CSCI 6907 Software Handheld Devices

Final Project Report on

Budget

Jaywant Kapadnis

\

Abstract

Budget application features effortless methods for adding new transactions and going over your allotted chunk of spending money, and comes with awesome features that are missing from some of the other apps. A good example is when adding a new transaction you'll be presented with options for including location information or a quick note, which effectively pushes the customizability and flexibility of your transactions beyond most basic budgeting apps.

Contents

1. Introduction

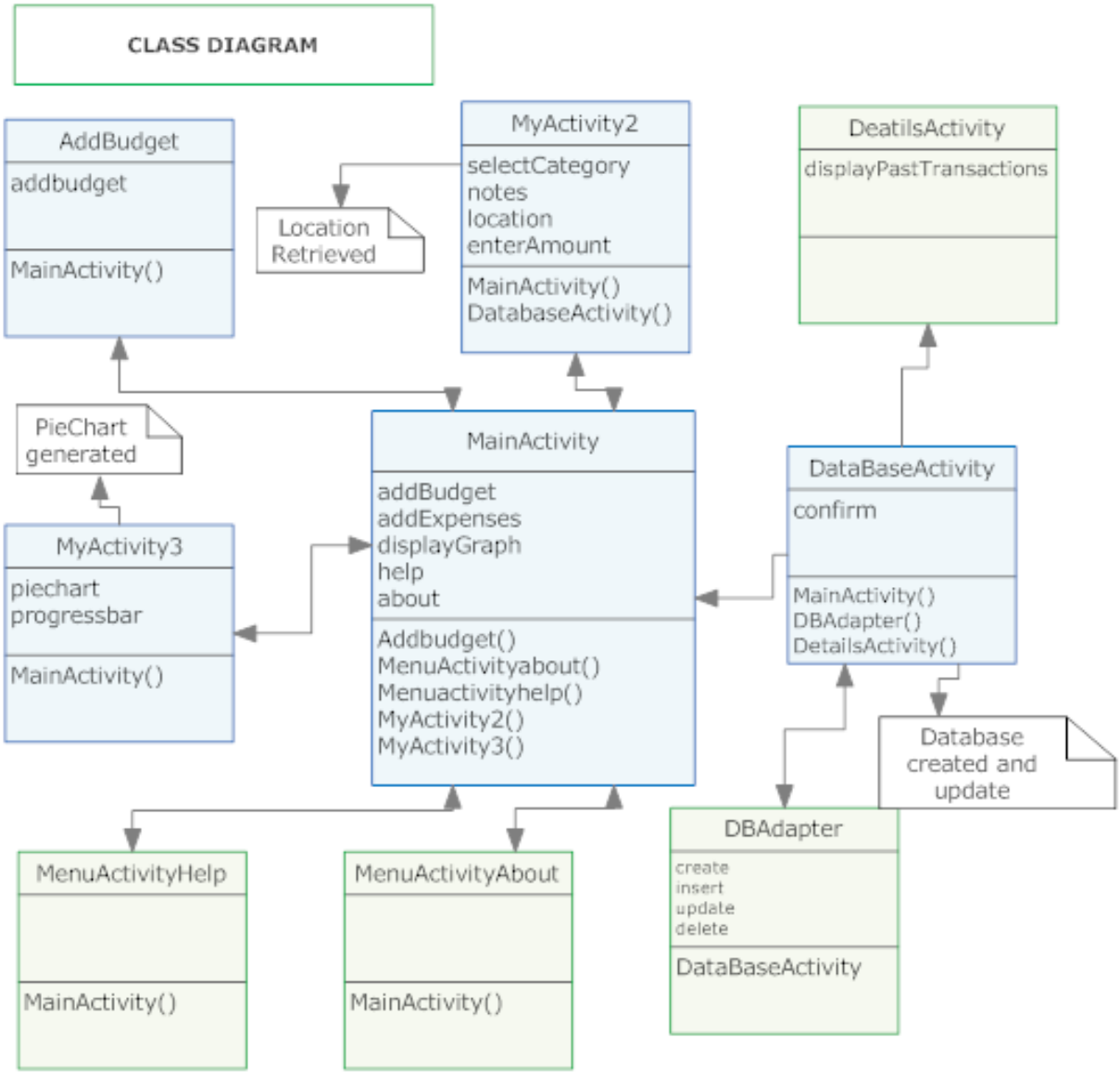
This project implements a monthly budget application that will keep a track your monthly expenses which includes the total expenditure during that particular month. This application gives a detail overview of the expenditure during that month. Each entity in the expenditure will be accompanied with Category, Note, amount and Location information. The application also displays a graph of the overall expenditure and gives progress bar telling the remaining balance. Users are given choice to dynamically adjust the budget for the month.
This application helps to manage the monthly budget more effectively with a simple user interface.

2. Project Configurations

- IDE: Eclipse SDK
  Version: 3.7.0
- System Configuration: Mac OS X lion (10.7.2)
- Development Platform: Java SDK 1.6.0
- Database: Android SQLite
- Target Name: Google APIs
- Platform 4.0
- API level 14

3. Architecture

Class diagram



4. How this Application Works?

This application is made to have a simple and effective user interface that includes all the possible entity that a user would like to add when one make a transaction.

When a user purchases a product he opens the application to add the product in the application. User is given different option to add that transaction like Category, Notes if any, amount and location information associated with the product that was purchased.
User is given ten different categories to choose from. When a product is purchased user can add note associated which the product and the amount.
With a simple click application gets the users location coordinates in Latitude and Longitude, the application then uses these coordinates to get the address of the current location. Total four different address associated with the coordinates is available to user to select from. Then after selection user is gives the confirmation. The application then shows the user preferences selected associated with that product. User has to choice to edit that data or save the transaction in the Database. In the same page user is given a choice to view all the past transactions. After saving the transaction at the Home page the user can view a detail pie chart of all the transactions made and view the available balance.

5. Implementation

5.1 Add Budget

The first step in the application is to enter a correct monthly budget. This is very easy with a simple click of a button. This budget is then used by the application for all the calculations. Budget can be modified anytime in between the application and accordingly the calculations changes giving the corresponding Pie Chart and available budget for the month.

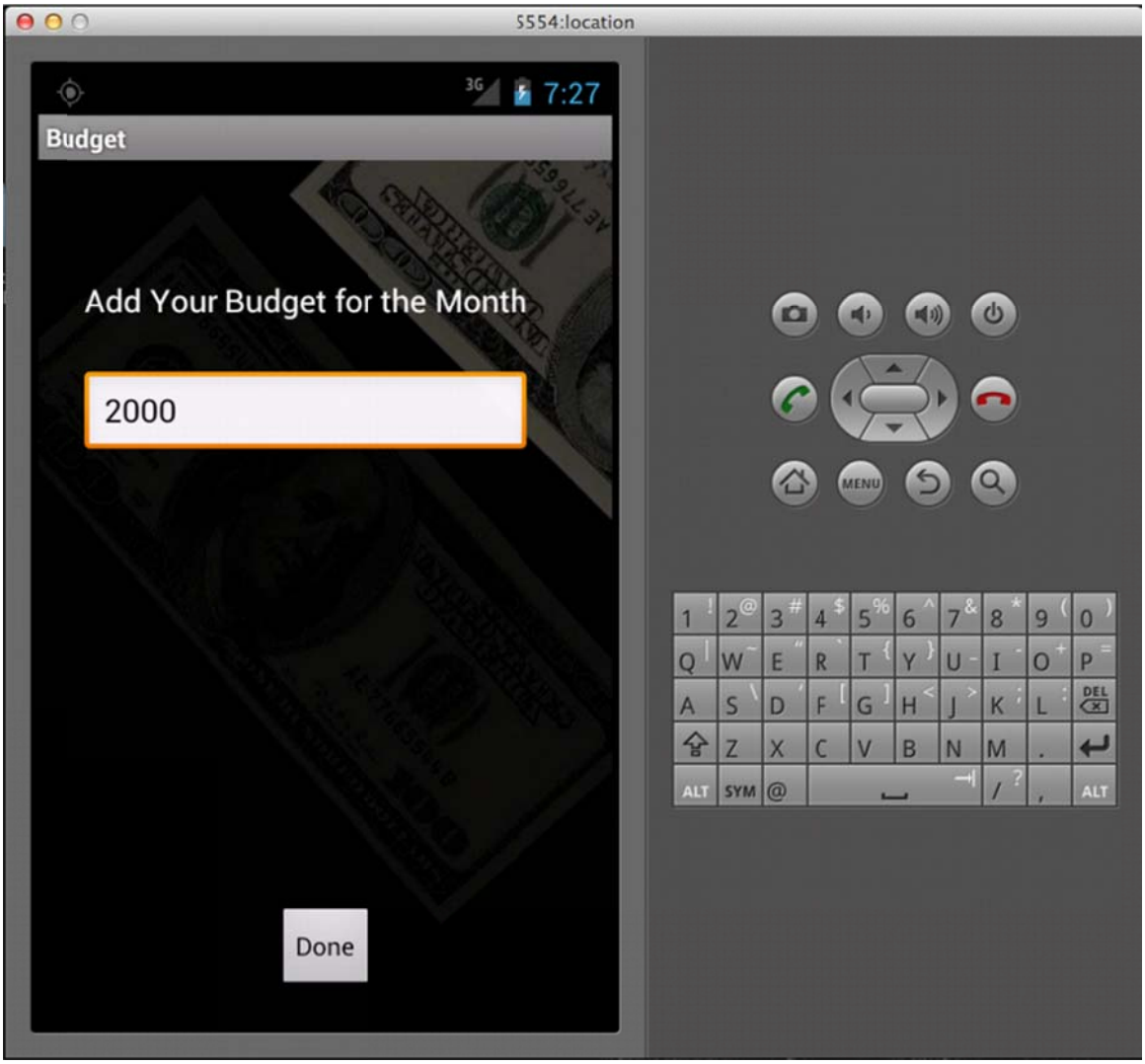


Figure 1: Add you Monthly Budget in $

5.2 Transaction

All the entities associated with the transaction are save in SQLite database, which is implemented in the program. With four column as shown in the figure:

Category (_id)	Notes	Address	Price

This database table is updated as transaction is added. Show transaction triggers the database and retrieves the past transaction.

These transactions are then viewed in the list view with four columns. Every-time as the transaction is added the Database is updated which can view using show transaction button.
The figure below shows the actual database form that is used in the application.

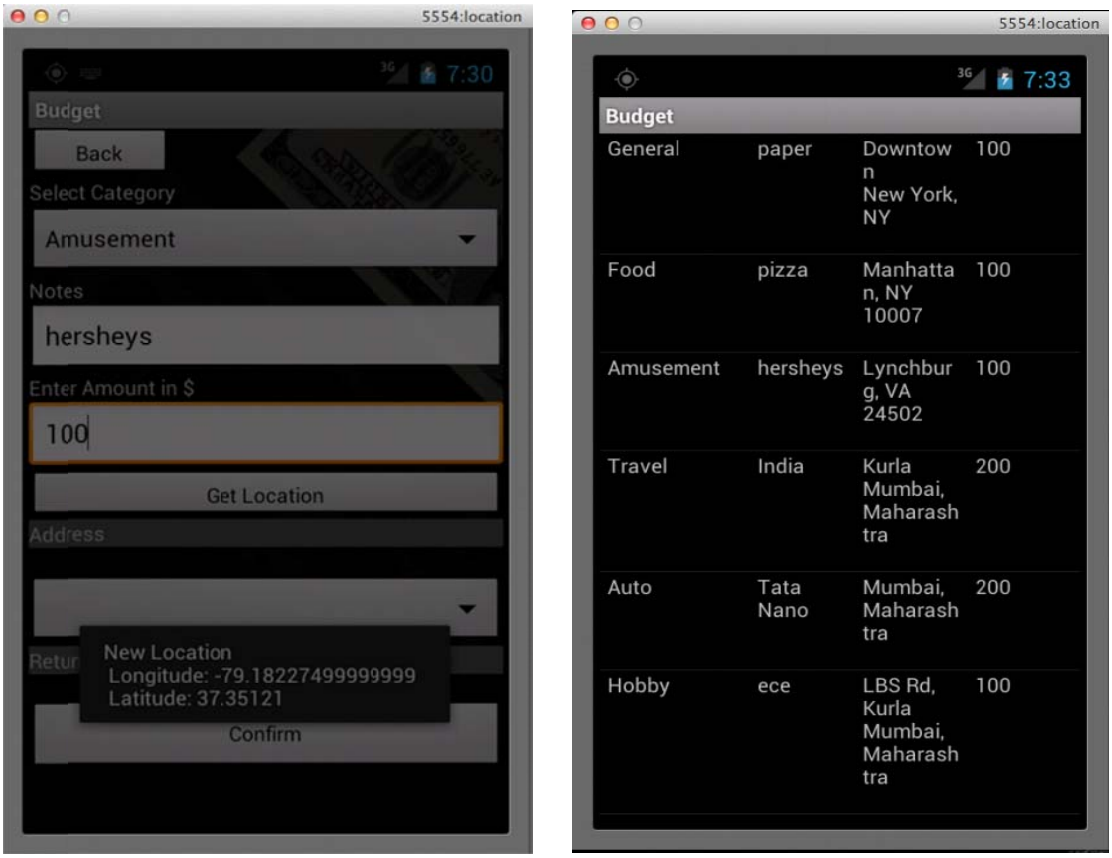


Figure 2: The Database used in the application

5.3 Android SQLite Database

In this application I have implemented a helper class called DBAdapter that creates, opens, closes, and uses an SQLite database.
The DATABASE_CREATE constant contains the SQL statement for creating the titles table within the budget database.

The DBAdapter class extends the SQLiteOpenHelper class—an Android helper class for database creation and versioning management. In particular it override the onCreate() and onUpgrade() methods.
The onCreate() method creates a new database if the required database is not present. The onUpgrade() method is called when the database needs to be upgraded. This is achieved by checking the value defined in the DATABASE_VERSION constant. For this implementation of the onUpgrade() method, you will simply drop the table and create the table again.
This allows the application to define various methods to open and close the Database as well as the methods for adding/editing/deleting rows in the table.

Android uses the Cursor class as a return value for queries. Cursor can be thought as a pointer to the result set from a database query. Cursor allows Android to more efficiently manage rows and columns as and when needed. Now the put() method allows you to insert keys with values of different data types.
To add a title into the titles table, use the insertTitle() method of the DBAdapter class.
To retrieve all the titles in the titles table, use the DBAdapter class' getAllTitles() method.
The result is returned as a Cursor object. To display all the titles, the application first calls the Cursor object's moveToFirst() method. If it succeeds (which means there is at least one row available), adds the details of the title to the array using the DisplayTitle() method .To move to the next title, call the Cursor object's moveToNext() method.

5.4 Address Retrieval

Budget uses Location based Services that is provided by the GPS in the android device. The latitude and Longitudes obtained are very accurate which allows second part of the method to give even more accurate addresses surrounding these coordinates.
The first step is to take reference of the LocationManager class, which provides access to the system location services. This is done using the getSystemService of the application activity. Then, application requests

updates of the device's location using the method requestLocationUpdates. In that method, I have provided the name of the preferred location provider (in our case GPS), the minimum time interval for notifications (in milliseconds), the minimum distance interval for notifications (in meters) and finally a class implementing the LocationListener interface. That interface declares methods for handling changes in the user's location as well as changes in the location provider's status.

For the LocationListener interface, I have implemented the MyLocationListener inner class. The methods in that class just use Toasts to provide info about the GPS status or any location changes. The only interface element is a Button, which gets hooked up, with an OnClickListener and when it is clicked, the showCurrentLocation method is invoked. Then, the getLastKnownLocation of the LocationManager instance is executed returning the last known Location. From a Location object we can get information regarding the user's altitude, latitude, longitude, speed etc. In order to be able to run the above code, the necessary permissions have to be granted. These are:

- ACCESS_FINE_LOCATION
- ACCESS_MOCK_LOCATION
- ACCESS_COARSE_LOCATION

Now we have the coordinates to map the correct address surrounding users position.

5.5 Android Geocoder

This class is used for handling geocoding and reverse geocoding. Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate. In our case we are using Reverse geocoding.
Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into a (partial) address. The amount of detail in a reverse geocoded location description may vary, for example one might contain

the full street address of the closest building, while another might contain only a city name and postal code. The Geocoder class requires a backend service that is not included in the core android framework. The Geocoder query methods will return an empty list if there no backend service in the platform.

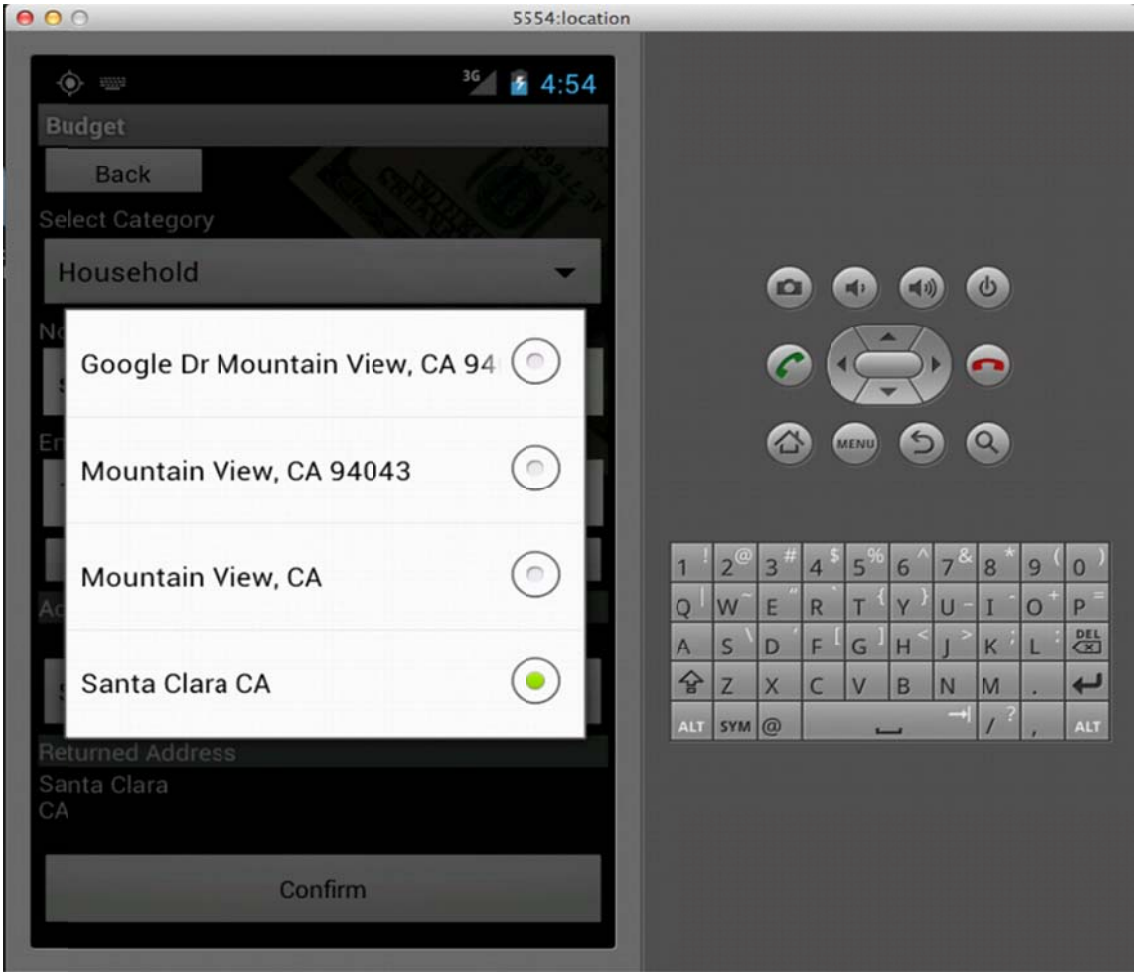


Figure 3: Four Address retrieved from the available coordinated that were provided my the Emulator settings
Application allows the user to select from the available locations. This is further stored in the database, which can be retrieved in future.

The final confirmation screen look like as shown below:

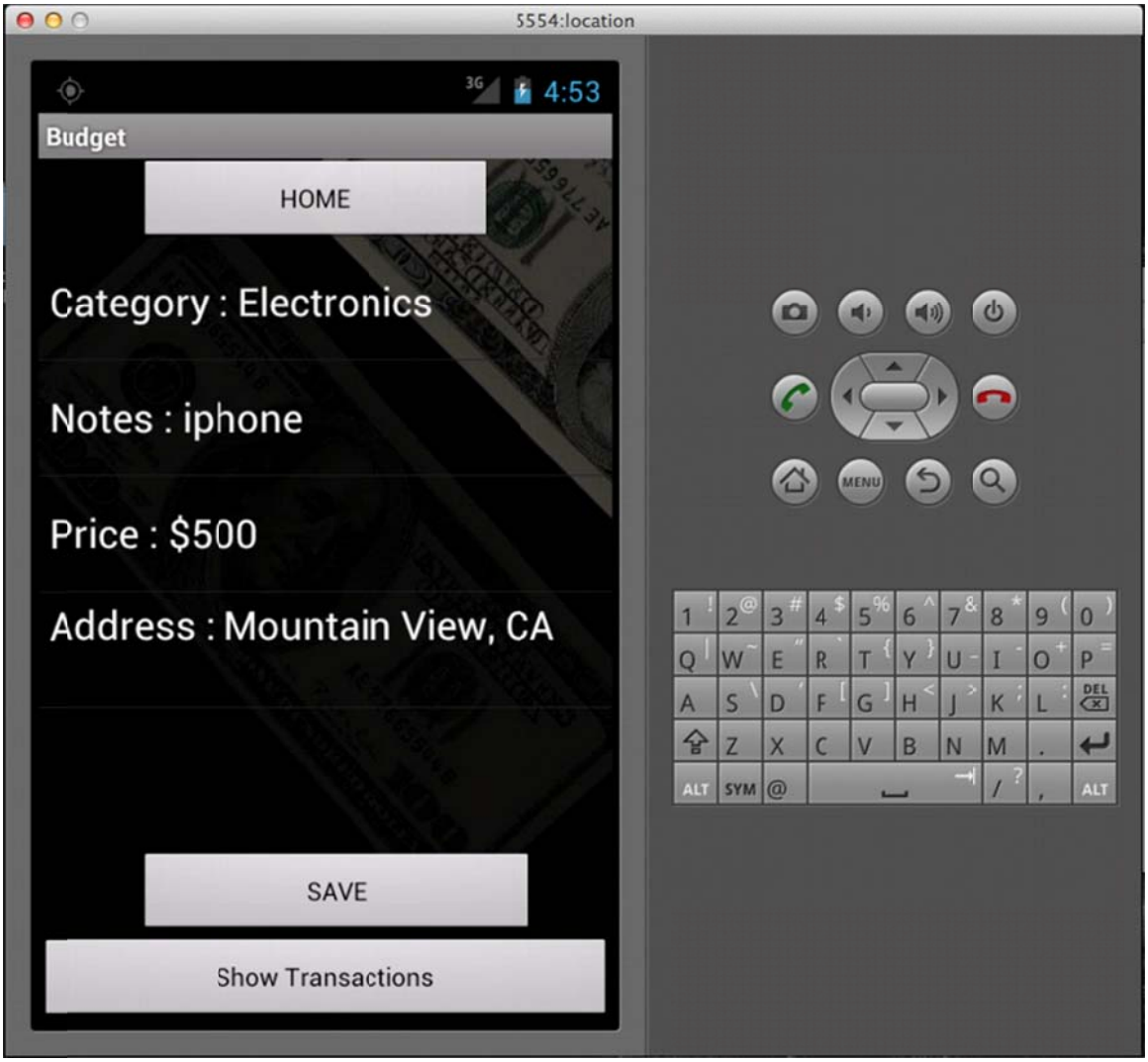


Figure 4: Final Transaction confirmation Screen.

5.6 Getting the Pie Chart

This application uses Google Chart API.

The Google Chart API returns a chart image in response to a URL GET or POST request. The API can generate many kinds of charts, from pie or line charts to QR codes and formulas. All the information about the chart that user wants such as chart data, size, colors, and labels, are part of the URL. Using the web-view format provided by the android layout this API produces the chart that gives an accurate chart as per your expenditure and available balance.

The progress bar, which is again provided by the android layout manager, is then adjusted to give the available balance.

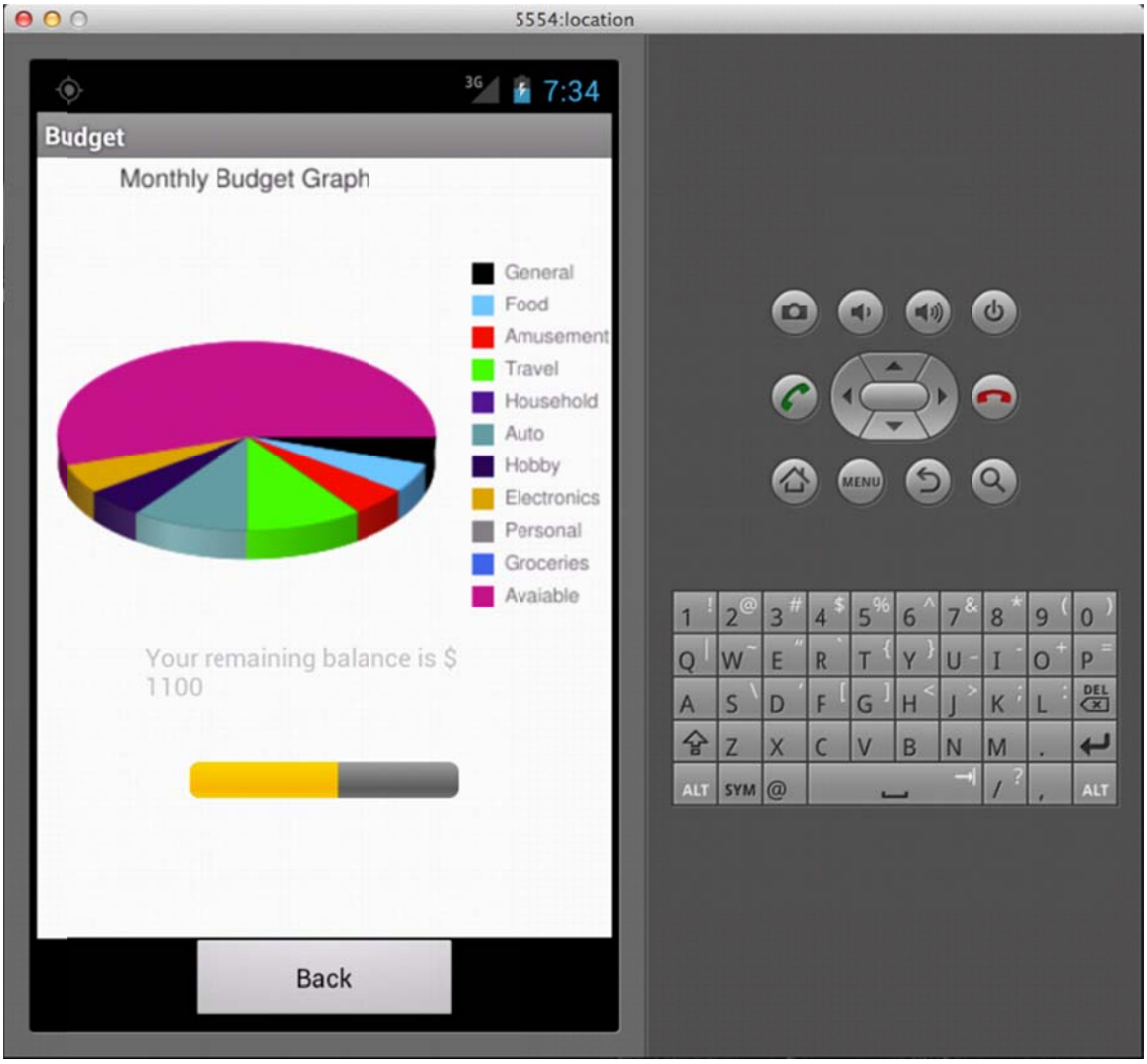


Figure 5: Detail Pie Chart and Remaining balance information

6. Problems and Issues

I wanted to add image of the product purchased, Image would be selected from the camera library or user will have a choice to take snapshot of the product using the camera. I also wanted an option to the user to select the image from the Internet. But I couldn't figure out how

to save the image in the database and retrieve it along with the particular transaction this was too time consuming so I scrapped it.

Also I wanted the database to be more user friendly giving the user an option to directly delete a particular transaction, which will automatically update the pie chart and available balance. I did try to delete a particular transaction but it wasn't full proof to add in application and this was making the application more complex.

7. Conclusion

I was able accomplished all the things I wanted in this project and I have implemented all the elements that were required in the BUDGET application.
This application can be used in real life and this can surely keep track of your monthly budget more efficiently. There are many budget applications available in the Android Market but what makes this unique is its ability to get the location information from where the product was purchased. This allows the user shortlist the store locations; with the database available user can select the store that provides the product at most affordable rate.

References

1.http://developer.android.com/reference/android/location/Geocoder.html
2.http://www.javacodegeeks.com/2010/09/android-location-based-services.html
3.http://stackoverflow.com/questions/472313/android-reverse-geocoding-getfromlocation
4. http://code.google.com/apis/chart/