**Project Report**

Group members: David Alexander Adams and Aleix Molla

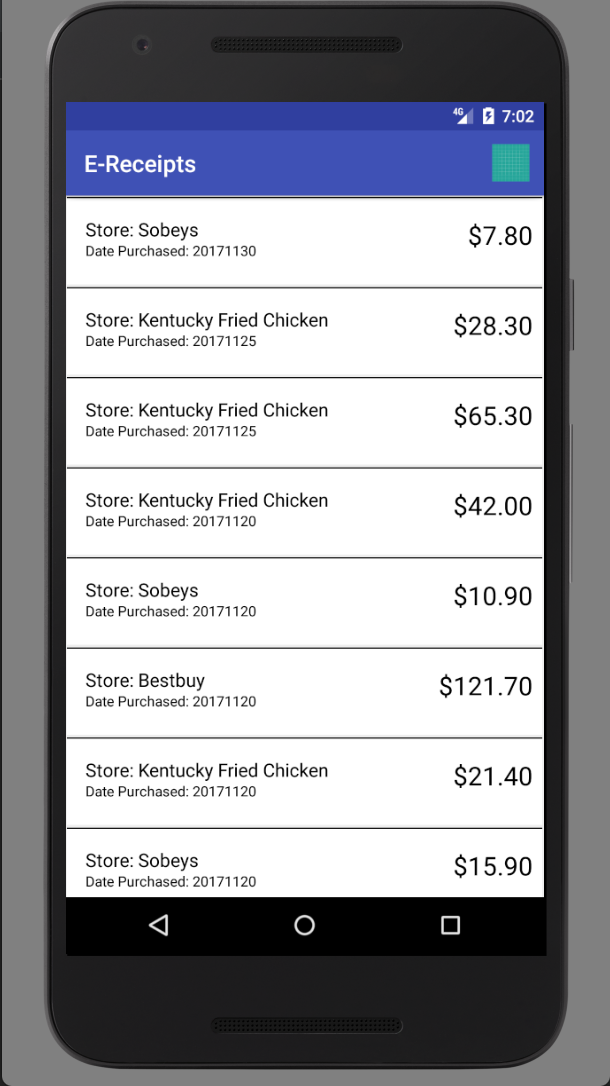
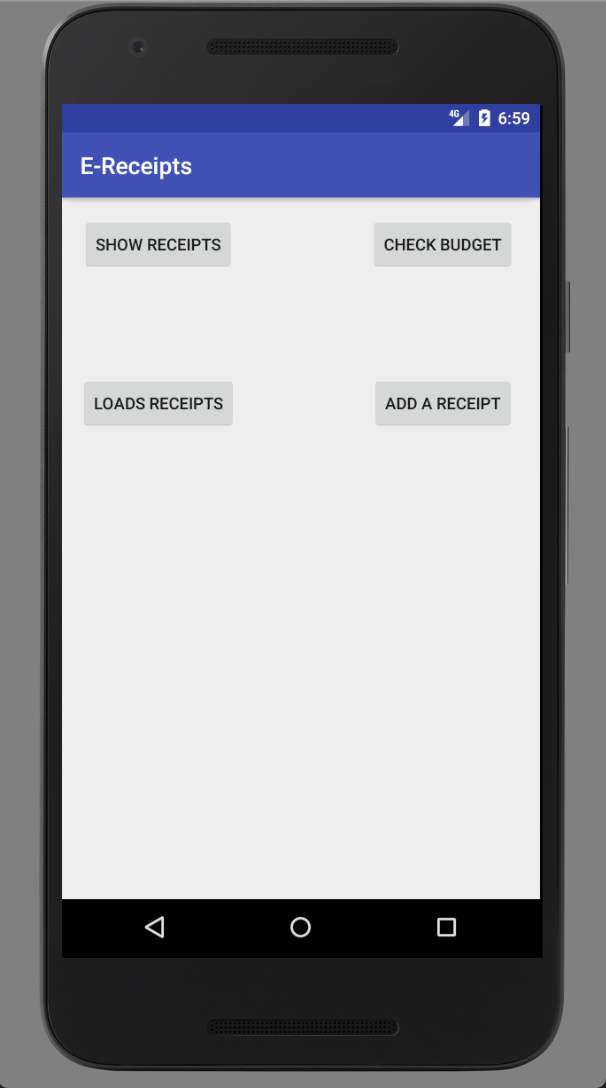
**Project Description**

The app, E-receipt, emphasizes around the idea of storing and accessing receipts on your phone or tablet. In an ideal situation, the app would implement a unique bar code that individual to the device that would be used to access client information. At the checkout, the user would simply scan the bar code, and the receipt will be sent to the phone once the transaction is completed. A full version of this app could also implement Android Pay within the app, such that the user does not have the exit the app.

The app is ideally targeted for consumers who like to keep track of their receipts and finances. In the demonstrated version of the app, pre-created receipts will be simply loaded from a remote server (link provided below) through a JSON object and parsed into a database. The user can then further interact with the receipts by using a calculate a monthly budget option that will graph the receipts from the last two to four months.

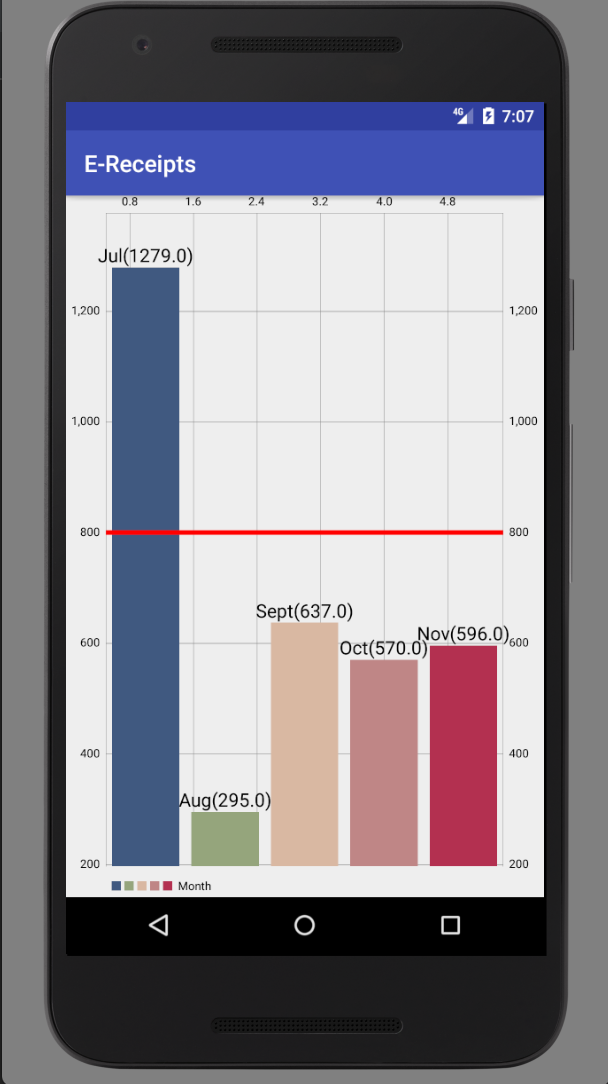
**Operating Instructions**

* Main Activity provides the user with four options
  + Show Receipts
  + Check Budget
  + Load Receipts
  + Add Receipt
* Click Load Receipts
  + Data is loaded from a server
  + User can view receipts
* Click Show Receipts
  + It is only possible to displays receipts if Load Receipts was previously clicked
  + User is able to load a list of receipts from several months
  + User can click on an individual receipt to display the items purchased and their total
* Click Show Budget
  + Application will display a graph of monthly spending from the last five months
* Click Add Receipt
  + User can scan their QR code at the cashier (feature not implemented yet)



**Figure 2 ReceiptListActivity**

**Figure 1: MainActivity**



**Figure 3 Graph Activity**

**Design and Implementation**

The app uses the following classes:

**Callbackable interface:** This interface has one method that is called when the information from the server has finished being added to the database. This method is called in “onPostExecute” in class ReceiptLab. The interface allows the user to access the receipts after they have been added to the database.

**Item:** This class serves as a part of the receipt. Essentially, the receipt contains information about the items bought and the price associated with each item. Since the receipt activity and

**MainActivity:** This class serves as the first page (activity) that the user sees and interacts with after the Splash Screen has finished loading. The MainActivity has four buttons: load receipts, show receipts, budget, and add receipt.

* The load receipts button makes a call to the serves and fetches receipts from the last thirty days. (Note: There may be more than one receipt from one day).
* The show receipts buttons simply loads up an activity that displays a list of receipts that are loaded from the database.
* The budget button provides a visual representation of the total spending of a month
* Add receipt button displays an image of a barcode that later with further development can transfer information.

**Receipt:** This class represents the general receipt object. It contains fields such as store name, date purchased, and items (name of item and price).

**ReceiptActivity:** This class displays one receipt from a particular store and date. It uses textviews to displays the name of the store and the date, and a scroll view to display the list of items and the prices associated with them.

**ReceiptLab:** This class

**ReceiptListActivity:**

**ReceiptPagerActivity:**

**DataBase:**

* + ReceiptCursorWrapper
  + ReceiptDbSchema
  + ReceiptHolder

**Use Cases**

**Implementation Issues and Bugs**

Database Issues: We created just one table with all the information we need from the server but we our items data is an array of paired Item objects of Strings and Doubles link together. Like the database only receives primitive data types (String, char, int, etc.), we need to store them as a JsonArray String and parse it back to Items every time we do a query for every receipt. A solution given by Andrew was to create a second Database Table with those Item elements that shared the UUID of their Receipts in the other Database Table, for time issues we did not change it.

Creating Receipts: As we cannot correctly simulate any Store machine to send the receipt data, we need to make calls to a randomly generated receipt from a server, creating specific design and layout implementations that won’t be use in the original intended App.

Implementing with Array: The server that generates receipts is parsed into an array (not array list) to be displayed when a receipt is clicked from the recycler view (of receipts). The issue that we had was not being able to load a parcelable array. We tried using a serializable but the activity launched from the recycler view would crash. Therefore, we had to change our Items to be held in an ArrayList since that was not causing any issues.

**Important Features**

Traverse through your Receipts sorted by Date of purchase or Store name and be able to View the full contend of your selected receipt.

Look into your expenses of the last 6 month to plan a budget providing a chart, also you will be able to set a limit bar on your chart to know when you pass the limit of your budget.