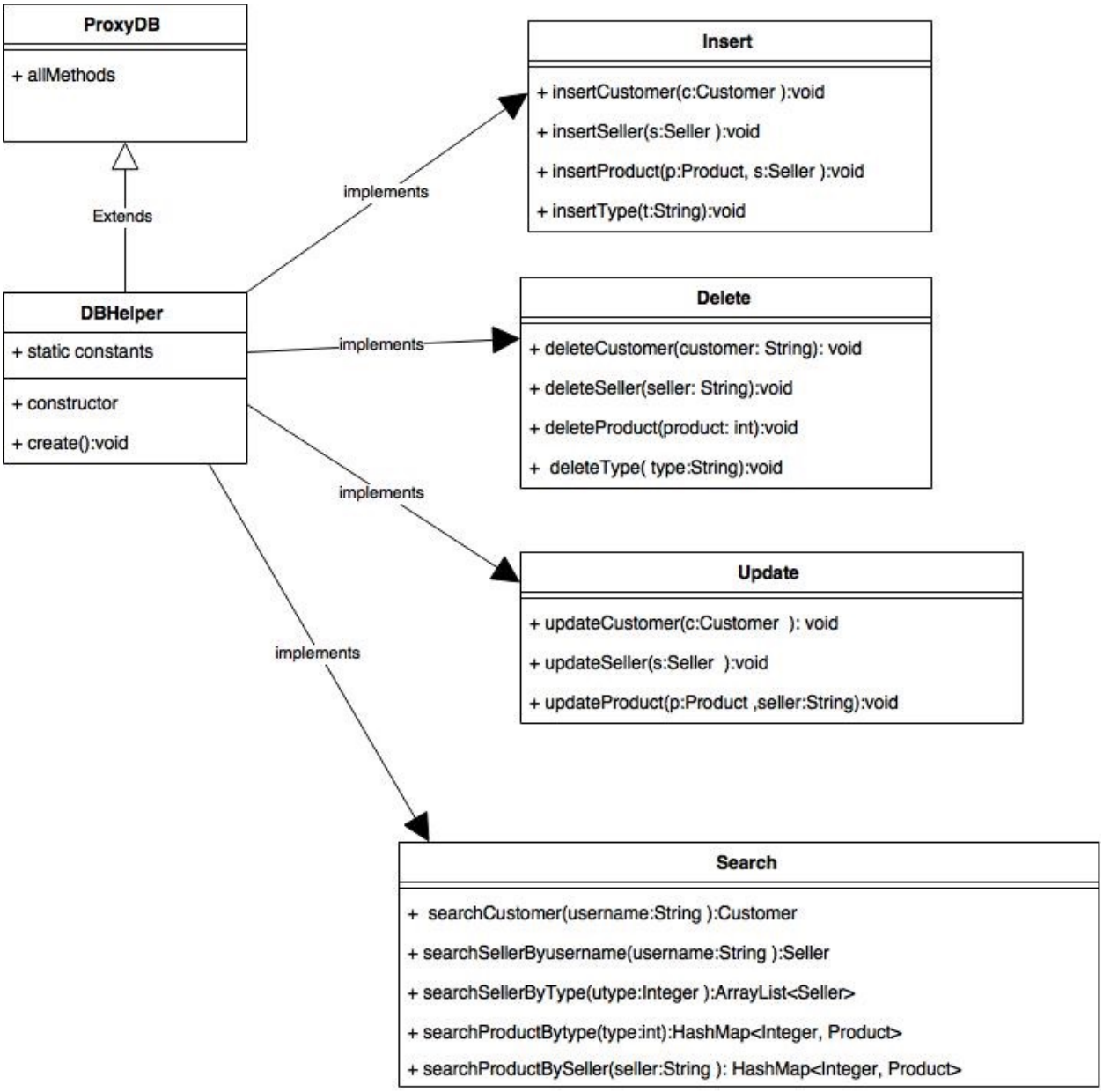


The relationship of the above tables is shown below.



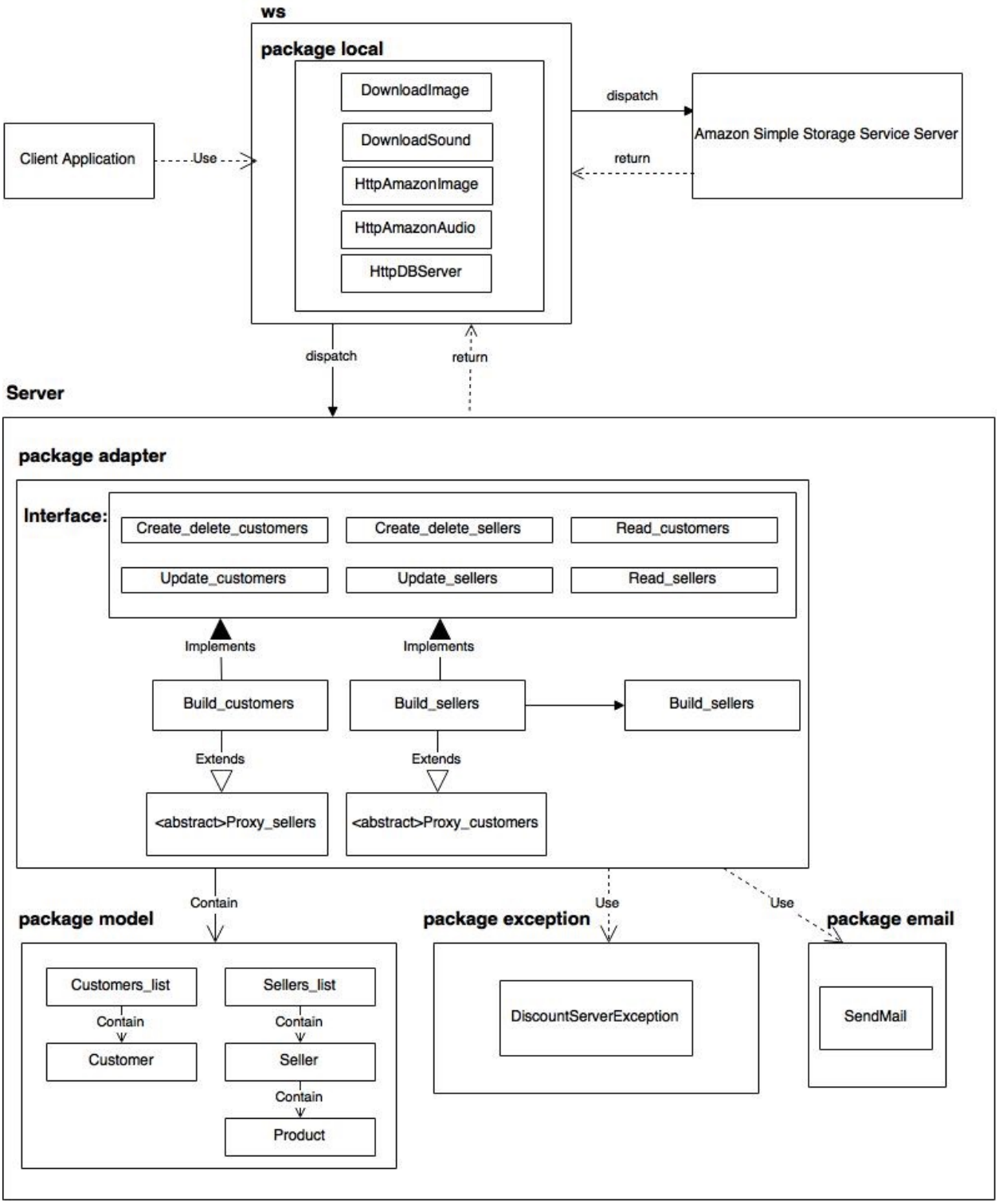
A seller may have several products. Several products could be assigned to the same product type.

3.Database implementation and its operations



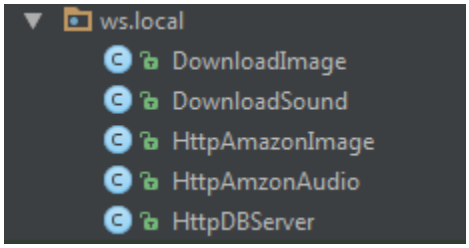
Step 4. Application Tier, Web service and Exceptions handling

Overview of the application tier:



Web service

Web service package provides basic methods for client application to interact with both Amazon Simple Storage Service server and our application server. The structure of this package is as follow,



Following is the class diagram for the ws.local package:

Class: DownloadImage
- url_s: String - bitmap: Bitmap
+ DownloadImage(String) + run(): void + getBitmap():Bitmap

Class: DownloadSound
- url_s: String - uri: Uri
+ DownloadSound(String) + getUri(): Uri

Class: HttpAmazonImage
- Directory: String - Filename:String - Key: String - url: String - file1: File - file2: File - original: Bitmap - basicAWSCredentials: BasicAWSCredentials - BucketName: String - S3Client: AmazonS3Client
+HttpAmazonImage(String,String) +HttpAmazonImage(File) +run():void +getUrl(): String

Class: HttpAmazonSound
- Key: String - url: URL - basicAWSCredentials: BasicAWSCredentials - BucketName: String - S3Client: AmazonS3Client
+HttpAmazonSound(File) +run():void +getUrl():String

Class: HttpDBServer
- url: URL - params: Map<String,String> - encode: String - requestType: int - result: String
+ HttpDBServer(Map<String,String>,String,int) + run():void + getRequestData(Map<String,String>,String,int):StringBuffer + dealResponseResult(InputStream):String + getResult():String

Classes Introduction:

DownloadImage.java: provide client application a method to download image from Amazon Simple Storage Service server.

DownloadSound.java: provide client application a method to download introduction audio of a seller from Amazon Simple Storage Service Server.

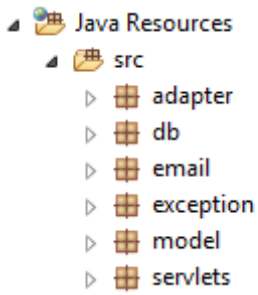
HttpAmazonImage.java: contain a method for client application to upload image to Amazon Simple Storage Service server.

HttpAmazonAudio.java: contain a method for client application to upload audio to Amazon Simple Storage Service server.

HttpDBServer.java: the class providing all kinds of methods for client application to interact with our application server.

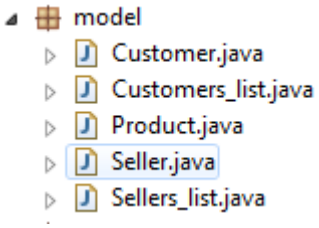
Application Server

The Server project folder contains basic objects to be utilized in presentation tier. The structure of this folder is as follow,



Server project folder has three packages. The model package contains basic classes to be utilized in application tier itself. Adapter package contains abstract classes and interfaces for **encapsulation** and they are exposed to presentation tier. Exception package contains classes for exceptions handling.

At first, the structure of the model package is as follow:



Classes Introduction:

Customer.java: This class contains all attributes for a customer user. In addition, it contains the methods of creating a customer object, setting and getting attributes.

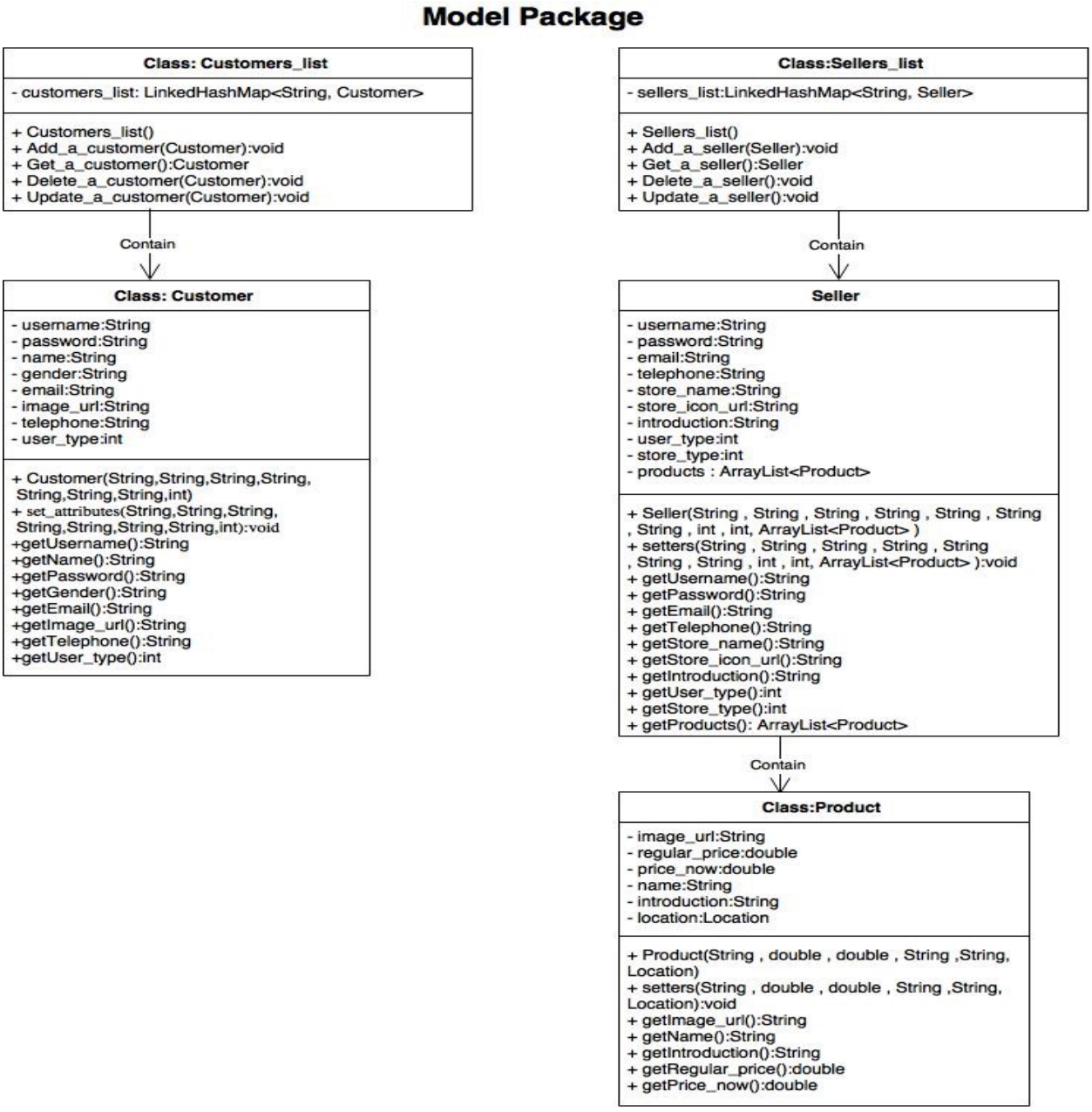
Customer_list.java: This class contains a linked hash map object of customers and its relevant methods which can be manipulated by classes in adapter package.

Seller.java: This class contains all attributes for a seller user and the methods of creating a seller object, setting and getting attributes. An array of Product objects is one of the attributes in this class.

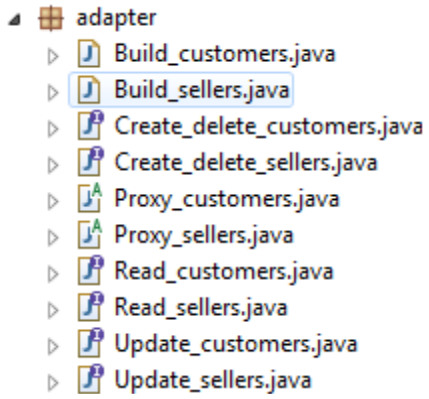
Sellers_list.java: The class containing a linked hash map object of sellers and its operating methods for classes in adapter package.

Product.java: This class is the class containing attributes and methods for a specific item that a seller user wants to sell.

Following is the class diagram for the model package:



At second, the structure of the adapter package is as follow:



Classes Introduction:

Build_customers.java: This is an empty class for encapsulation and it extends the abstract class Proxy_customers.

Build_sellers.java: This is an empty class for encapsulation which extends the abstract class Proxy_sellers.

Proxy_customers.java: The abstract class containing the linked hash map object of customers from model package. It will implements all methods about customers declared in interfaces of this package.

Proxy_sellers.java: The abstract class containing the linked hash map object of sellers from model package. It will implements all methods about sellers declared in interfaces of this package.

Create_delete_customers.java: The interface containing methods for creating and deleting a customer from the linked hash map of customers.

Create_delete_sellers.java: The interface containing methods for creating and deleting a seller from the linked hash map of sellers.

Read_customers.java: The interface declaring methods of reading the information about a customer.

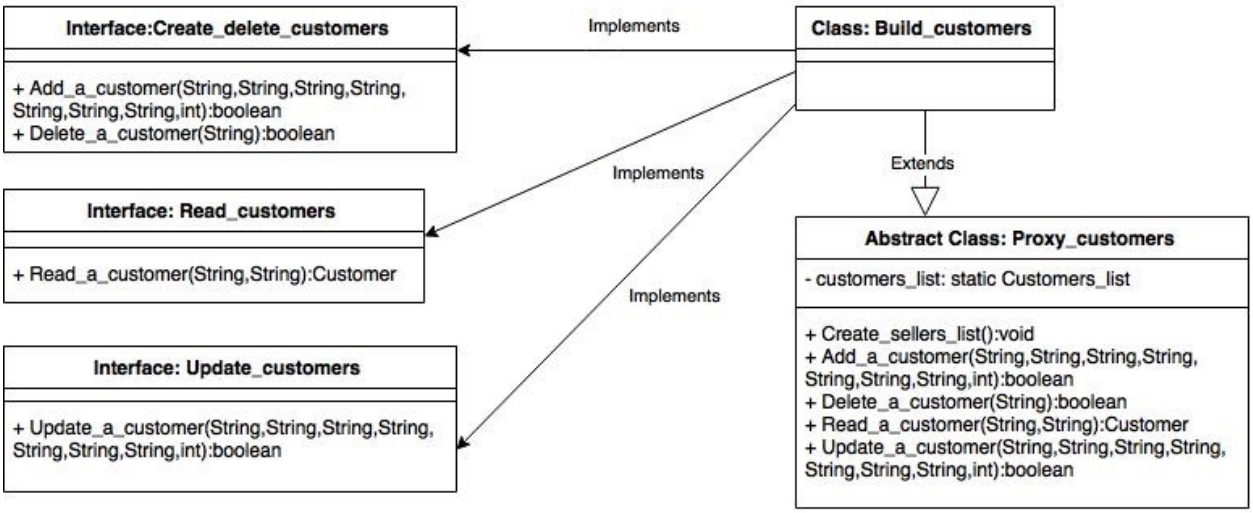
Read_sellers.java: The interface declaring methods of reading the information of a seller.

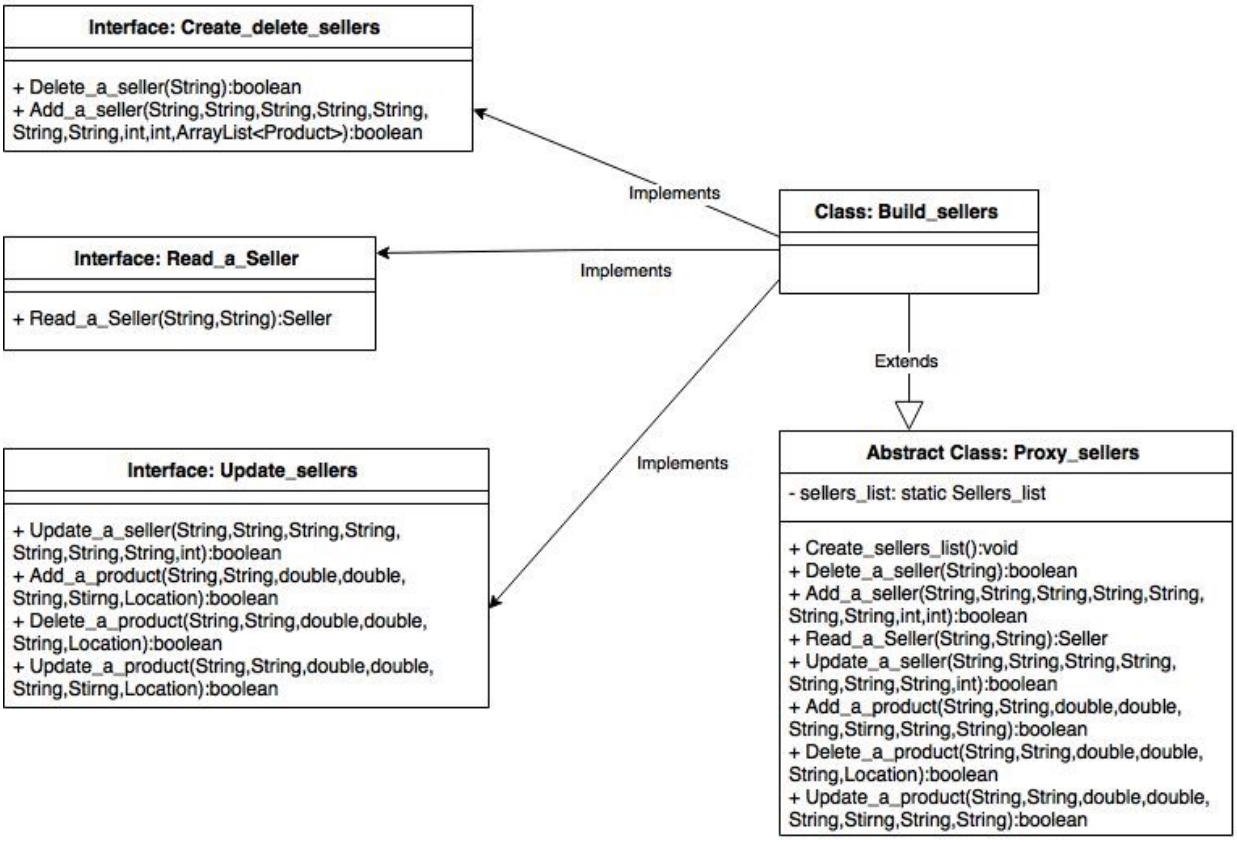
Update_customers.java: The interface contains methods of updating account information for a customer.

Update_sellers.java: The interface containing methods of updating account information for a specific seller user.

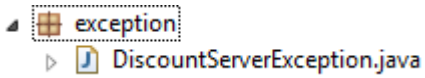
Following is the class diagram for the adapter package:

adapter package





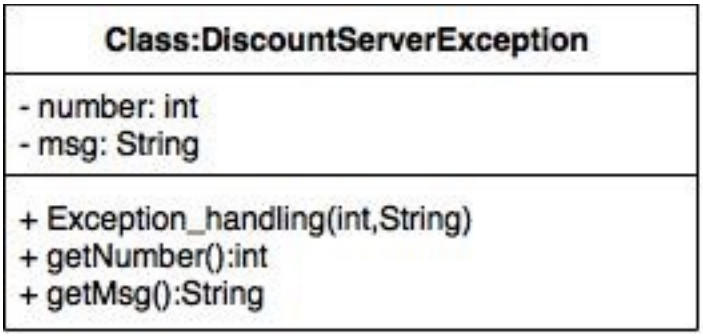
Finally, the structure of the exception package is as follow:



Classes Introduction:

DiscountServerException.java: This is the customer exception class to handle the exceptions in operating the linked hash map objects of customers and sellers.

Following is the class diagram for the exception package:



Step 5. Integration Tier

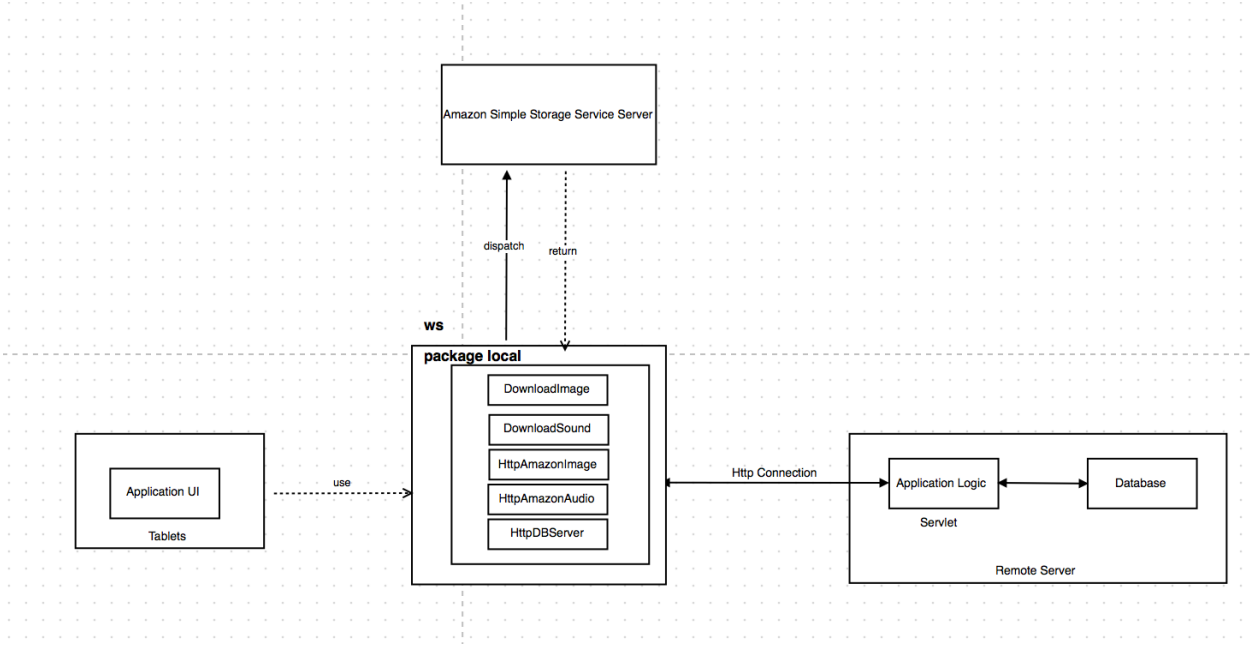
The application is set up as 3 layers in separate devices.

Presentation layer is about the user interfaces, interacting with users to receive request and send request to remote server, receiving data from remote server.

Logic layer is the models of the application. It handles the requests from user and read and write data from/to database. To some extent, it is a bridge between user and permanent data.

Database layer is about CRUD manipulation of the database. It makes the user data permanent, sharable and portable.

UI layer is primarily on tablets or mobiles. Application logic and database will be running on a remote server to manage information and serve clients. The remote server may be a laptop with Tomcat server running on it.



Tablets and remote server are connected through HTTP connection. On the tablet, there will be a package called ‘local’ with a class named HttpDBServer to send request and receive response.

On the remote server, there will be a package called ‘remote’ with several servlets class to handle request and send response to clients.