

National College of Ireland  
BSc in Computing  
2015/2016

Mahmoud Azzam  
13110241  
X13110241@student.ncirl.ie

## ***LocalDeal***

Final Report



# Table of Contents

## Table of Contents

Executive Summary .....	4
1 Introduction .....	5
1.1 Background.....	5
1.2 Aims.....	5
1.3 Technologies .....	5
2 System.....	6
2.1 Requirements .....	6
2.1.1 Functional requirements.....	6
2.1.2 Data requirements.....	6
2.1.3 User requirements.....	6
2.1.4 Environmental requirements .....	6
2.1.5 Usability requirements.....	6
2.2 Design and Architecture.....	7
2.3 Implementation .....	10
2.3.1 Coding principles:.....	10
2.3.2 Responsive design:.....	14
2.3.3 Features implementation:.....	14
2.3.4 Deployment: .....	26
2.3.5 Problems solved:.....	26
2.4 Testing.....	29
2.4.1 Unit Testing: .....	29
2.4.2 Integration testing:.....	29
2.4.3 Backward compatibility testing: .....	29
2.4.4 User acceptance testing:.....	30
2.4.5 System testing.....	30
2.5 Graphical User Interface (GUI) Layout.....	31
2.6 Evaluation .....	33
3 Conclusions .....	34
3.1 Advantages of LocalDeal .....	34

3.2	Disadvantages of LocalDeal .....	34
3.3	Market analysis: .....	34
4	Further development or research.....	35
5	Bibliography .....	36
6	Appendix.....	37
6.1	Project Proposal .....	37
6.2	Project Plan .....	39
6.3	Monthly Journals.....	39

## **Executive Summary**

Nowadays, big companies dominate most of the businesses so books are bought from Amazon, groceries are bought from big supermarkets, etc. This phenomena is affecting local businesses which has good products to offer as well but do not get the attention of the buyers. This phenomena has few reasons, firstly big companies are always having offers beside which they are selling you other non-discounted products, secondly their addresses are well-known so you do not have to look around in addition to other reasons such as the variety of products, etc.

My application is made to solve the first two problems so local businesses are able to advertise their offers to attract customers in addition clear on map address of the business, clear description and pre-set expiry date of the offer.

This solution will be very appealing to customers as well who are always looking for cheaper products in their neighbourhood.

The application provides two interfaces, one for sellers to post, view and edit their deals. Another interface for buyers to locate local deals based on search word or category and the maximum distance accepted between them and the searched deal and then locate it on map.

# **1 Introduction**

## **1.1 Background**

The intended customers are buyers looking for bargains, deals, discounts within close distance of their geolocation (1km, 5km, 25 km) and sellers targeting immediate close potential buyers.

## **1.2 Aims**

Is to help local businesses gain some market share that has been taken by big companies, help them reach out to customers and offer their good deals, help them reduce the cost of advertising, help buyers find appealing deals within their neighbourhood or city.

## **1.3 Technologies**

Android – Operating system targeted

Java – Programming language used

XML – User interface

JSON – data transmission between server and client

Mysql – used for database

PHP – Used to build interfaces that handle database queries

Wamp – local server used for development

## **2 System**

### **2.1 Requirements**

#### **2.1.1 Functional requirements**

1. Seller should be able to subscribe
2. Seller should be able to post deal
3. Buyer should be able to view deal
4. Buyer should be able to locate deal
5. Buyer should be able to leave feedback

#### **2.1.2 Data requirements**

- Data is provided by sellers to be later viewed by buyers
- Data is stored on the server
- Calls to the database using PHP are required to save or fetch data from the database

#### **2.1.3 User requirements**

- User needs to have an Android phone with an operating system version 2.2 and above
- Android phone needs to be connected to the internet
- Android phone needs to have GPS receiver

#### **2.1.4 Environmental requirements**

- Android device to run the application
- Internet connection
- GPS signal

#### **2.1.5 Usability requirements**

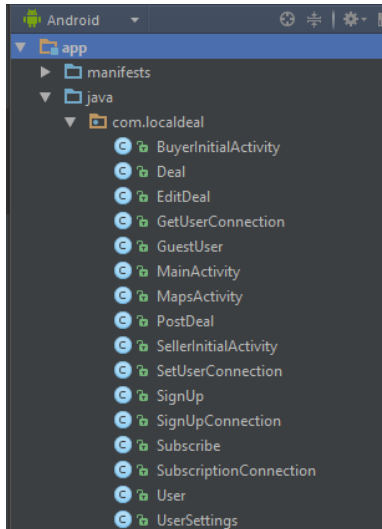
- Understandable : Layout's elements should be easy enough and self-explanatory for user to navigate through the application
- Operable : Steps should be consistent with popup messages to guide the user
- Learnability : User should be able to learn the features of the application and use them quickly
- Consistency : Layout and features should be consistent with the Android guidelines and design so user will feel familiar with the application's components

## **2.2 Design and Architecture**













The application is targeting the Android platform (version 2.2 and above). Java is used since it is the native language to program for Android. Mysql is used on the server side to store the data. PHP interfaces are used to coordinates the calls between the application and database.

### **Design**

**Java classes** are written to handle the user actions like PostDeal, EditDeal, Subscribe, etc

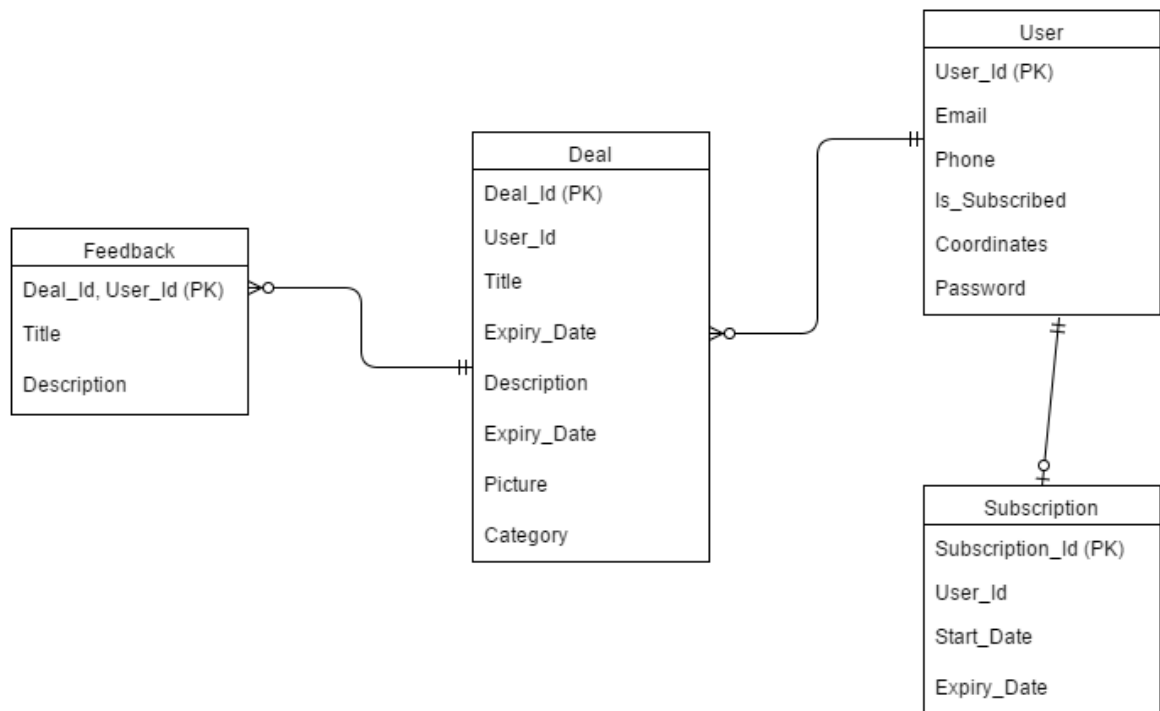


**PHP files** are written to handle calls from the application to the database

-  deleteDeal.php
-  editDeal.php
-  getMyDeals.php
-  getSubscriptionAttributes.php
-  getUserAttributes.php
-  HelloWorld.php
-  index.php
-  searchDeals.php
-  setDealAttributes.php
-  setUserAttributes.php
-  signup.php
-  uploadPicture.php

**Mysql database** is the main storage place for all the applications' data

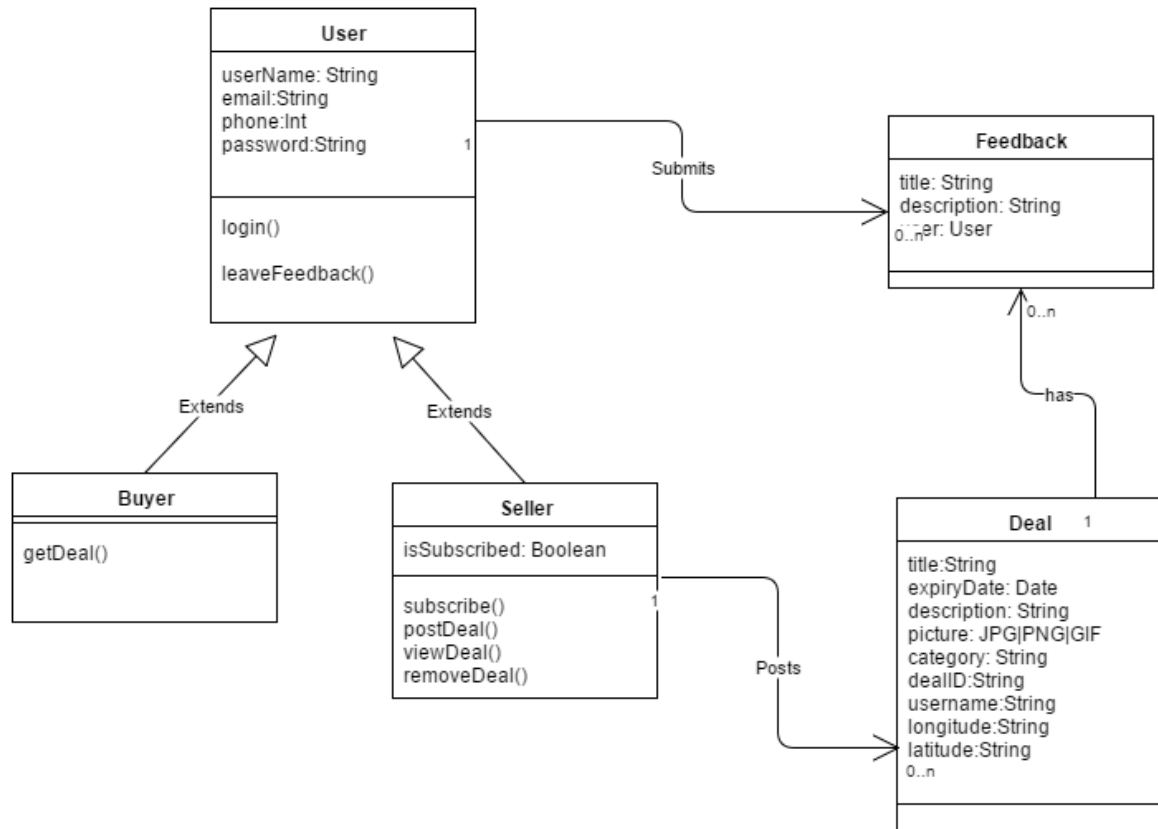
## *LocalDeal*





# Architecture

## Class Diagram

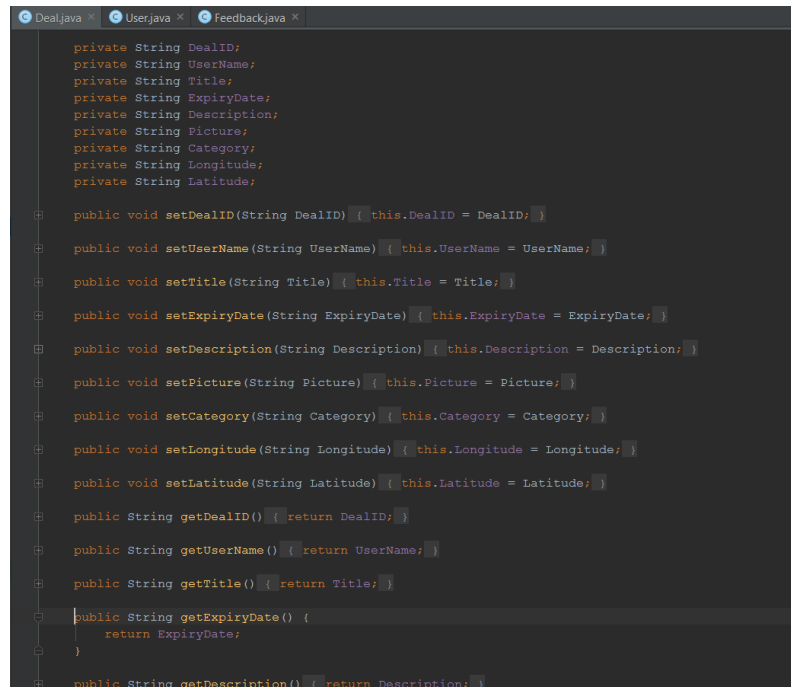


## 2.3 Implementation

### 2.3.1 Coding principles:

Were taken into consideration during the implementation from day one whether in the three main components of localdeal:

- In Java:
  - Object oriented main principles like Abstraction, encapsulation, inheritance and polymorphism were incorporated in the design and implementation of localdeal.
  - Encapsulation was achieved by having classes that represents the Deal, User and Feedback, holding their attributes and providing getters and setters for each of them. (See below)



```
Deal.java × User.java × Feedback.java ×
private String DealID;
private String UserName;
private String Title;
private String ExpiryDate;
private String Description;
private String Picture;
private String Category;
private String Longitude;
private String Latitude;

public void setDealID(String DealID) { this.DealID = DealID; }
public void setUsername(String UserName) { this.UserName = UserName; }
public void setTitle(String Title) { this.Title = Title; }
public void setExpiryDate(String ExpiryDate) { this.ExpiryDate = ExpiryDate; }
public void setDescription(String Description) { this.Description = Description; }
public void setPicture(String Picture) { this.Picture = Picture; }
public void setCategory(String Category) { this.Category = Category; }
public void setLongitude(String Longitude) { this.Longitude = Longitude; }
public void setLatitude(String Latitude) { this.Latitude = Latitude; }
public String getDealID() { return DealID; }
public String getUsername() { return UserName; }
public String getTitle() { return Title; }
public String getExpiryDate() {
    return ExpiryDate;
}
public String getDescription() { return Description; }
```

```

Deal.java x User.java x Feedback.java x
package com.localdeal;

import ...

/**...*/
public class User implements Serializable {

    protected String username;
    protected String email;
    protected int phone;
    protected String password;
    protected int subscriptionID;
    protected int subscriptionStartDate;
    protected int subscriptionEndDate;

    public User(String username) { this.username = username; }

    public int getSubscriptionID(String username) {...}

    public int getSubscriptionStartDate(String username) {...}

    public int getSubscriptionEndDate(String username) {...}

    public void setEmail(String email) {...}

    public String getEmail(String username) {...}

    public void setPhone(String phone) {...}

    public int getPhone(String username) {...}

    public void setPassword(String password) {...}

    public String getPassword(String username) {...}

}

```

```

Deal.java x User.java x Feedback.java x
package com.localdeal;

import java.io.Serializable;

/**
 * Created by mazzam on 5/6/2016.
 */
public class Feedback implements Serializable {

    private String DealID;
    private String UserName;
    private String Title;
    private String Description;
    private String Rating;
    private String Date;

    public void setDealID(String DealID) { this.DealID = DealID; }

    public void setUserName(String UserName) { this.UserName = UserName; }

    public void setTitle(String Title) { this.Title = Title; }

    public void setDescription(String Description) { this.Description = Description; }

    public void setRating(String Rating) { this.Rating = Rating; }

    public void setDate(String Date) { this.Date = Date; }

    public String getDealID() { return DealID; }

    public String getUserName() { return UserName; }

    public String getTitle() { return Title; }

    public String getDescription() { return Description; }

    public String getRating() { return Rating; }

    public String getDate() { return Date; }
}

```

- Inheritance was used in many situations to minimise the duplication of code and to write understandable reusable and extensible pieces of code. (See below)

```

User.java x
public int getSubscriptionEndDate(String username) {...}

public void setEmail(String email) {...}

public String getEmail(String username) {
    this.username = username;
    String field = "Email";
    GetUserConnection handler = new GetUserConnection();
    handler.execute(username, field);
    try {
        email = handler.get();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }

    return email;
}

public void setPhone(String phone) {...}

public int getPhone(String username) {
    this.username = username;
    String field = "Phone";
    GetUserConnection handler = new GetUserConnection();
    handler.execute(username, field);
    try {
        phone = Integer.parseInt(handler.get());
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }

    return phone;
}

```

- Threads are used whenever possible to handle processing expensive tasks like accessing the network, extensive calculations to determine deal distance from the user location and getting the location coordinates. I use them to improve performance and avoid blocking the main UI thread
- In PHP, I used similar object oriented principles in PHP to minimize the duplication of code, make the code simpler and maintainable. Like creating a class that holds the database login details that is extended later by the other files so if I have to change the DB password, I will have to change only in one class which makes the files easy to maintain. (See below)

```

<?php
require_once('./Confidential/connection.php');

$con=mysqli_connect($host,$user, $pass, $db);
if (mysqli_connect_errno($con))
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$dealID = $_POST['dealID'];

mysqli_query($con,"DELETE FROM deals WHERE DealID='$dealID'");














































mysqli_close($con);






```

- In Mysql, I used the normalization process during the design to create a duplication free database and remove transitive dependencies. (See below)

Server: localhost Database: a4117012\_ID Table: deals

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[Import](#)
[Operations](#)
[Empty](#)
[Drop](#)





	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	DealID	mediumint(9)			No		auto_increment	    
<input type="checkbox"/>	UserName	varchar(30)	latin1_general_ci		No			    
<input type="checkbox"/>	Title	varchar(50)	latin1_general_ci		No			    
<input type="checkbox"/>	ExpiryDate	int(11)			No			    
<input type="checkbox"/>	Description	blob		BINARY	Yes	NULL		    
<input type="checkbox"/>	Picture	blob		BINARY	Yes	NULL		    
<input type="checkbox"/>	Category	varchar(50)	latin1_general_ci		Yes	NULL		    
<input type="checkbox"/>	Longitude	double			Yes	NULL		    
<input type="checkbox"/>	Latitude	double			Yes	NULL		    

[Check All](#) / [Uncheck All](#) With selected:     

[Print view](#)
[Propose table structure](#)

Add  field(s)
 ☒ At End of Table
 ☐ At Beginning of Table
 ☐ After DealID
 [Go](#)

Indexes: ?

Keyname	Type	Cardinality	Action	Field
PRIMARY	PRIMARY	3	 	DealID
UserName	INDEX	1	 	UserName

Create an index on  columns [Go](#)

Server: localhost Database: a4117012\_LD Table: feedbacks

[Browse](#)
[Structure](#)
[SQL](#)
[Search](#)
[Insert](#)
[Export](#)
[Import](#)
[Operations](#)
[Empty](#)
[Drop](#)

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	<u>DealID</u>	mediumint(9)			No			
<input type="checkbox"/>	<u>UserName</u>	varchar(30)	latin1_general_ci		No			
<input type="checkbox"/>	<u>Title</u>	varchar(30)	latin1_general_ci		No			
<input type="checkbox"/>	<u>Description</u>	varchar(50)	latin1_general_ci		Yes	NULL		
<input type="checkbox"/>	<u>Rating</u>	float			No			
<input type="checkbox"/>	<u>Date</u>	int(11)			No			

[Check All](#) / [Uncheck All](#) With selected:

[Print view](#)
[Propose table structure](#)

[Add](#) 1 field(s)
 ☒ At End of Table
 ☐ At Beginning of Table
 ☐ After DealID
 [Go](#)

**Indexes:** ?

Keyname	Type	Cardinality	Action	Field
PRIMARY	PRIMARY	4		DealID UserName
UserName	INDEX	2		UserName
DealID	INDEX	2		DealID

Create an index on 1 columns [Go](#)

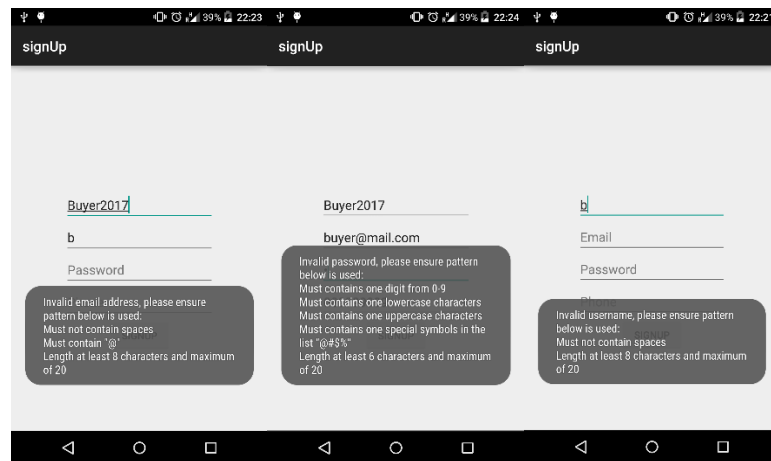
### 2.3.2 Responsive design:

Was achieved by following the advices and suggestions on the Google developers' platform like:

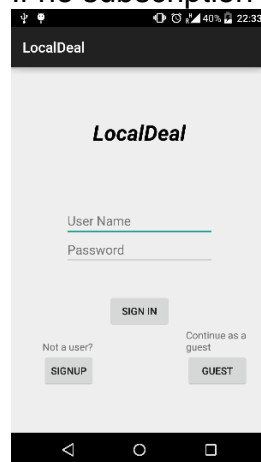
- Using density independent pixels whenever possible to ensure consistent layout across devices with different screen resolutions.
- Using wrap contents for layout dimensions.
- Avoiding hard coded pixel values in my code.
- Using relative layout mainly for all activities for positioning of child views.
- Testing application on multiple devices with different screen sizes and resolutions.

### 2.3.3 Features implementation:

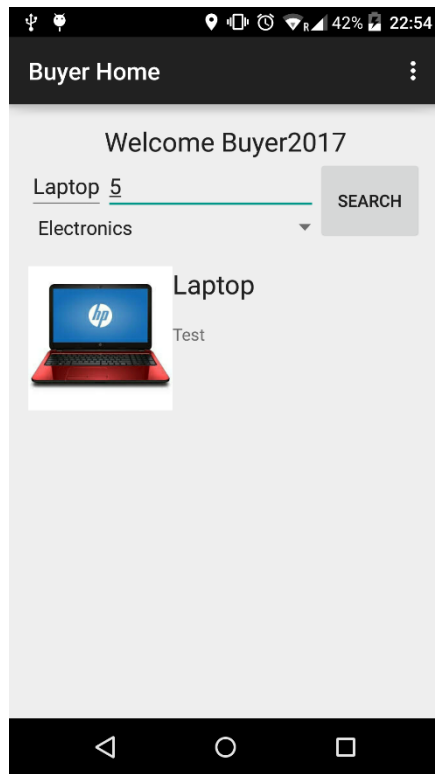
- Signup:
  - Same signing up steps for both buyers and sellers
  - Signup is handled solely by the app
  - Validation of name, phone, email and password rules is done during the signup process.



- Guest user:
  - This option is available for users who want to serve the application with signing up
  - It provides similar functionality like a normal user but without the option of providing feedback
- Main activity:
  - Main activity provides sign in facility to the user.
  - If the user has a valid subscription attached to their account, It will open the seller page
  - If no subscription available, buyer interface will be opened

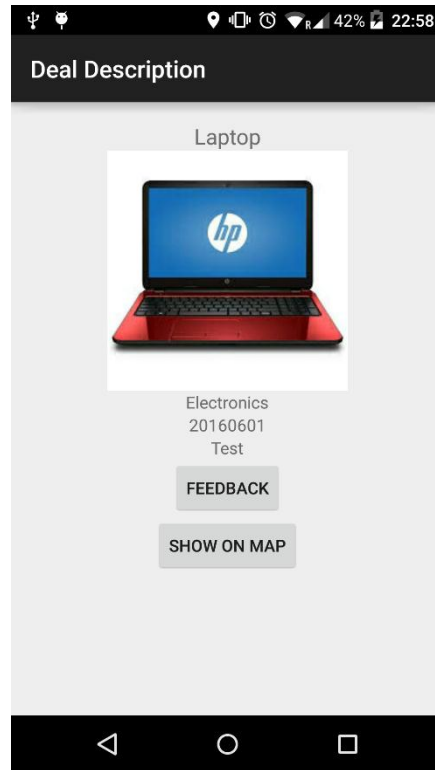


- Buyer Home:
  - It provides the buyer with three fields, search word, category and search distance.
  - Once the user clicks search, the application makes http request to the PHP file responsible for this action on the server which is in turn executes a sql query and send the data back to the application in form of JSON.
  - Data gets fetched in a list view using an adapter.

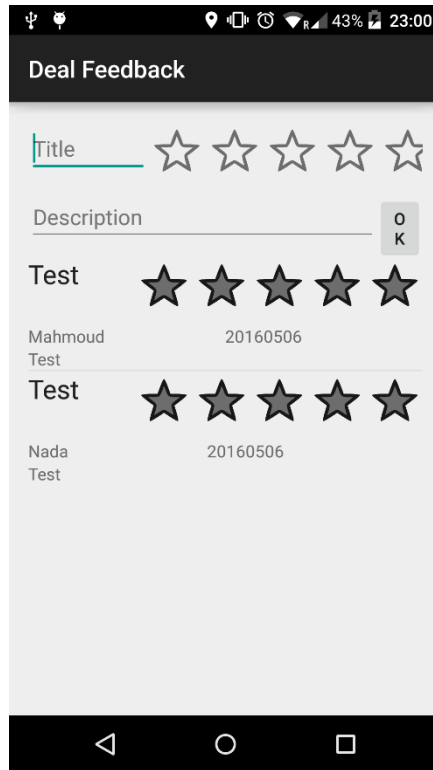


- Deal description:
  - Triggered by clicking on a deal from the list
  - Provides detailed information on the deal
  - Provides a link to deal's feedback
  - Provides an option to view the deal on the map





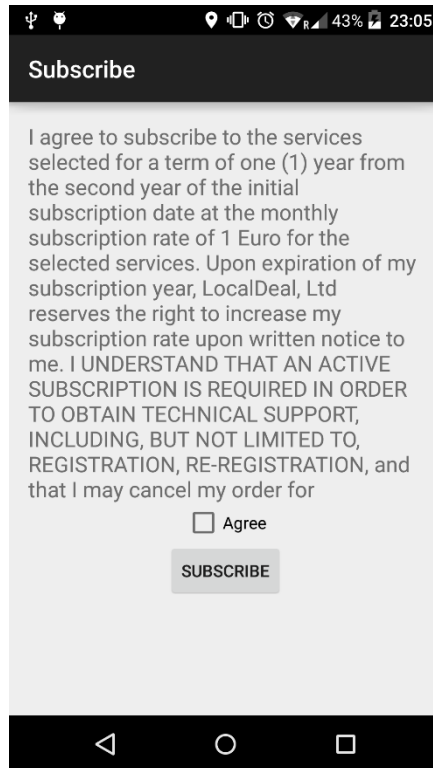
- Deal feedback:
  - Provides an option to enter new feedback
  - Provides a list of existing feedback



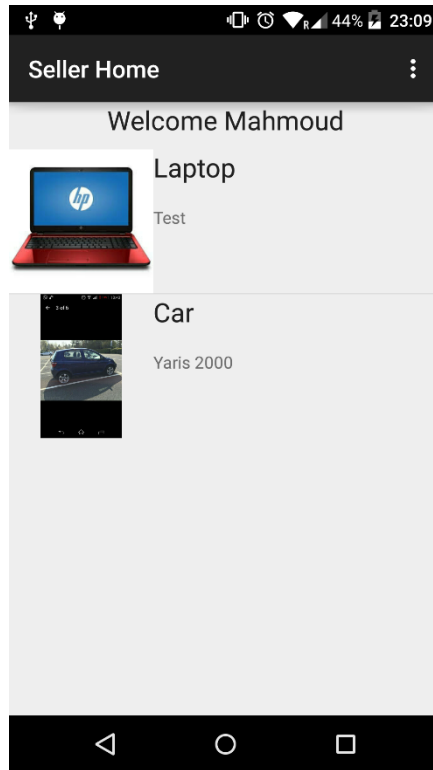
- Show on map:
  - Show the user the deal location on map
  - Implemented using Google maps, Google maps API, Google play services



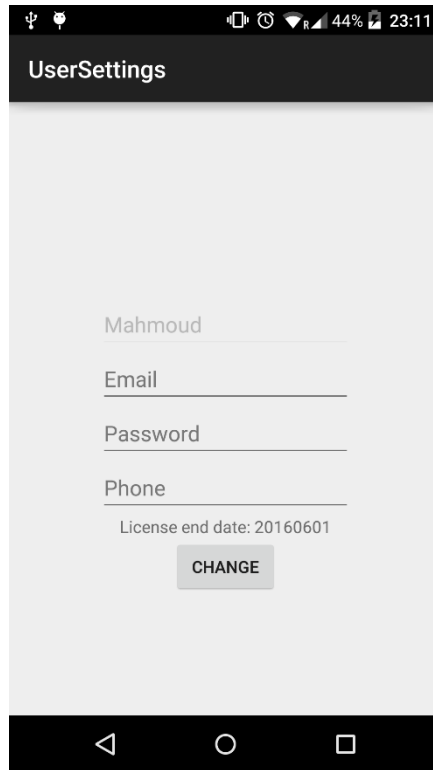
- Subscribe:
  - Allow any user to obtain a free subscription for the first year
  - Subscription details is entered into the database



- Seller home:
  - Allow sellers to view their deals in a list



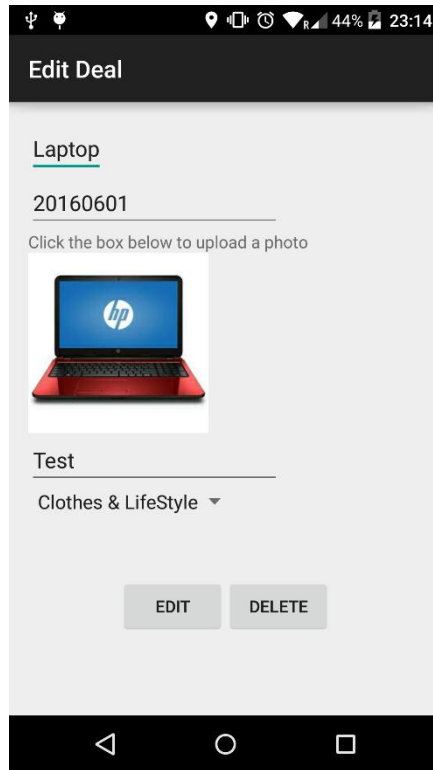
- User settings:
  - Allow the user to change their details and view subscriptions end date



- Post deal:
  - Allows the user to post deal

The screenshot shows the 'PostDeal' screen of a mobile application. At the top is a black header with the text 'PostDeal'. Below the header, the form contains the following elements: a text input field labeled 'Title'; a date input field labeled 'Expiry date: YYYYMMDD'; a text prompt 'Click the box below to upload a photo' above a large, empty square box; a text input field labeled 'Description'; a dropdown menu currently showing 'Clothes & LifeStyle'; a button labeled 'ENTER ADDRESS MANUALLY'; and a button labeled 'POST DEAL'. The bottom of the screen features a black navigation bar with three white icons: a back arrow, a circle, and a square.

- Edit deal:
  - Triggered by clicking on a deal in the seller's list
  - Allows editing deal details
  - Allows deleting a deal



- Address finder:
  - Allows seller to enter address manually if they do not want address to be determined using GPS



Address Finder

Address

City

County

Country

FIND ADDRESS

VERIFY ON MAP

Addresses found ☐ Use

USE THIS ADDRESS

### 2.3.4 Deployment:

- Database is hosted on 000webhost server
- PHP files are stored on the server side
- Data transfer is done using JSON
- HTTP calls are made from the application
- Deal pictures are stored on the server with reference of each picture in the deals table of the database

### 2.3.5 Problems solved:

- One of the problems I faced while working on this project was to retrieve multiple records from the database in the form of a JSON array and then parse this array and cast it into a java class. The problem was that PHP returns JSON in the form of an Array but to parse it in Java, the parent element has to be a JSON object and the child element has to be the JSON array. After many days of research and tweaking, the only way to do was by simply changing the returned String and to be able to parse it. Please find a snippet below.

```
result = '{ ' + "\"Deals\"" + ':' + sb.toString() + ' }';  
  
JSONObject parentObj = new JSONObject(result);  
JSONArray array = parentObj.getJSONArray("Deals");  
deals = new ArrayList<>();  
  
for (int i = 0; i < array.length(); i++) {  
    JSONObject jsonobject = array.getJSONObject(i);  
    Deal deal = new Deal();  
  
    deal.setDealID(jsonobject.getString("DealID"));  
    deal.setUsername(jsonobject.getString("UserName"));  
    deal.setTitle(jsonobject.getString("Title"));  
    deal.setExpiryDate(jsonobject.getString("ExpiryDate"));  
    deal.setDescription(jsonobject.getString("Description"));  
    deal.setPicture(jsonobject.getString("Picture"));  
    deal.setCategory(jsonobject.getString("Category"));  
    deal.setLongitude(jsonobject.getString("Longitude"));  
    deal.setLatitude(jsonobject.getString("latitude"));  
  
    deals.add(deal);  
}
```

- Another challenge was to fetch the returned result into a listview. I had to write a customised adapter to be able to achieve the required result. Please find a snippet below.

```
public class DealAdapter extends ArrayAdapter{

    private List<Deal> deals;
    private int resource;
    private LayoutInflater inflater;

    public DealAdapter(Context context, int resource, List<Deal> objects) {
        super(context, resource, objects);

        this.deals = objects;
        this.resource = resource;
        inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        if(convertView == null){
            convertView = inflater.inflate(resource, null);
        }

        ImageView dealIcon;
        TextView dealTitle;
        TextView dealDescription;

        dealIcon = (ImageView)convertView.findViewById(R.id.dealIcon);
        dealTitle = (TextView)convertView.findViewById(R.id.dealTitle);
        dealDescription = (TextView)convertView.findViewById(R.id.dealDescription);

        dealTitle.setText(deals.get(position).getTitle());
        dealDescription.setText(deals.get(position).getDescription());
        try {
            Bitmap image = new DownloadImage(deals.get(position).getPicture()).execute
                ().get();
            dealIcon.setImageBitmap(image);
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (ExecutionException e) {
            e.printStackTrace();
        }

        return convertView;
    }
}
```

- Another challenge was to enable the user to upload an image of the deal, encode this image into a bitmap, upload it to the server and save its URI into the database for retrieving it later. Please find below a small part of the solution.

```

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
image.compress(Bitmap.CompressFormat.JPEG, 100, byteArrayOutputStream);
byte[] imageBytes = byteArrayOutputStream.toByteArray();
encodedString = Base64.encodeToString(imageBytes, Base64.DEFAULT);

```

```

<?php

header('Content-type : bitmap; charset=utf-8');

if(isset($_POST["encoded_string"])){

    $encoded_string = $_POST["encoded_string"];
    $image_name = $_POST["image_name"];
    $username = $_POST['username'];
    $expiryDate = $_POST['expiryDate'];
    $description = $_POST['description'];
    $category = $_POST['category'];
    $longitude = $_POST['longitude'];
    $latitude = $_POST['latitude'];

    $decoded_string = base64_decode($encoded_string);

    $path = 'pictures/' . $username . $image_name . $expiryDate . '.JPG';

    $file = fopen($path, 'wb');

    $is_written = fwrite($file, $decoded_string);

    fclose($file);

    if($is_written > 0){

        $con = mysqli_connect("localhost", "root", "localdeal");
        $query = "INSERT INTO deals (UserName, Title, ExpiryDate, Description, Picture, Category,
        $result = mysqli_query($con, $query);

        if($result){
            echo 'Success';
        }else{
            echo 'Failed';
        }

        mysqli_close($con);
    }
}

```

- Calculating the distance between the user and the deal was achieved by using a sample code provided by (Anon., n.d.) that calculates the distance between two coordinates and tweaking it so It can return deals based on their distance from buyers.

```

private static double distance(double lat1, double lon1, double lat2, double lon2, String unit) {
    double theta = lon1 - lon2;
    double dist = Math.sin(deg2rad(lat1)) * Math.sin(deg2rad(lat2)) + Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2))
    dist = Math.acos(dist);
    dist = rad2deg(dist);
    dist = dist * 60 * 1.1515;
    if (unit == "K") {
        dist = dist * 1.609344;
    } else if (unit == "N") {
        dist = dist * 0.8684;
    }

    return (dist);
}

private static double deg2rad(double deg) {
    return (deg * Math.PI / 180.0);
}

private static double rad2deg(double rad) {
    return (rad * 180 / Math.PI);
}

```

## 2.4 Testing

Since testing plays an important role in the development of any application, it was on top of my priorities since the beginning.

### 2.4.1 Unit Testing:

Was done at each step of development and after implementing every feature. It was done by basically run the application on my phone and monitor its behavior, testing every input and output.

### 2.4.2 Integration testing:

Is conducted to make sure that all components of the application are integrated properly and data is passed from one component to another as expected.

### 2.4.3 Backward compatibility testing:

Was done by testing the application on previous Android versions and ensure same behaviour is obtained

#### **2.4.4 User acceptance testing:**

Was done by getting colleagues and friends to test the application and gathering their feedback about usability and application functionalities whether there are dead links, exceptions, crashes or unexpected outputs.

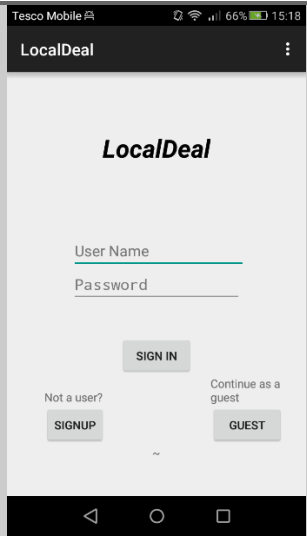
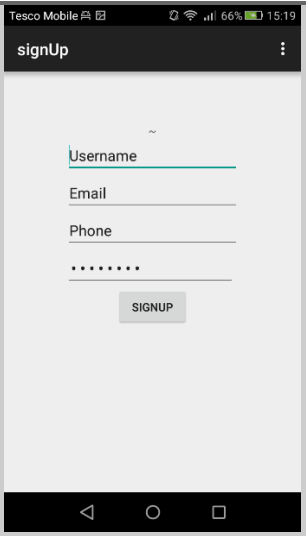
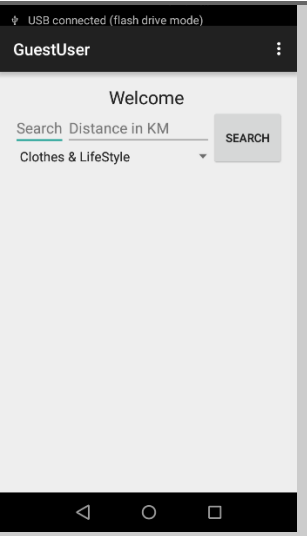
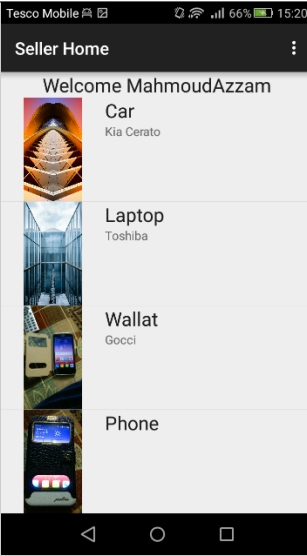
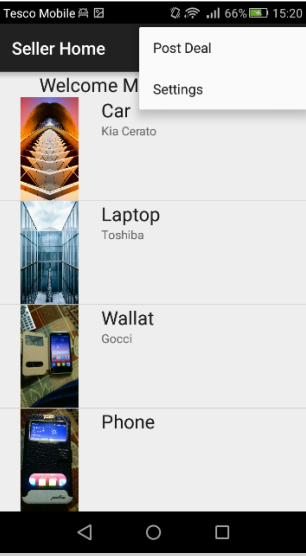
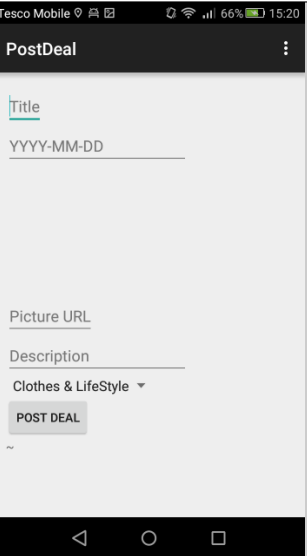
#### **2.4.5 System testing**

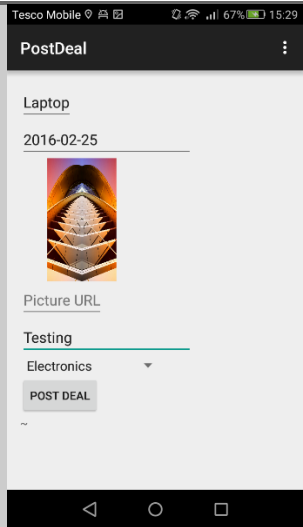
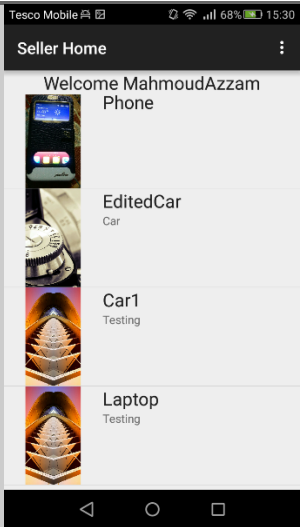
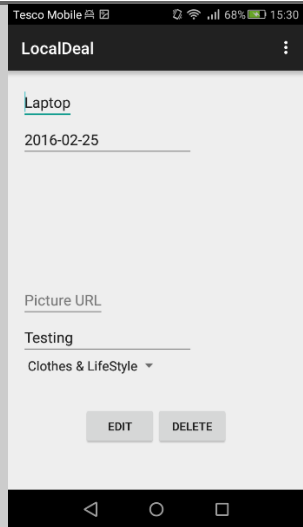
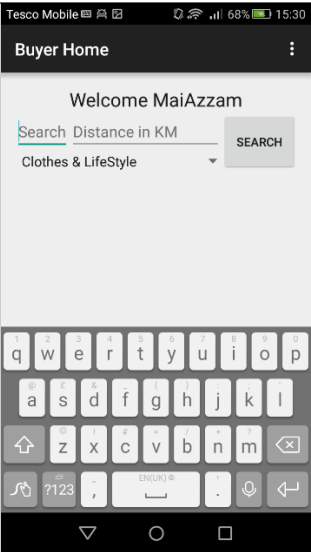
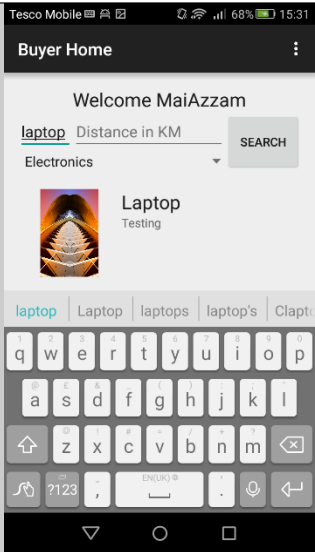

Has been done on the current version of LocalDeal to ensure it compliance with the initial specified requirements. Types of system testing performed:

- GUI testing: Done by testing every input and output of the application and ensuring that every field and button are working as designed.
- Performance testing: Done by testing the application performance by measuring the time taken to respond to user actions, skipped frames and make sure it is within the acceptable standards.
- Security testing: Done by testing different inputs to each field and make sure that validation is working as designed.
- Usability Testing: Done by getting different users to test the application from different backgrounds and ages to ensure that the application is easy to use and understand.
- Installation testing: Done by testing the final APK of the application and installing it on different devices with different operating system, bandwidth to make sure the size of the application is within the normal limits and that users with different operating systems face no issues during the installation.

## 2.5 Graphical User Interface (GUI) Layout

Provide screenshots of key screens and explain.

Initial login	Sign up	Guest User
		
Seller deals	Seller to post deal	Post deal interface
		

Posting a deal	View new deal	Edit or deleted deal
		
Buyer initial page	Search a deal	Show deal on map
		



## **2.6 Evaluation**

I am always evaluating the system performance and speed by running the application on different devices and make sure that performance is not lagging on of them. I always monitor database and test that information being written there in an adequate time.

## 3 Conclusions

### 3.1 *Advantages of LocalDeal*

- Is to enable local businesses from reaching out to the local customers, gain some market share, advertise their good deals and offers, lower their marketing costs in an easy and simple way. Also
- Is to allow local buyers to locate good deals and offers in their local neighbourhood. Find what they are looking for not what advertisers are suggesting them. Spend less by getting good deals without the need to drive far to big shops. Stay up to date with the latest deals offered by their local businesses.
- It will help communities keeping local jobs and keep their local businesses open and thriving.

### 3.2 *Disadvantages of LocalDeal*

- Only available for Android users
- It requires an Android phone for both sellers and buyers to use the application
- It needs sellers to post appealing deals to buyers

### 3.3 *Market analysis:*

- The idea behind **LocalDeal** has not been implemented before, some features are available by Google ads, Facebook ads, Donedeal, etc however there is no available application on the market that focus on enabling local businesses and empower them and buyers without collecting lots of their personal data.
- The closest idea to LocalDeal is a software called “Pointy” that allows shops to update their products’ database on the internet by scanning each products using special equipment provided by Pointy during the sale. By doing so, users can search the pointy database and find accurate information on available products within a specified area. Comparing it to LocalDeal, it requires hardware installation, costly equipment, training of users and disclosure of sensitive data of the subscribed businesses which is not the case with LocalDeal.

## 4 Further development or research

**LocalDeal** is a mature business idea that does not have much competition at the moment, sounds appealing to both local businesses and buyers.

With more resources, more features could be implemented like:

- Push notifications of surrounding offers while buyer is close by.
- Offer local businesses to have store fronts on **LocalDeal**
- Buyers would be able to reserve an offer and pay for it
- An Iphone application to be developed

## 5 Bibliography

Anon., n.d. *Sample Codes (Java)*. [Online]  
Available at: <https://www.geodatasource.com/developers/java>  
[Accessed 4 2 2016].

## 6 Appendix

### 6.1 Project Proposal

#### 1. Objectives

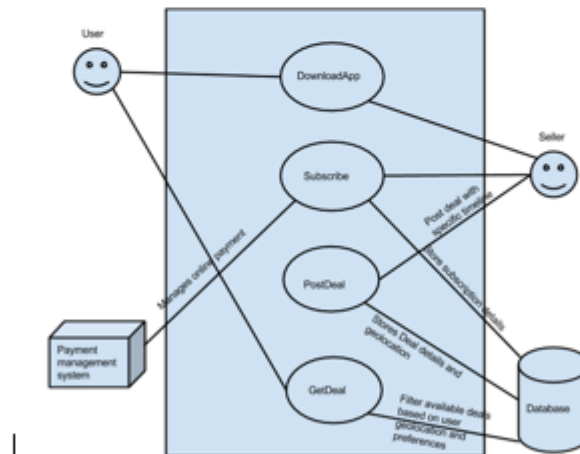
Application is to assist buyers locating bargains, deals, discounts within close distance of their location (1km, 5km, 25 km) and assist sellers targeting immediate nearby potential buyers.

#### 2. Background

For long time we been receiving ads via post, email, etc. Advertisers try to broadcast their products in every possible way however in many cases their ads just cause annoyance to their audience. Ads not always useful for receivers or satisfies their requirements. Not every small shop or company can pay for ads premiums and if they managed to do so, the life time of the ad is very limited and difficult to keep it up to date. 'LocalDeal' will allow local shops and companies reach out to potential customers who really have interest in their products. 'LocalDeal' will help potential buyers find deals, discounts and offers which they have interest in without being surrounded with irrelevant ads.

#### 3. Technical Approach

Use Case:



OS targeted:

The app will be available to Android users in the first place however Iphone app is a must.

**Revenue generated:**

The app will be subscription-based. App will be free to buyers and small monthly subscription will be charged for sellers. Subscriptions will be segmented per number of Ads a seller can post per month. Trial month for every subscriber.

**Market:**

The App is not bound to a specific market however the focus will be on the Irish market at the beginning.

**Login:**

- Users will not be required to create an account or post any of their details
- Sellers will be required to create an account for payment and subscriptions management reasons

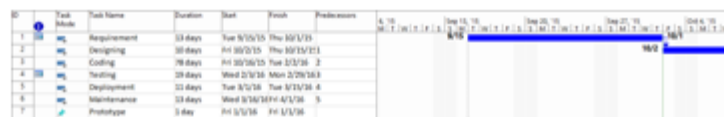
**Marketing:**

- FB page to be created
- One month trial for any seller to promote the application

## 4. Special resources required

App will be developed mainly in Android Studio. Android device will be used for development. MySQL database will be created to hold subscribers details.

## 5. Project Plan



## 6. Technical Details

- Language: Java
- Database: MySQL
- IDE: Android studio
- Version Control: GIT
- Libraries: `android.location`

## 7. Evaluation

Different tests will ~~takeplace~~ including Unit Testing, Integration testing and System testing. Testing will ~~takeplace~~ throughout the development cycle using ~~Android~~ virtual and physical devices. Real users will participate in testing the application. Feedback will be collected after the prototype is available.

## 6.2 Project Plan

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	6, '15	Sep 13, '15	Sep 20, '15	Sep 27, '15	Oct 4, '15
1	■	Requirement	13 days	Tue 9/15/15	Thu 10/1/15		M	9/15			
2	■	Designing	10 days	Fri 10/2/15	Thu 10/15/15	1	T				
3	■	Coding	78 days	Fri 10/16/15	Tue 2/2/16	2	W				
4	■	Testing	19 days	Wed 2/3/16	Mon 2/29/16	3	T				
5	■	Deployment	11 days	Tue 3/1/16	Tue 3/15/16	4	F				
6	■	Maintenance	13 days	Wed 3/16/16	Fri 4/1/16	5	S				
7	■	Prototype	1 day	Fri 1/1/16	Fri 1/1/16		M				

## 6.3 Monthly Journals

### September

#### Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: September

#### My Achievements

This month, I was able to ...

- Setup my coding environment
- Setup my AVD (Android virtual device) and real device
- Examine competitors
- Determine weak points and strength points
- Learn Android core structure
- Sketch the layout and main flow

#### My Reflection

I examined all the available IDEs for developing Android apps. I found that most of the available videos and books are using Eclipse however it was announced that it will be discontinued. I found that Android studio will be the main IDE for Android development offered by Google. I chose to develop in Android studio however I faced some challenges to find enough learning resources using the same IDE.

I struggled in setting up the Android virtual device due to the Intel virtualization technology and HAXM. I managed to get it working eventually. I struggled to setup the real device due to lack of driver for my device but I managed to do it at the end.

Before starting the project, There was not on the market any similar applications however recently an application been released which is called "Shpock". The application implements my idea but in a different way. My idea is for any advertisers to broadcast their offers and deals to potential buyers in their surroundings. Shpock is for users to sell second hand stuff within their surroundings.

I read a book about Android studio and watched different courses on "Pluralsight" about Android. I managed to learn the core structure of Android.

#### Intended Changes

Next month, I will try to get the database designed, get the layout designed. Start coding the main activities. Do some testing and get feedback about the UI and the database design.

#### Supervisor Meetings

Date of Meeting: Email sent asking for opinion, Meeting to be arranged.

Items discussed: Opinion about my idea.

Action Items: Reflective journals to be sent.

End of document ■

### October

## Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: October

### My Achievements

This month, I was able to ...

- Design the database
- Learn about the location services provided by Android
- Create a demo application in Android (Note taking app)
- Have a more detailed image of activities and classes needed
- Setup my Github repository

### My Reflection

I think I spent the right amount of time to get the database design right as it has a big role in my application. I tried to keep it simple as much as possible to avoid any complications in the future. I followed a course on Pluralsight that helped me creating a demo application about Android. The application I created was very important as I learned the core components of an Android application like activity, intent, service and view. I have enough understanding to help me start coding on my main application.

I found that it is important to know about the location services of Android since the beginning as it is a main part of my application idea so I followed another course to have a good understanding.

### Intended Changes

I realised that I need to ...

- Use PHP to connect to database in order to use MySql

Next month, I will try to

- Wire my database with the application using PHP.
- Start on the layout and styling
- Start coding.

### Supervisor Meetings

Date of Meeting: Meeting to be arranged

Items discussed: Database design to be discussed and overview on my progress

Action Items:

End of document ■

## November

## Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: November

### My Achievements

This month, I was able to

- Create a MySQL database
- Create PHP files to connect the database to LocalDeal application
- Setup a local server to test the database and PHP files
- Get a good grasp of the transmission of data between the server and the application
- Start creating the layout of the application using XML
- Start styling the layout created using XML
- Start coding the login activity

### My Reflection

I think I have a good understanding of the requirements to connect the application to the database using PHP. I managed to create the PHP pages required for the login. I setup the local server using WAMP to be able to test throughout the development process. Soon I will buy a domain and make sure that my database, PHP files and application works as it should be on a remote server. I created the layout of the main page and styled it using XML. I started creating a theme to be used throughout the application using XML. Started coding the login activity.

However, I managed to setup WAMP on my machine however I struggled a bit to configure it as needed.

### Intended Changes

Next month, I will try to get big part of user interface created. I will spend more time on coding the activities. I will start to prepare for the prototype.

I realised that I need to get the database and PHP files on a remote server to test my progress on a real production environment to avoid any issues at the final deployment.

### Supervisor Meetings

Date of Meeting: To be arranged

Items discussed: Progress to be discussed

Action Items: Focus on coding

End of document ■



## December

### Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: December

#### My Achievements

This month, I was able to

- Created the main activities
- Created the main java classes
- Created most of the PHP files needed
- Finished the login, sign up, subscribe, initial seller activity and initial buyer activity
- Wrote the code that enables the application to get the GPS coordinates

#### My Reflection

I managed to write the main java classes needed, understand how to use intents and move data across different activities. Managed to write the code to get the GPS coordinates.

However I struggled to save the GPS coordinates to the database but I managed to get around by saving the coordinates as a string and cast back to double whenever needed.

#### Intended Changes

Next month, I will get the seller to be able to post deals, view deals in a view list, edit deal and delete deal.

#### Supervisor Meetings

Date of Meeting: To be arranged

Items discussed: Progress to be discussed

Action Items: Focus on coding

End of document ■

## January

### Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: January

#### My Achievements

This month, I was able to

- Create the class that post deal to database along with the GPS coordinates of the seller.
- View seller's deals in a view list
- Save every deal's image to the server along with its URI to the database
- Create the class to edit deal and delete deal
- Create the class that get deals to buyer based on certain search criteria set by the buyer
- Parse returned JSON array and cast it to a deal object

#### My Reflection

This month was exceptional since I managed to achieve few milestones in getting the buyer and seller interfaces done. Now seller can post, view, edit and delete deals. Buyers can search and view deal and can locate them on the map.

I faced few challenges:

- Firstly, to retrieve multiple records from the database in the form of a JSON array and then parse this array and cast it into a java class. The problem was that PHP

returns JSON in the form of an Array but to parse it in Java, the parent element has to be a JSON object and the child element has to be the JSON array. After many days of research and tweaking, the only way to do was by simply changing the returned String and to be able to parse it.

- Secondly, to fetch the returned result into a listview. I had to write a customised adapter to be able to achieve the required result.
- Thirdly, to enable the user to upload an image of the deal, encode this image into a bitmap, upload it to the server and save its URI into the database for retrieving it later.

#### Intended Changes

Next month, I will implement the algorithm by which buyer can return deals based on their distance from him. I will work as on enhancing the layouts and styling them.

#### Supervisor Meetings

Date of Meeting: To be arranged

Items discussed: Progress to be discussed

Action Items: Focus on coding

End of document ■

## February

## Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: February

### My Achievements

This month, I was able to

- Getting ready for the mid-point presentation
- Getting most of the features to work
- Implementing an algorithm for returning deals based on their distance
- Perform system testing

### My Reflection

This month was exceptional since the mid-point presentation took place. I managed to get the project to partially finished state.

### Intended Changes

Next month, I will focus on deploying the application to a server and implement changes suggested by supervisor.

### Supervisor Meetings

Date of Meeting: 25/2/2016

Items discussed: Feedback after the mid-point presentation

Action Items:

- To allow seller to enter address manually possibly using something like eircode.ie
- Market analysis to be enhanced
- Responsive design to be taken into consideration

End of document ■

## March

## Reflective Journal

Student name: Mahmoud Azzam

Programme: BSc (Hons) in Computing - Evening

Month: February

### My Achievements

This month, I was able to

- Deploy PHP files and Mysql database to 000webhost
- Design the application icon
- Enhance the UI

### My Reflection

This month I managed to deploy the application which was crucial for any further development, I managed to design the application icon which reflects the application purpose and goal.

### Intended Changes

Next month, I will focus styling the UI taken into consideration the responsive design concepts.

### Supervisor Meetings

Date of Meeting: 14/3/2016

Items discussed: Action plan to finish the remaining parts of the project, Icon's design

Action Items:

- To allow seller to enter address manually possibly using something like eircode.ie
- To change the colours of the Icon

End of document ■