

PROJECT REPORT

ON



AT



SUBMITTED IN FULFILLMENT FOR AWARD OF DEGREE

IN

Master of Computer Applications

(BATCH 2010-2013)

SUBMITTED BY

(Sample)



Atmiya Institute of Technology and Science

Gujarat Technological University.

Certificate:

Acknowledgment:

To get the practical knowledge and to make a report on a certain topic without anybody's help is really a task of great difficulty. Fortunately in preparation of this report i got the help of many known as well as unknowns for providing the valuable information about each and every stage of this project.

I am thankful to all those persons because the activity if going through industrial orientation it'll bridged me to become good IT Professional. Naturally, it requires lot of people support to complete this project. We take this opportunity to acknowledge their support to us.

First of all I am extremely thankful to M.C.A (Master of Computer Application) department of Atmiya Institute of Technology & Science, Rajkot for giving permission to do project in outside of campus.

I consider myself privileged having found an opportunity to express our heartily thanks to our head of the M.C.A. department.

The success of any project is never limited to the individual undertaking the project. However, there are some key personalities whose role is very vital in project. So i greatly obliged to express my gratitude to all the mentioned officials of "**Kevalam Software**" for giving me assistance, guidance, inspiration in this project.

I would like to thank Mr. Saral Doshi and Mr. Amit Santoki, "Kevalam Software" for giving me permission to do a project in his company. I will always remain grateful to him and whose constant care about me has provided a new direction to work.

With sincerely regards,

ABC

Table of Contents

1. Company Profile:	1
2. Project Profile:	2
2.1 Project Definition:	2
2.2 System requirements:	3
2.2.1 Project Profile:	3
2.2.2 Project Tools:	3
2.2.3 Software Requirements:	3
2.2.4 Hardware Requirements (Recommended):	3
3. Analysis:.....	4
3.1 About existing System:	4
3.2 Feasibility Study:.....	4
3.2.1 Operational Feasibility:.....	4
3.2.2 Technical Feasibility:.....	4
3.2.3 Economical Feasibility:	4
3.3 Limitation of existing system:.....	5
3.4 Scope of Proposed System:	5
3.5 Limitation of proposed system:.....	5
3.6 Static View:	5
3.6.1 What is object diagram?.....	5
3.7 Dynamic View:	6
3.7.1 What is use case?	6
4. Design:.....	11
4.1 Diagrams:	11
4.1.1 Sequence Diagram:	11
4.1.2 Activity Diagram:	13
4.1.3 Class Diagram:	18
4.2 Data Dictionary:	20
5. Implementation and Testing:.....	26

5.1	Input Event:	26
5.2	Report Layout:.....	40
5.3	Testing:.....	41
5.3.1	Testing Plan:	41
5.3.2	Testing Strategy:	41
5.3.3	Testing Methods:	43
5.3.4	Test Cases:	44
5.3.5	Future Enhancements:.....	45
6.	About the Tools & Technology:.....	46
6.1	Develop for Android:	46
6.1.1	Android Architecture:	47
6.1.2	Applications:.....	47
6.1.3	Development Toolkit:	49
6.2	About EllisLab:	52
6.2.1	History:	52
6.2.2	Our Business Model:.....	52
6.2.3	Our Customers:	52
6.2.4	What is CodeIgniter?	52
6.3	Model View Controller (MVC):	53
6.4	MySQL:.....	54
7.	Learning during Project Training:	55
7.1	List of such task:	55
8.	Conclusion:	56
9.	Role Statement:	57
10.	References:.....	58

1. Company Profile:

2. Project Profile:

2.1 Project Definition:

In single line **Couponicus** define as to convert your **Facebook** fans, **Twitter** followers and **Internet** users into your Customers. The main vision of this application is to build a brand around transparency and trust.

Local businesses invest precious time and money into advertising, building a website, gathering social media followers, and collecting email addresses. We believe you should always understand the true returns on any of these investments. How many customers do you earn for every dollar that you spend? Our goal is to automate the creation and distribution of content that drives sales, and then give you insight into the analytics that really matter: customers.

We're a team of passionate engineers, problem-solvers and coffee aficionados working around the clock to simplify the way you market your business online and drive customers into your locations.

We love working with our small business partners all across the country. Couponicus has helped independent and multi-unit restaurants, clothing stores, coffee shops, massage studios, adventure companies and more. Join other business owners across the country and use Couponicus to get the most out of your online marketing.

This Android application is a great way to reach customers, gain new ones and expand our brand through social networks.

It facilitates to display all the details about offers which are created by promoter. Hear old promoters can login and new promoters can create him/her account.

This application displays different types of offers like Live Campaign, Pending offers, recently run, Draft and shows three types of results like Total customer gained, Source of customer, Result by campaign.

In this application you can edit offers title and description and also edit start date and end date of particular offer. Here you can also publish the offers which are pending, **end** the offers which are currently live and delete the offers.

Couponicus provides facility to edit Business Details like business name, business category and here you can also change your email id and password.

Do not worry if you forgot your password, from this application you can send link of reset password to your mail address. It is better option for logout application, it is also provided by this application.

2.2 System requirements:

2.2.1 Project Profile:

Project Title:	
Organization:	
Developed by:	
Duration:	
Internal Guide:	

2.2.2 Project Tools:

Front-end Tools:	Eclipse
Back-end Tools:	MySQL Server
Web Service:	PHP codeigniter framework
Platform:	Android
Others:	Achartengine library for Chart

2.2.3 Software Requirements:

Application Tools:	Front-end	Eclipse Helios (with ADT plug-in Android 2.1)
Database Back-end Tools:		MySQL Server
Office Automation Tools:		Microsoft office 2007, MS Visio

2.2.4 Hardware Requirements (Recommended):

Application Installation	Android enabled GSM/CDMA phone
Internet Connection	Required
Version	2.1
Memory	128MB

3. Analysis:

3.1 About existing System:

Couponicus is available in market. Its contains features such as register new promoter, login for old promoter, display different types of offer and also display different types of chart about customer. It is also facilitates to change all details about business profile and user profile.

3.2 Feasibility Study:

3.2.1 Operational Feasibility:

By feasibility study we mean study of the current operational systems and brief consideration of alternative methods computerizing these tasks. The purpose of the feasibility study is to investigate the present system, evaluate the possible application of computer-based methods, select a tentative system, evaluate the cost and effectiveness of the proposed system, evaluate the impact of the proposed system on existing personal and ascertain the need for new personnel.

From the user point of view, a Couponicus here is easy to operate because it only uses simply step to by step to perform any task. The application simple for smart phone user. So it no needs to give any instruction how to use these systems. There is no need to learn any special command and training to use this system.

3.2.2 Technical Feasibility:

Technical feasibility refers to the ability of the process to take advantages of the current state of the technology in pursuing further improvement. The front end tool proposed is easily compatible with the current hardware configuration in the organization. The back end tool proposed has the capacity to hold the data required for using the new system.

Couponicus uses the existing technology only. It uses internet for sending link of new register user and forgot password. Now a day these technologies are improved day by day. So it's convenient to us use these technology. So there is no need to develop any hardware to provide these facilities. Couponicus is run on android operating system which is available in most of smart phone devices. So there is no need to install any software or driver to run this application.

3.2.3 Economical Feasibility:

For declaring that the system is economically feasible, system will be cost effective and budgetary constraints, it should be cheap and quick. There isn't any extra requirement of peripheral or software for development of system as it can be completed with the available resource.

3.3 Limitation of existing system:

The current application is available in market is use only for displaying offer details and results related to customer. To create the new offer and other details, you have to visit official website.

3.4 Scope of Proposed System:

Couponicus application is mainly useful to promoters to view the details about customers and offers. Promoter can see all the offers and results but he can create offers from online site.

Email is now general use service in smart phones devices. The technical or non technical person both are use email services in its daily life. So we can use proper use of email and its feature. Gmail provided by Google now widely used email service in the world. In this application we are using Gmail for link of new register promoter and link of forgot password.

3.5 Limitation of proposed system:

This project only works on devices, which have android operating system.

If internet is disabled then promoter cannot use this application so Internet connection is required.

3.6 Static View:

3.6.1 What is object diagram?

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams.

Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams are used to render a set of objects and their relationships as an instance.

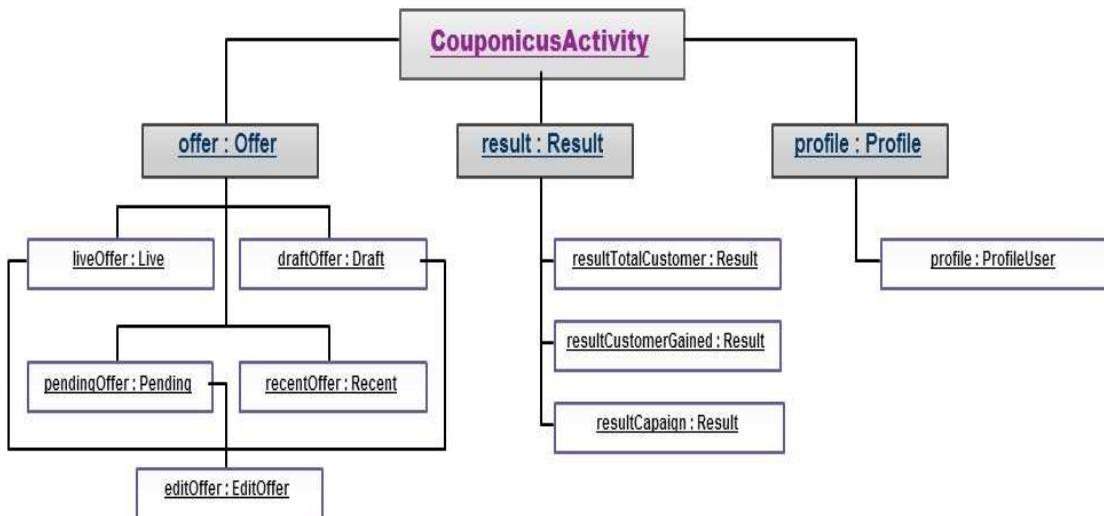
The purpose of a diagram should be understood clearly to implement it practically. The purposes of object diagrams are similar to class diagrams.

The difference is that a class diagram represents an abstract model consisting of classes and their relationships. But an object diagram represents an instance at a particular moment which is concrete in nature.

It means the object diagram is more close to the actual system behavior. The purpose is to capture the static view of a system at a particular moment.

So the purpose of the object diagram can be summarized as:

- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.
- Understand object behavior and their relationship from practical perspective



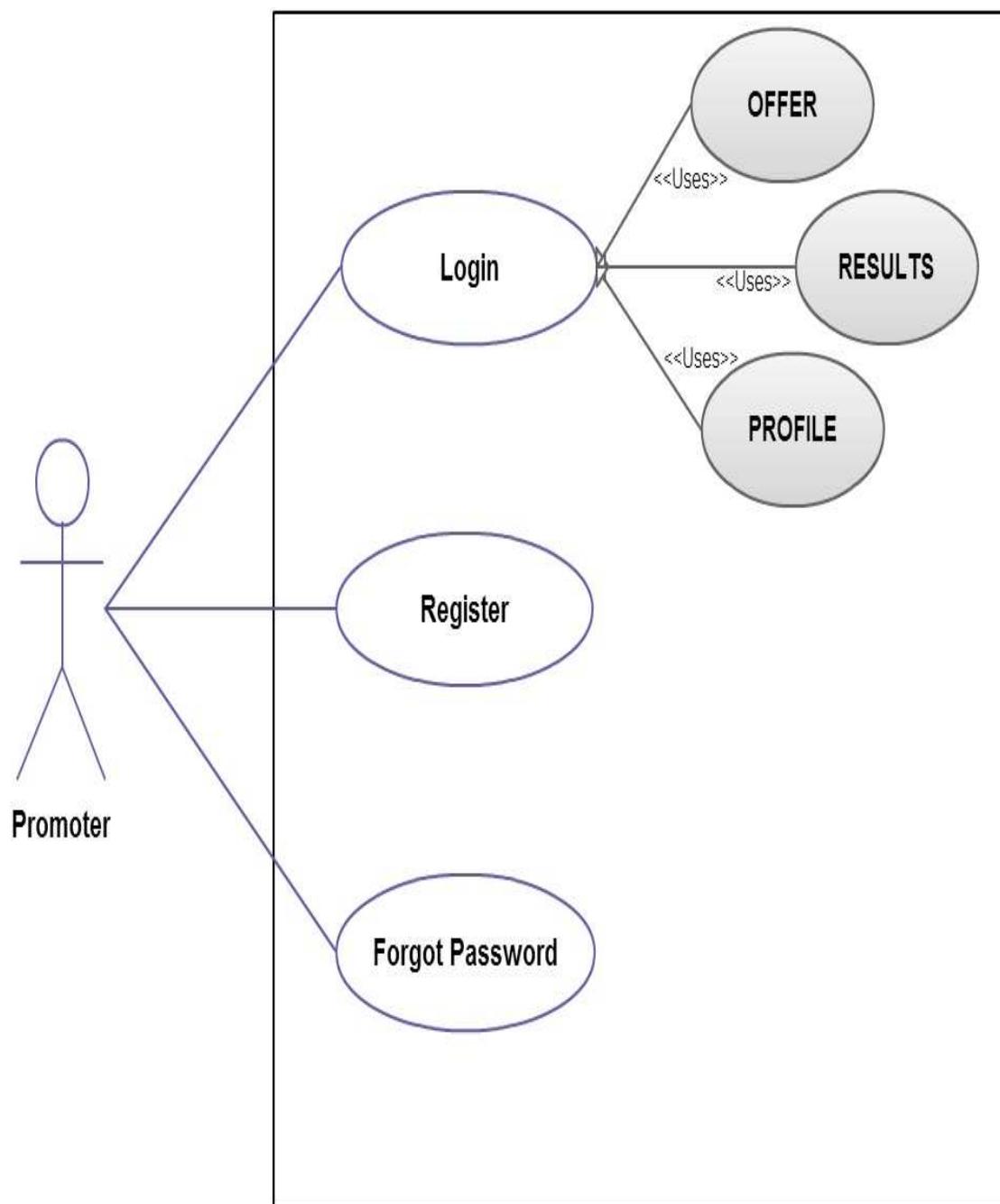
3.7 Dynamic View:

3.7.1 What is use case?

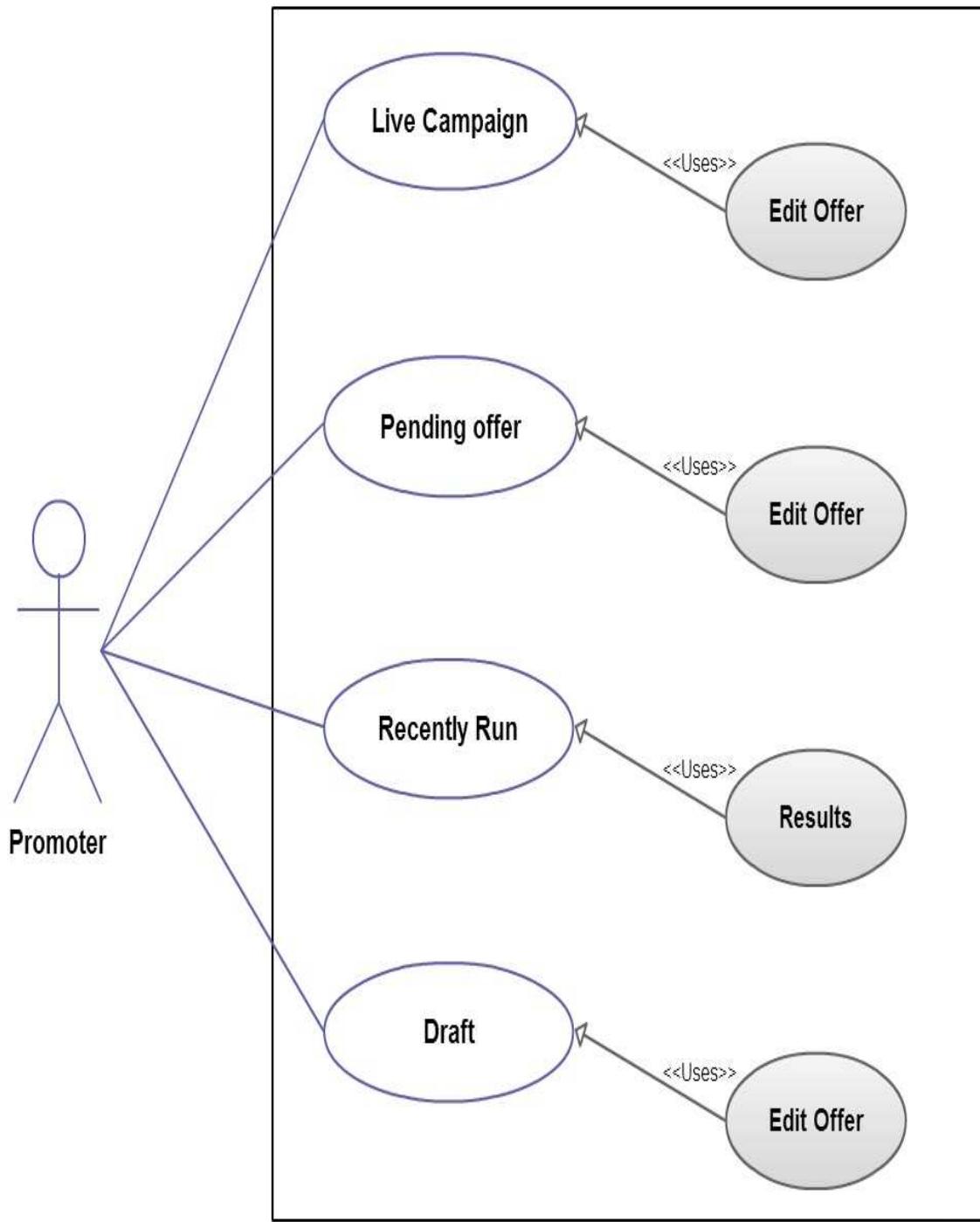
The Use Case diagram models the user's expectation for using the system. The people and systems that interact with the target system are called actors. The features of the system that the actors use are called use cases. Some use cases interact with other use cases, a relationship modeled using dependency arrows.

The goal of the Use Case diagram is to identify all the features that the clients expect the system to support, but it does not reveal any details about the implementation of these features.

- Use Case diagrams are valuable because they
- Identify the clients' expectations for the system.
- Identify specific features of the system.
- Identify shared behavior among system features.
- Provide a simple and easily understood way for clients to view their requirements.

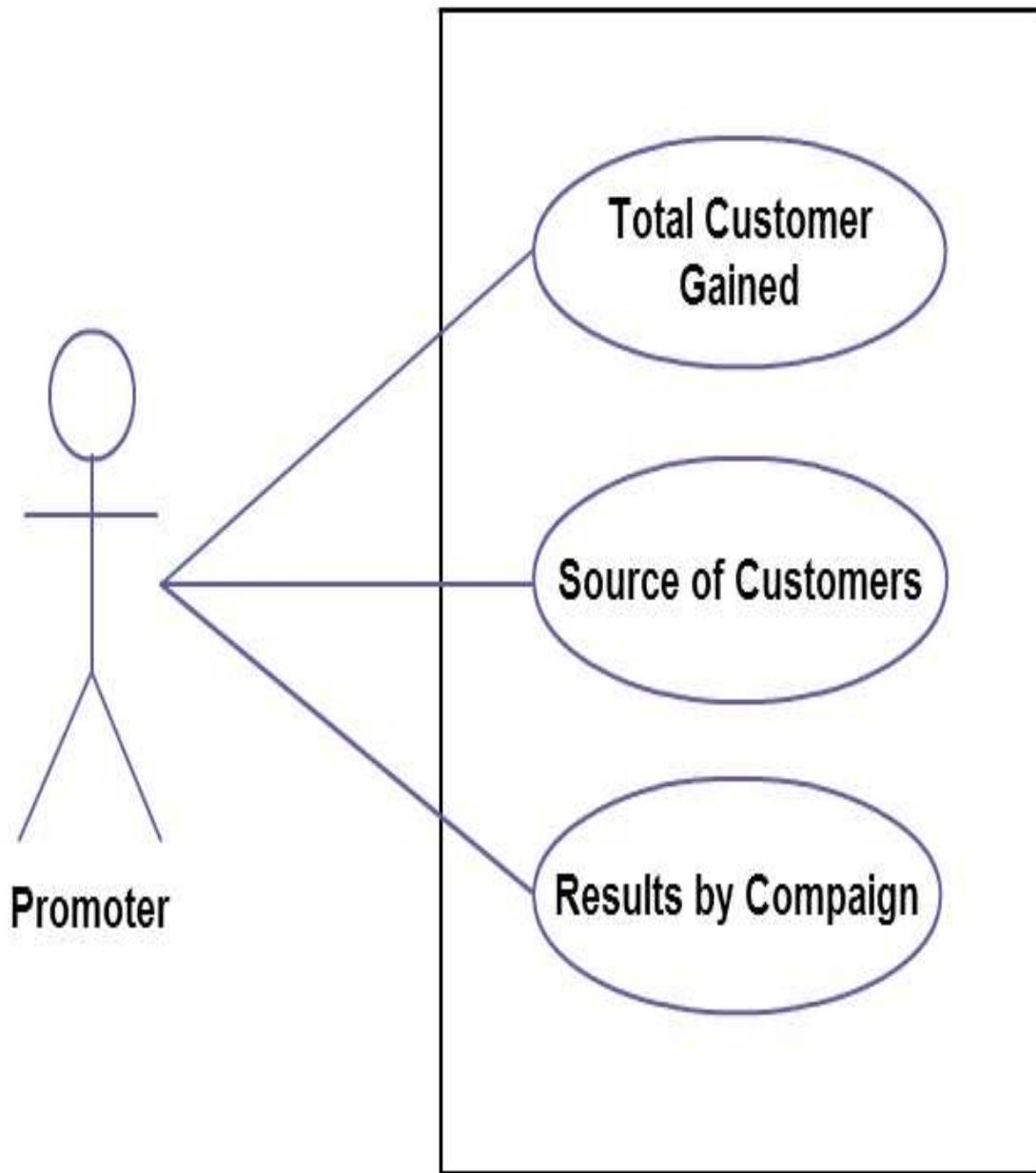
3.7.1.1 Home Screen:

3.7.1.2 Offer Tab Screen:

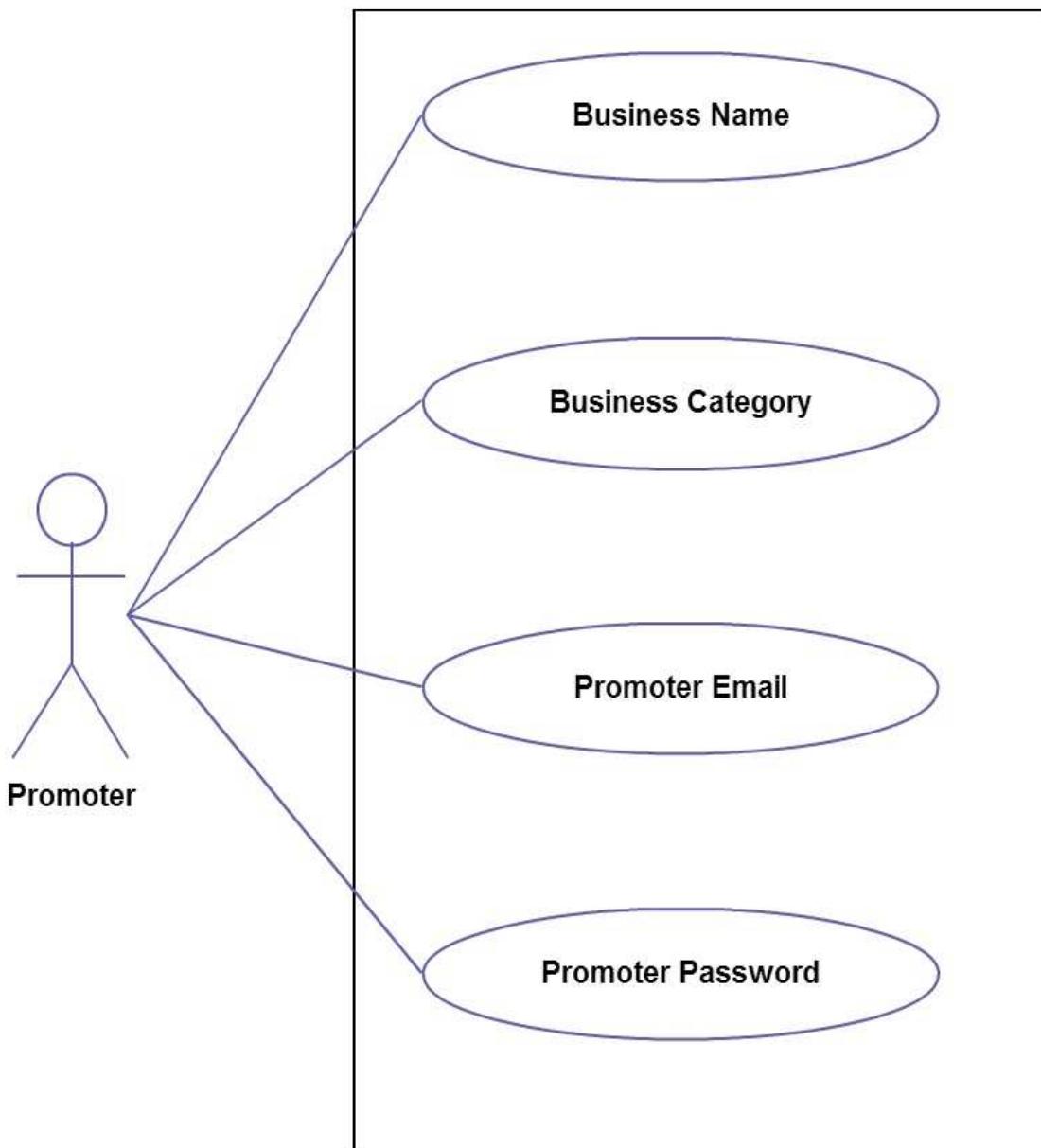


17

3.7.1.3 Result Tab Screen:



3.7.1.4 Profile Tab:



4. Design:

4.1 Diagrams:

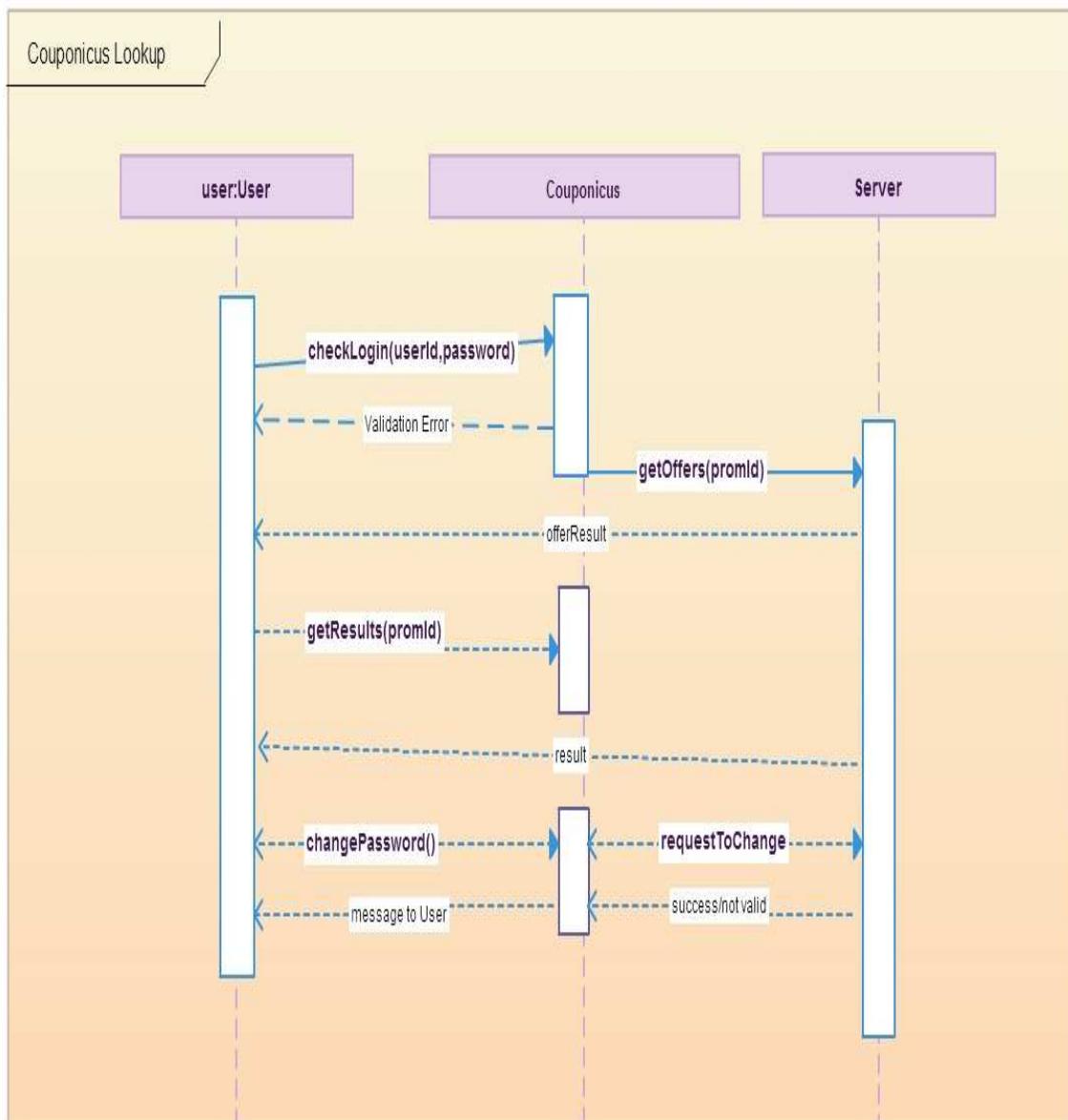
4.1.1 Sequence Diagram:

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagrams useful to communicate how the business currently works by showing how various business objects interact. Besides documenting an organization's current affairs, a business-level sequence diagram can be used as a requirements document to communicate requirements for a future system implementation. During the requirements phase of a project, analysts can take use cases to the next level by providing a more formal level of refinement. When that occurs, use cases are often refined into one or more sequence diagrams.

An organization's technical staff can find sequence diagrams useful in documenting how a future system should behave. During the design phase, architects and developers can use the diagram to force out the system's object interactions, thus fleshing out overall system design.

One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. In addition to their use in designing new systems, sequence diagrams can be used to document how objects in an existing (call it "legacy") system currently interact. This documentation is very useful when transitioning a system to another person or organization.

4.1.1.1 Couponicus Sequence Diagram:



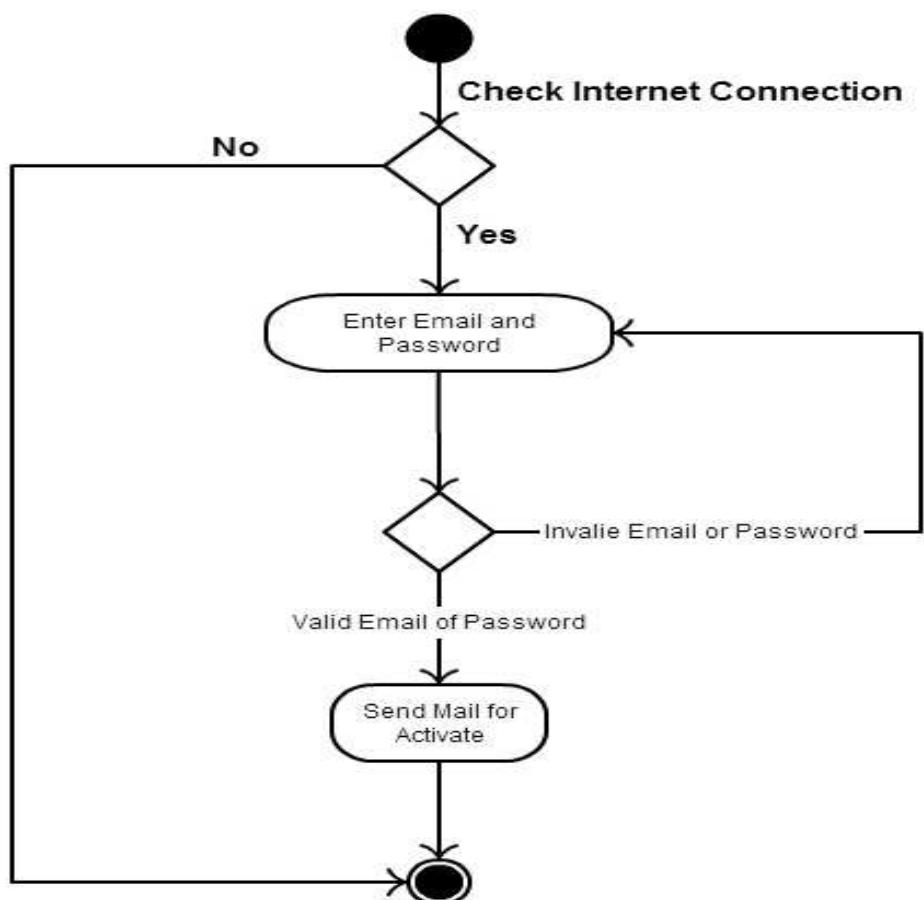
4.1.2 Activity Diagram:

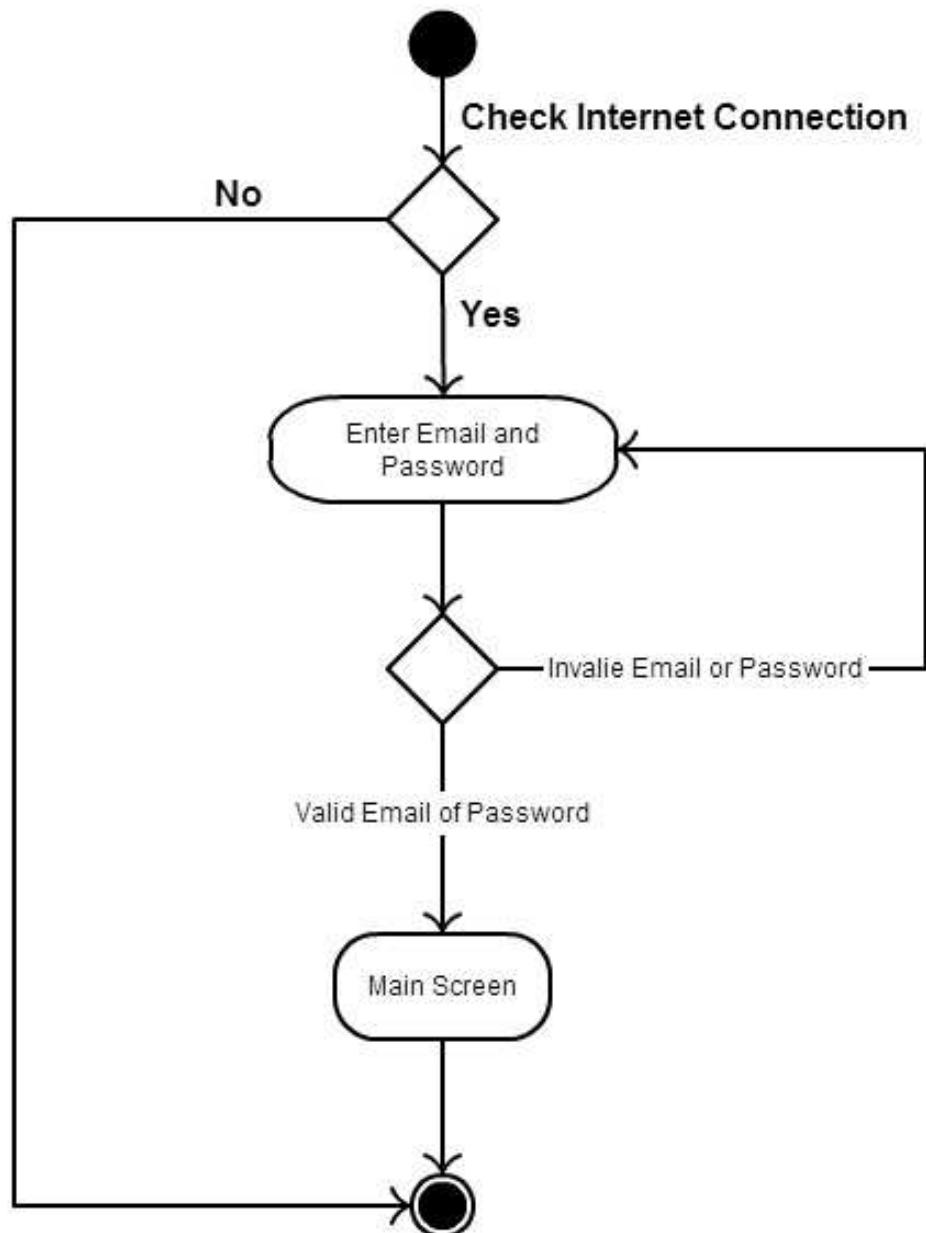
The Activity diagram models logic-any logic-from work flow to use cases to methods. It borrows most of its notation from flowcharts, but has added the concept of concurrency to support many modern applications.

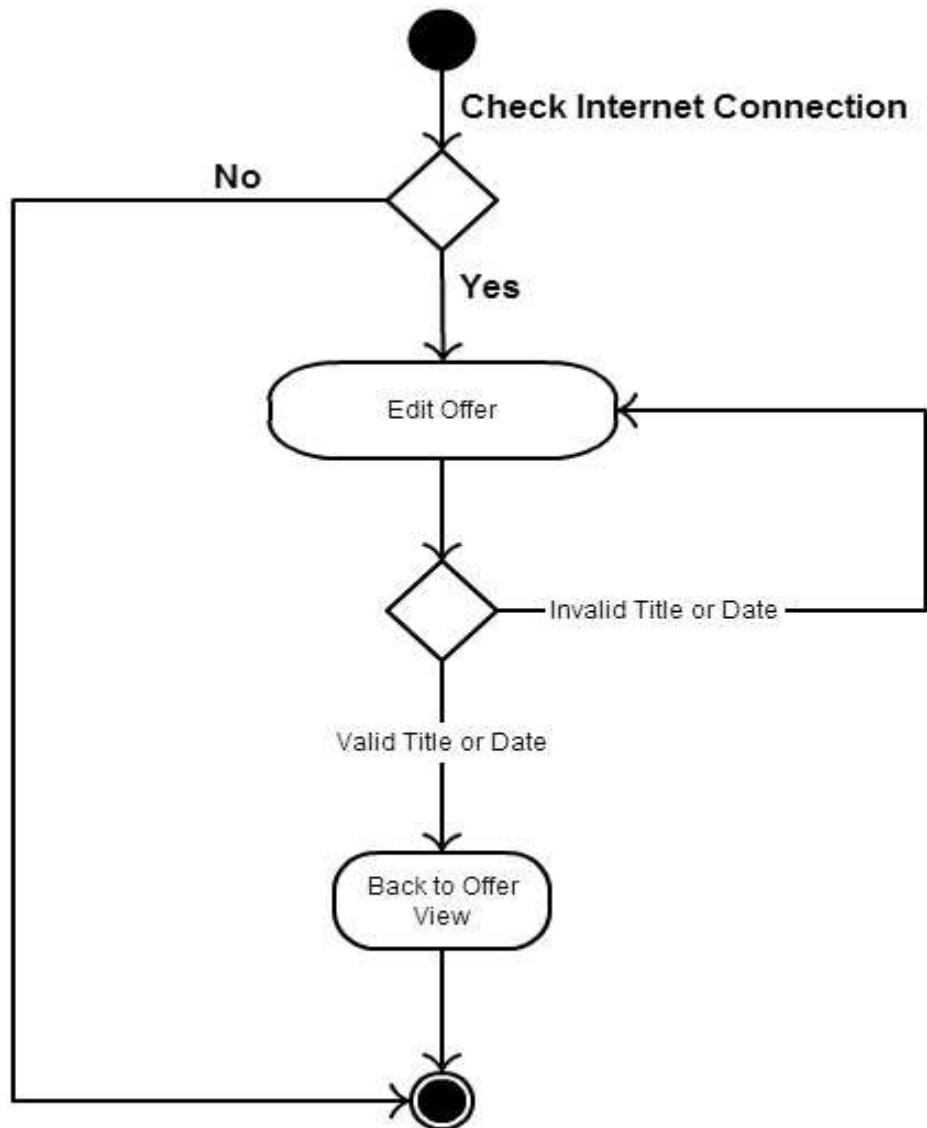
Activity diagrams are valuable because they

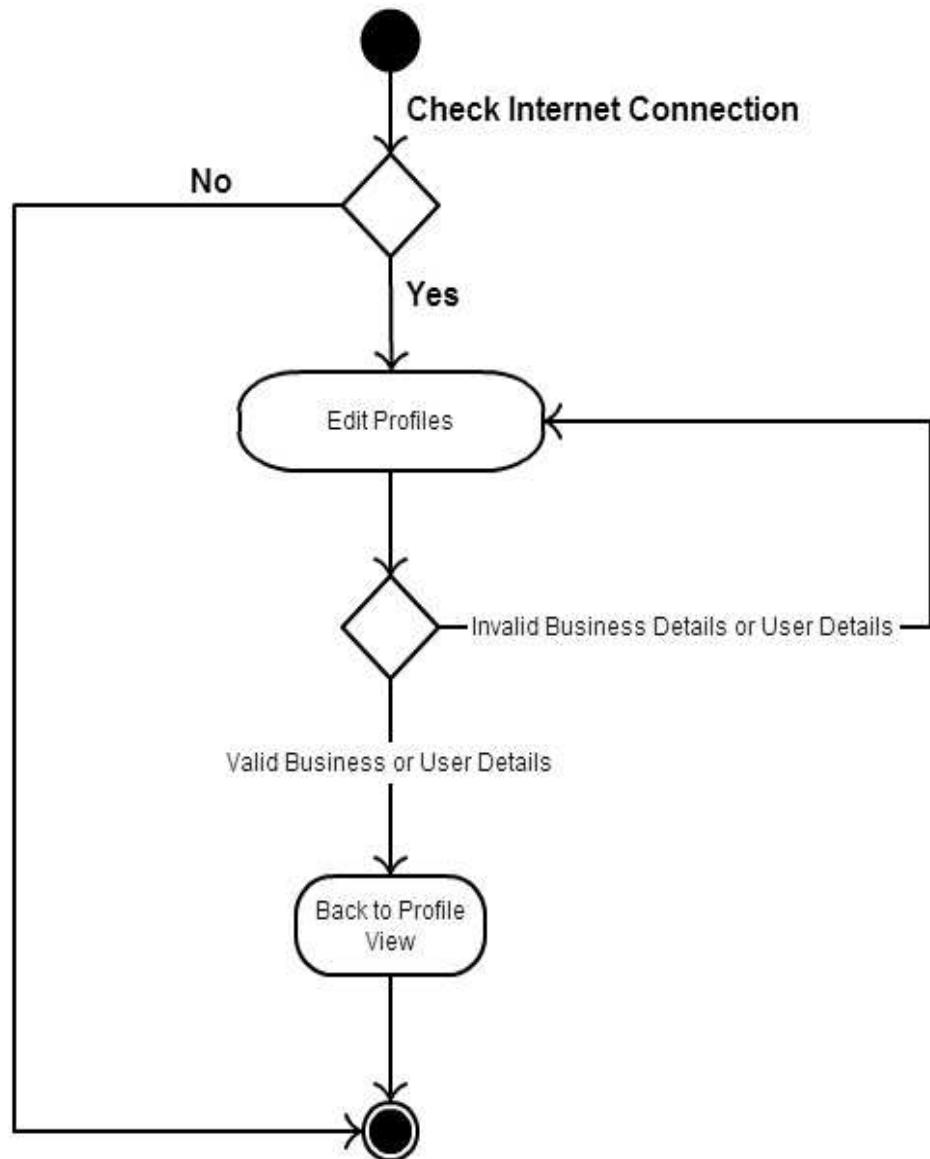
- Represent the logic required to implement system behaviors.
- Represent logic at any level the design needs, from system workflow to individual method implementations.
- Are simple enough to learn quickly.
- Are relatively familiar to users since they are often used in business training and procedures manuals.

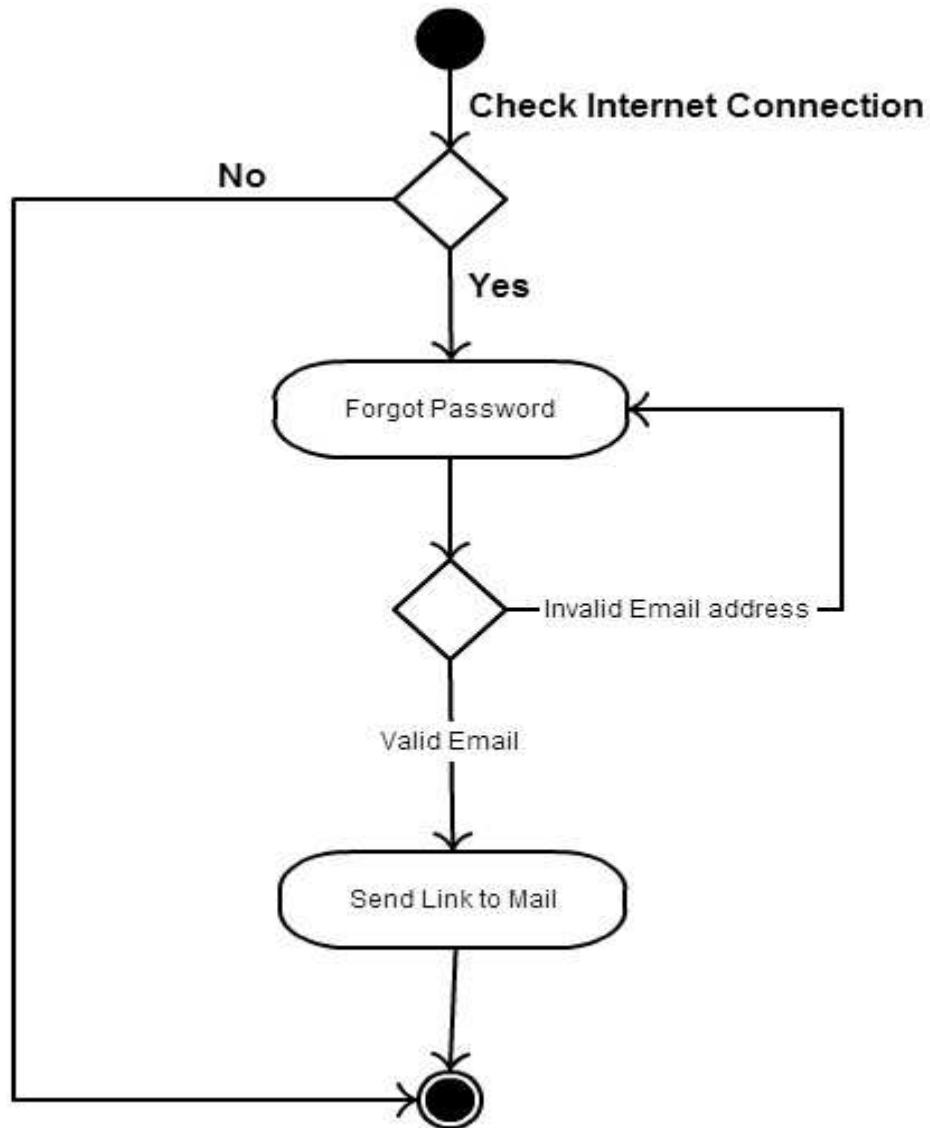
4.1.2.1 *Create Promoter:*



4.1.2.2 Login Activity:

4.1.2.3 Edit Offer Activity:

4.1.2.4 Edit Profile Activity:

4.1.2.5 *Forgot Password:*

4.1.3 Class Diagram:

The Class diagram is at the heart of the object modeling process. It models the definitions of resources essential to the proper operation of the system. All of the other modeling diagrams discover information about these resources (such as attribute values, state, and constraints on behavior) that ultimately must make its way onto the Class diagram. The Class diagram is the source for code generation (converting a model to code) and the target for reverse engineering (converting code to a model).

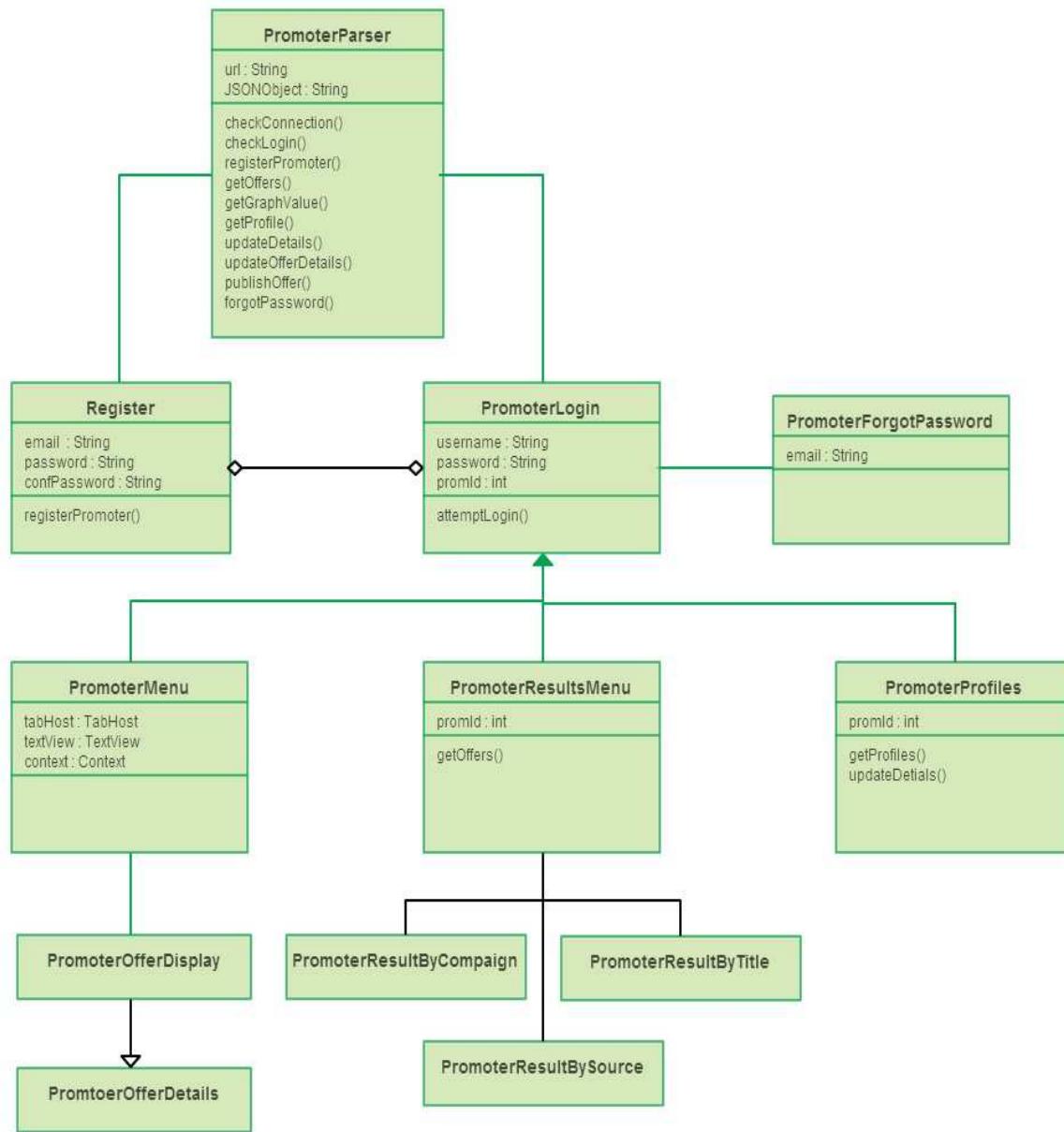
The Class diagram models the resources used to build and operate the system. Resources represent people, materials, information, and behaviors. Class diagrams model each resource in terms of its structure, relationships, and behaviors.

Class diagram models resources and relationships between resources. It defines how many objects may participate in the relationship and may even set rules defining which objects can participate.

Class diagrams are valuable because they

- Define the essential resources of a system.
- Define relationships between resources.
- Generate code.
- Model code (reverse engineering).
- Provide a focus for all the other diagrams.

4.1.3.1 Class Diagram for Couponicus:



4.2 Data Dictionary:

Data dictionary is a catalogue a repository of the elements in a system. They are prepared to meet user requirements to know system elements and their purposes.

Data dictionary consists of list of the elements composing the data flow through the system. The major elements are data flows, data stores and processes. The data dictionary stores detailed description for these elements.

The dictionary is developed during data flow analysis and assists the analysts involved in determining system requirements.

Data dictionary is one of the most important elements used during system analysis and design process.

The data dictionary is considered important because -

- To manage detail in large system.
- To communicate a common meaning for all system elements.
- To document the features of the system.
- To facilitate analysis of details in order to evaluate characteristics and determine whether system changes should be made.
- To locate error and omissions in the system.

a) Table name: **brand**

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id	int(11)			No	None	auto_increment	
<input type="checkbox"/> bussCatId	int(11)			Yes	NULL		
<input type="checkbox"/> promId	int(11)			Yes	NULL		
<input type="checkbox"/> brandName	varchar(50)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> storeCode	varchar(10)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> locationName	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> street	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> zip	varchar(5)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> city	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> contactNo	varchar(20)	utf8_general_ci		Yes	NULL		

b) Table name: **businesscategory**

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id	int(11)			No	None	auto_increment	
<input type="checkbox"/> catName	tinytext	utf8_general_ci		No	None		

c) Table name: **coupon**

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id	int(11)			No	None	auto_increment	
<input type="checkbox"/> offerId	int(11)			Yes	NULL		
<input type="checkbox"/> brandId	int(11)			Yes	NULL		
<input type="checkbox"/> promId	int(11)			Yes	NULL		
<input type="checkbox"/> claimFrom	enum('fb','tw','wb','em')	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> claimDate	timestamp			Yes	CURRENT_TIMESTAMP		
<input type="checkbox"/> redemptionDate	timestamp			Yes	NULL		
<input type="checkbox"/> firstName	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> lastName	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> email	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> redemptionFrom	int(1)			Yes	NULL		
<input type="checkbox"/> status	int(1)			Yes	NULL		

d) Table name: **customer**

Server: localhost ▶ Database: couponicus ▶ Table: customer											
	Field	Type	Collation	Attributes	Null	Default	Extra	Action			
	<code>id</code>	int(11)			No	None	auto_increment				
	<code>email</code>	varchar(100)	utf8_general_ci		Yes	NULL					
	<code>firstName</code>	varchar(100)	utf8_general_ci		Yes	NULL					
	<code>lastName</code>	varchar(100)	utf8_general_ci		Yes	NULL					

e) Table name: **facebook**

Server: localhost ▶ Database: couponicus ▶ Table: facebook											
	Field	Type	Collation	Attributes	Null	Default	Extra	Action			
	<code>id</code>	int(11)			No	None	auto_increment				
	<code>brandId</code>	int(11)			Yes	NULL					
	<code>offerId</code>	int(11)			Yes	NULL					
	<code>promId</code>	int(11)			Yes	NULL					
	<code>fbuserId</code>	varchar(60)	utf8_general_ci		Yes	NULL					
	<code>userName</code>	varchar(50)	utf8_general_ci		Yes	NULL					
	<code>firstName</code>	varchar(20)	utf8_general_ci		Yes	NULL					
	<code>lastName</code>	varchar(20)	utf8_general_ci		Yes	NULL					
	<code>otherInfo</code>	text	utf8_general_ci		Yes	NULL					
	<code>pageId</code>	varchar(100)	utf8_general_ci		Yes	NULL					
	<code>pageName</code>	varchar(100)	utf8_general_ci		Yes	NULL					
	<code>pageUrl</code>	varchar(100)	utf8_general_ci		Yes	NULL					
	<code>pageToken</code>	text	utf8_general_ci		Yes	NULL					
	<code>userToken</code>	text	utf8_general_ci		Yes	NULL					
	<code>status</code>	int(1)			Yes	NULL					

f) Table name: **groups**

Server: localhost ▶ Database: couponicus ▶ Table: groups											
	Field	Type	Collation	Attributes	Null	Default	Extra	Action			
	<code>id</code>	int(11)			No	None	auto_increment				
	<code>name</code>	varchar(20)	utf8_general_ci		No	None					
	<code>description</code>	varchar(100)	utf8_general_ci		No	None					

g) Table name: **offer**

Server: localhost ▶ Database: couponicus ▶ Table: offer								
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	<input type="checkbox"/> id	int(11)			No	None	auto_increment	       
	<input type="checkbox"/> brandId	int(11)			Yes	NULL		       
	<input type="checkbox"/> promId	int(11)			Yes	NULL		       
	<input type="checkbox"/> offerTitle	varchar(100)	utf8_general_ci		Yes	NULL		       
	<input type="checkbox"/> description	text	utf8_general_ci		Yes	NULL		       
	<input type="checkbox"/> startDate	timestamp			Yes	NULL		       
	<input type="checkbox"/> endDate	timestamp			Yes	NULL		       
	<input type="checkbox"/> offerImage	varchar(200)	utf8_general_ci		Yes	NULL		       
	<input type="checkbox"/> status	int(1)			Yes	NULL		       

h) Table name: **offerrestriction**

Server: localhost ▶ Database: couponicus ▶ Table: offerrestriction								
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	<input type="checkbox"/> id	int(11)			No	None	auto_increment	       
	<input type="checkbox"/> offerId	int(11)			Yes	NULL		       
	<input type="checkbox"/> restricTitle	text	utf8_general_ci		Yes	NULL		       
	<input type="checkbox"/> restricId	int(11)			Yes	NULL		       
	<input type="checkbox"/> restricDescription	text	utf8_general_ci		Yes	NULL		       

i) Table name: **offersuggestion**

Server: localhost ▶ Database: couponicus ▶ Table: offersuggestion								
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
	<input type="checkbox"/> id	int(11)			No	None	auto_increment	       
	<input type="checkbox"/> bussCatId	int(11)			No	None		       
	<input type="checkbox"/> title	varchar(100)	utf8_general_ci		No	None		       
	<input type="checkbox"/> description	text	utf8_general_ci		No	None		       

j) Table name: **promoter**

Server: localhost ▶ Database: couponicus ▶ Table: promoter								
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	id	int(11)			No	None	auto_increment	      
<input type="checkbox"/>	userId	int(11)			No	None		      
<input type="checkbox"/>	firstName	varchar(100)	utf8_general_ci		Yes	NULL		      
<input style="background-color: #e0ffe0; border: 1px solid #008000;" type="checkbox"/>	lastName	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	registrationDate	timestamp			No	CURRENT_TIMESTAMP		      
<input type="checkbox"/>	locName	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	street1	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	street2	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	zip	varchar(5)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	city	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	country	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	contactNo	varchar(20)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	storeCode	varchar(10)	utf8_general_ci		Yes	NULL		      

k) Table name: **restriction**

Server: localhost ▶ Database: couponicus ▶ Table: restriction								
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	id	int(11)			No	None	auto_increment	      
<input type="checkbox"/>	title	varchar(100)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	description	text	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	status	int(1)			Yes	NULL		      

l) Table name: **twitter**

Server: localhost ▶ Database: couponicus ▶ Table: twitter								
	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	id	int(11)			No	None	auto_increment	      
<input type="checkbox"/>	brandId	int(11)			Yes	NULL		      
<input type="checkbox"/>	offerId	int(11)			Yes	NULL		      
<input type="checkbox"/>	promId	int(11)			Yes	NULL		      
<input type="checkbox"/>	screenName	text	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	twuserId	varchar(60)	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	oauthToken	text	utf8_general_ci		Yes	NULL		      
<input type="checkbox"/>	oauthTokenSecret	text	utf8_general_ci		Yes	NULL		      

m) Table name: **users**

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id	int(11)			No	None	auto_increment	
<input type="checkbox"/> ip_address	varbinary(16)			No	None		
<input type="checkbox"/> username	varchar(100)	utf8_general_ci		No	None		
<input type="checkbox"/> password	varchar(80)	utf8_general_ci		No	None		
<input type="checkbox"/> salt	varchar(40)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> email	varchar(100)	utf8_general_ci		No	None		
<input type="checkbox"/> activation_code	varchar(40)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> forgotten_password_code	varchar(40)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> forgotten_password_time	int(11)		UNSIGNED	Yes	NULL		
<input type="checkbox"/> remember_code	varchar(40)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> created_on	int(11)		UNSIGNED	No	None		
<input type="checkbox"/> last_login	int(11)		UNSIGNED	Yes	NULL		
<input type="checkbox"/> active	tinyint(1)		UNSIGNED	Yes	NULL		
<input type="checkbox"/> first_name	varchar(50)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> last_name	varchar(50)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> company	varchar(100)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> phone	varchar(20)	utf8_general_ci		Yes	NULL		

n) Table name: **users_groups**

Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id	int(11)			No	None	auto_increment	
<input type="checkbox"/> user_id	int(11)			No	None		
<input type="checkbox"/> group_id	int(11)			No	None		

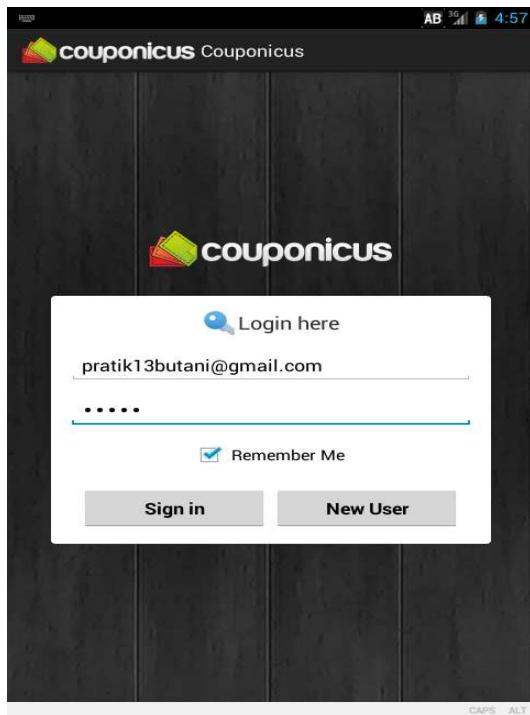
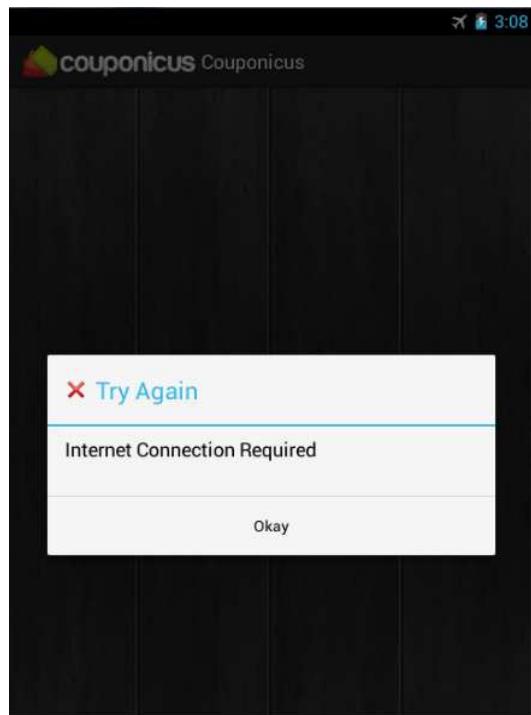
o) Table name: **widget**

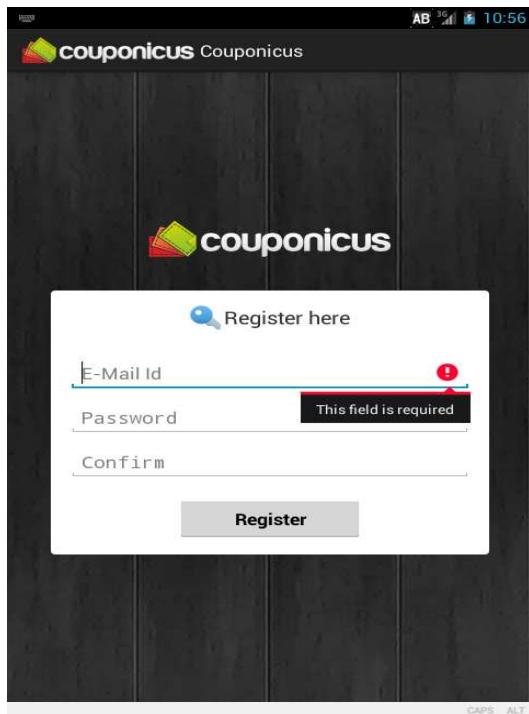
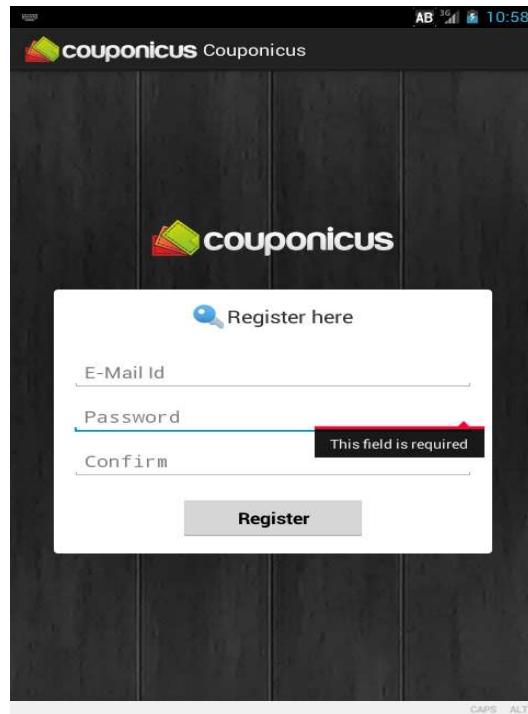
Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> id	int(11)			No	None	auto_increment	
<input type="checkbox"/> offerId	int(11)			Yes	NULL		
<input type="checkbox"/> brandId	int(11)			Yes	NULL		
<input type="checkbox"/> promId	int(11)			Yes	NULL		
<input type="checkbox"/> color	varchar(10)	utf8_general_ci		Yes	NULL		
<input type="checkbox"/> position	int(1)			Yes	NULL		

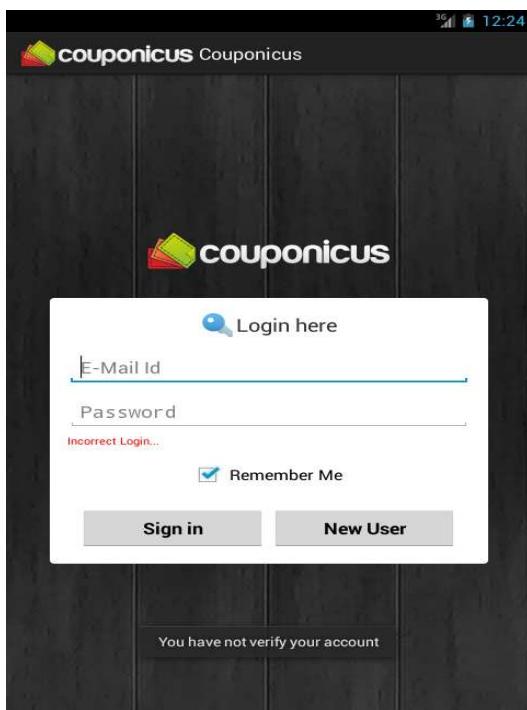
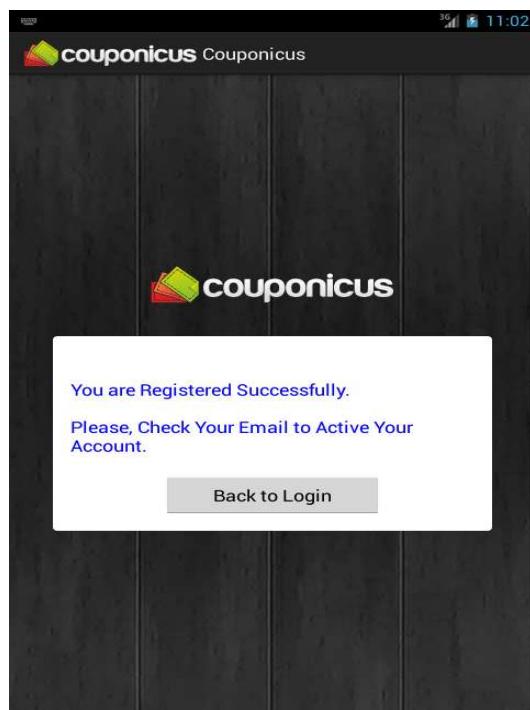
5. Implementation and Testing:

5.1 Input Event:

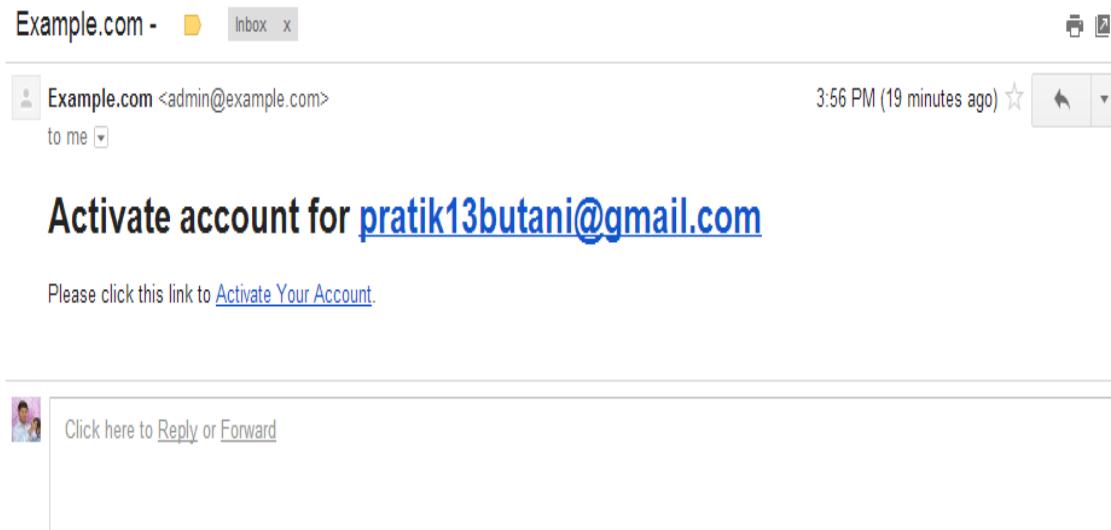
Input Design is the process of the converting user oriented inputs to a computer based format. Inaccurate Inputted data have the most common sense of error in data processing. Any ambiguity conceiving at input leads to a total fault in output. The goal of designing the input data is to make the data entry easy, logical and error free as possible, Inputs are made using forms and data validation is done at the time of inputting data.

1. Login Screen:**2. Internet Connection Error Screen:****3. Login (Email) Error Screen:****4. Login (Password) Error Screen:****5. Processing for Signing:****6. Register Screen:**

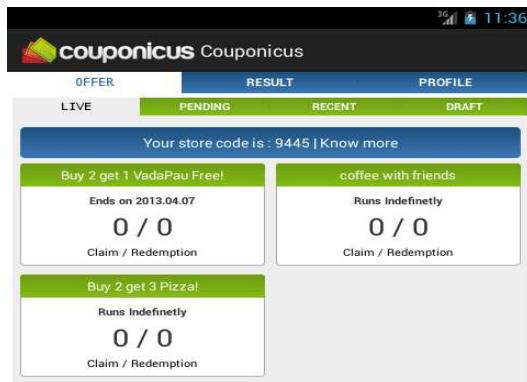
7. Register (Email) Error Screen:**8. Register (Password) Error:**

9. Register (Confirm Password) Error:**10. Register Email Already Exists:****11. Email not verified Error Screen:****12. Register Success Screen:**

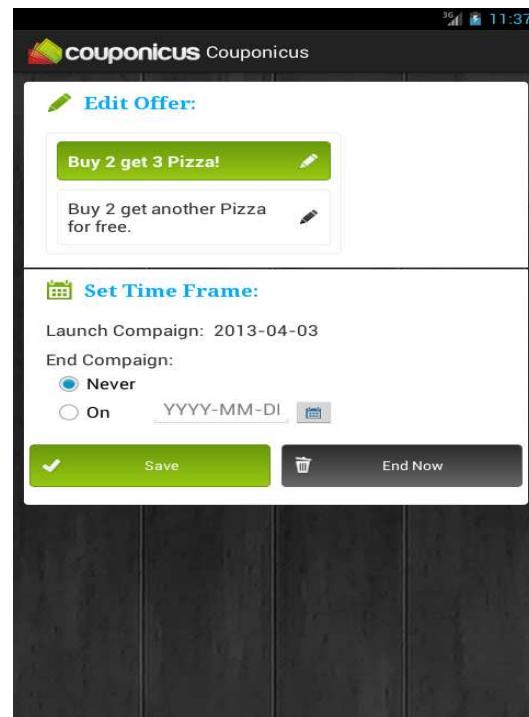
13. Mail to Active Account Screen:

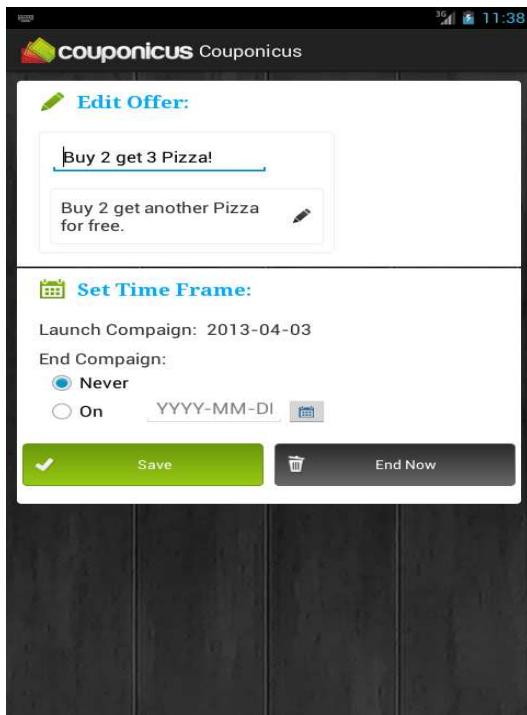
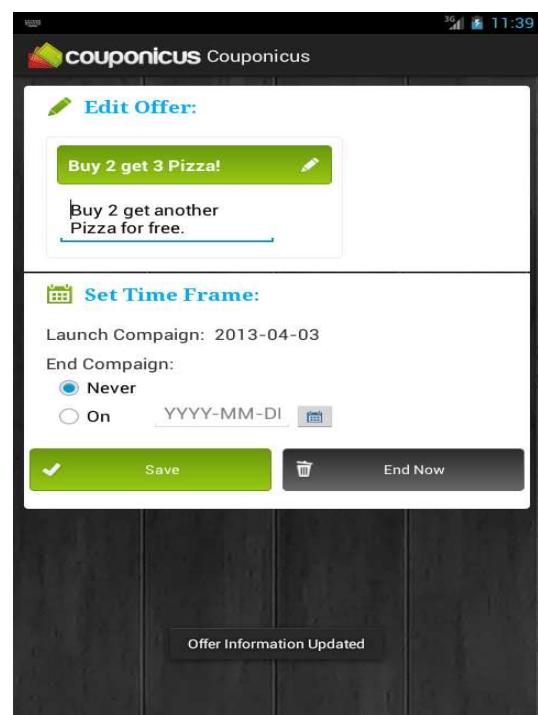
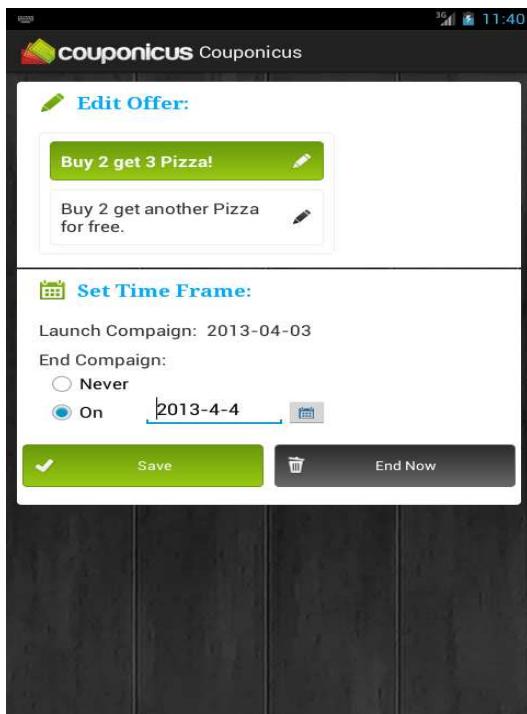
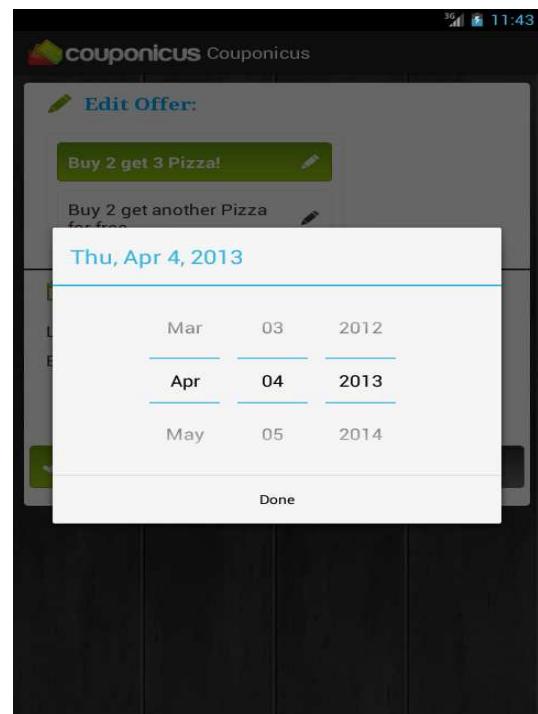


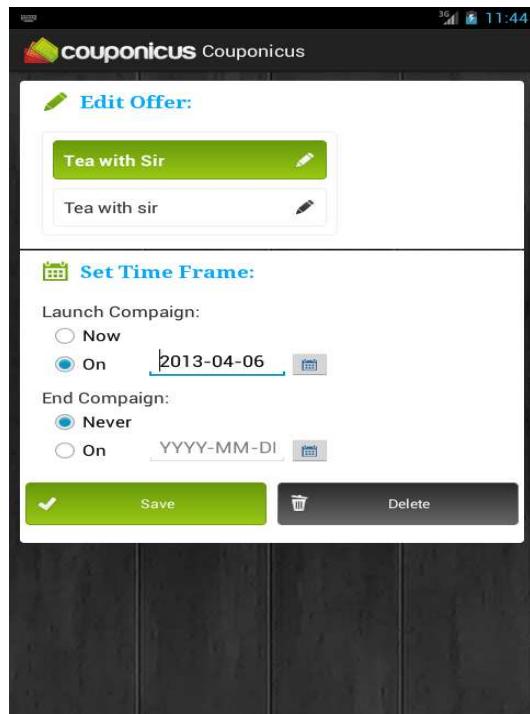
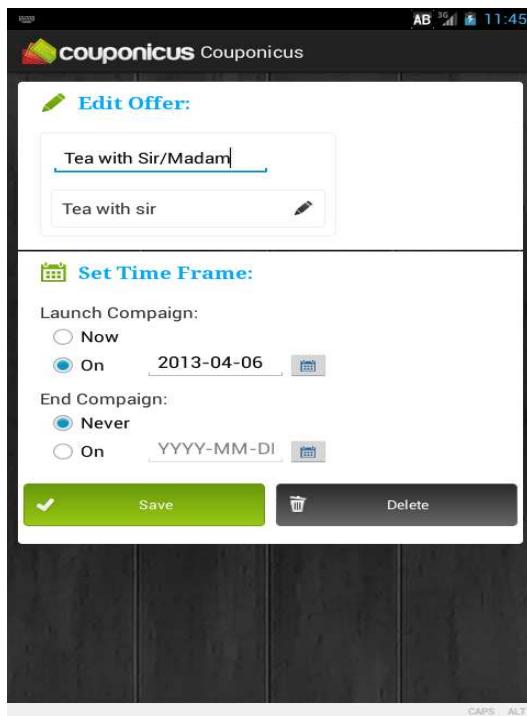
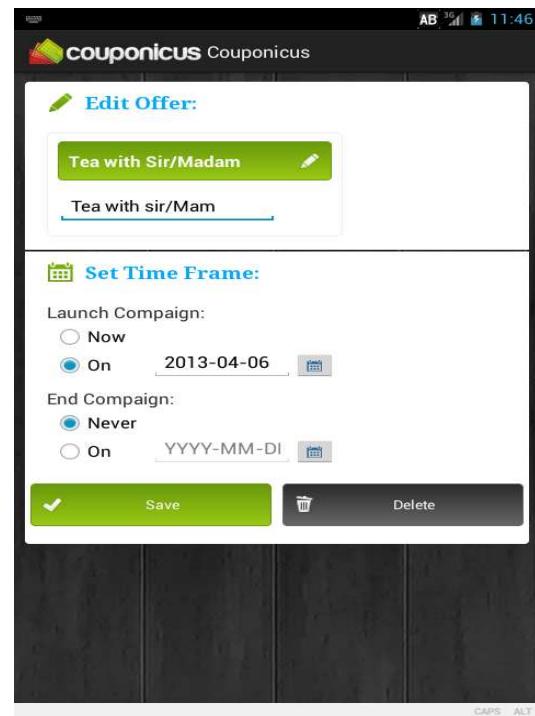
14. Live offers Screen:



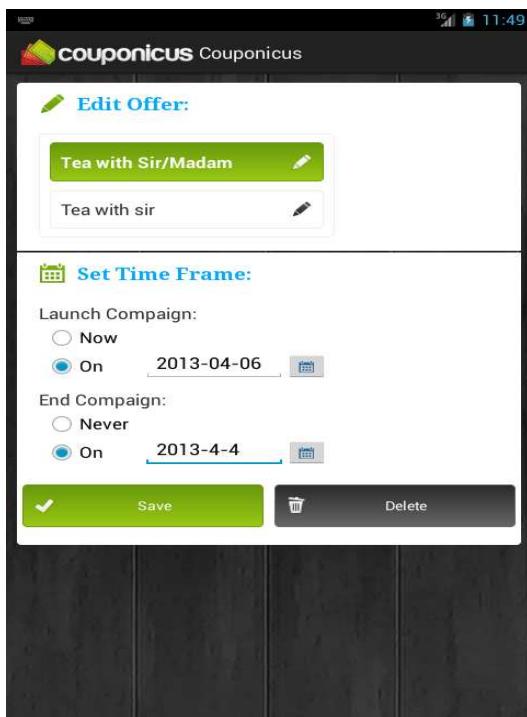
15. Live offers - Edit Screen:



17. Live offers – Edit Title Screen:**18. Live offers – Edit Description:****19. Live offers – Edit Date Screen:****20. Live offers – Edit Set Date:**

21. Pending offers Screen:**22. Pending offers - Edit Screen:****23. Pending offers – Edit Title:****24. Pending offer - Edit Description:**

25. Pending offers – Edit Date:



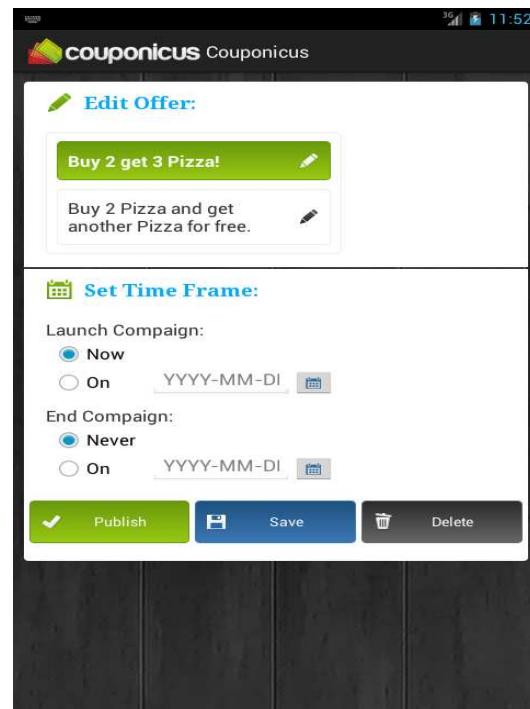
26. Recent offers Screen:

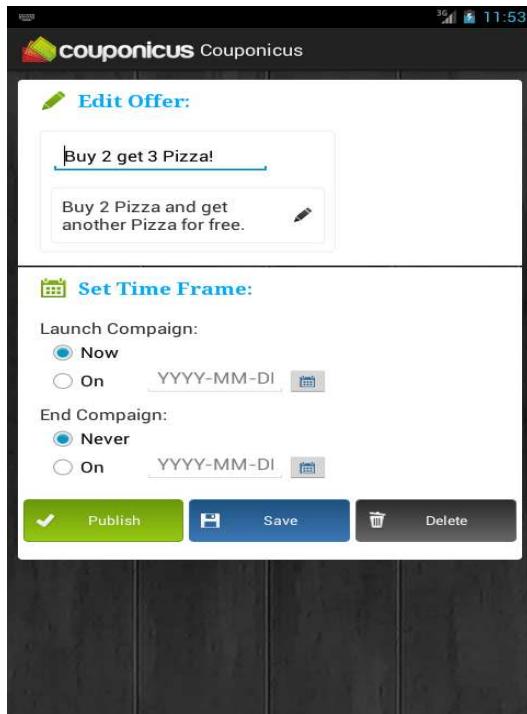
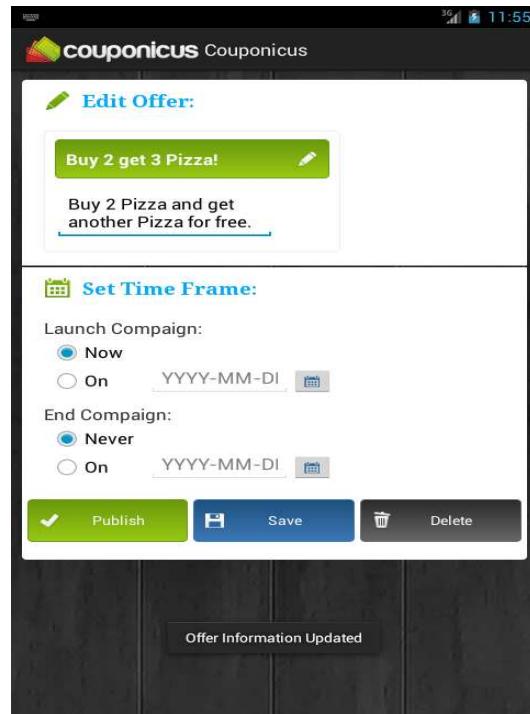
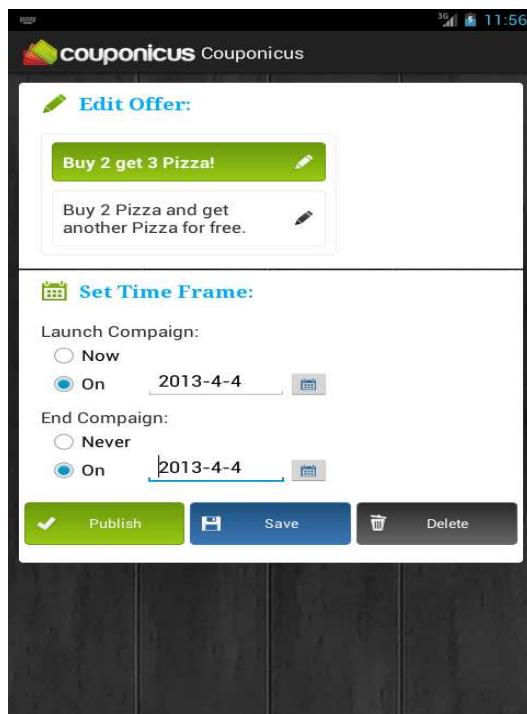


28. Draft offers Screen:

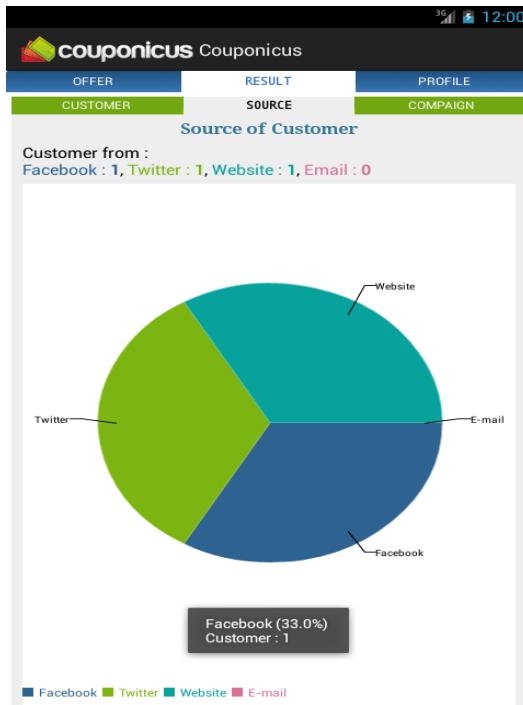


29. Draft offers - Edit Screen:

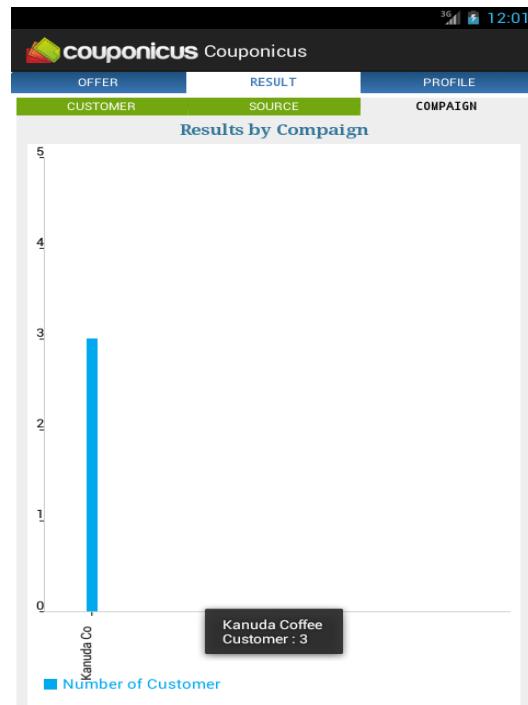


30. Draft offers – Edit Title:**31. Draft offers – Edit Description:****32. Draft offers – Edit Date:****33. Result-Total Customer Gained:**

34. Results – Source of Customer:



35. Results – Result By Compaign:



36. Profile Screen:

Bussiness Profile:

Business Name: Chatkazz

Business Category: Reataurant & Fast food

Account Detail:

Email: pratik13butani@gmail.com

Password : Change Password

37. Profiles – Edit Business Title:

Bussiness Profile:

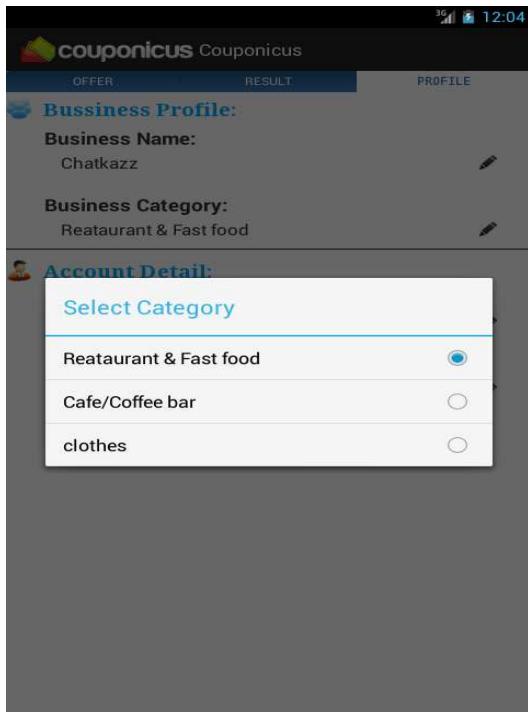
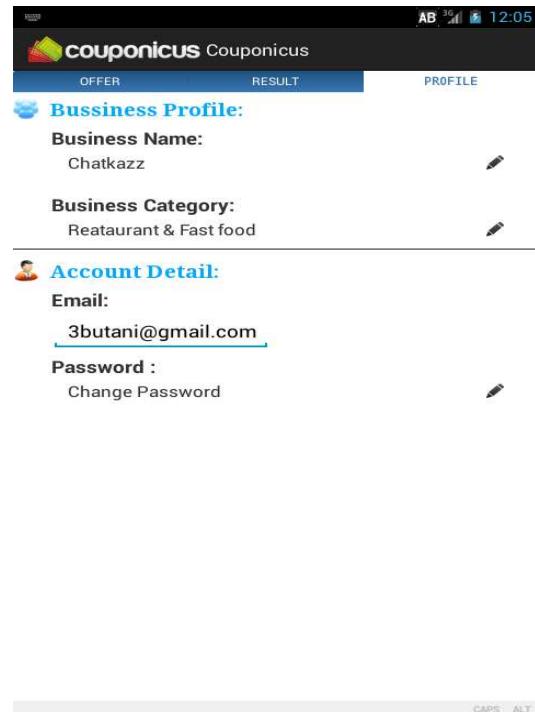
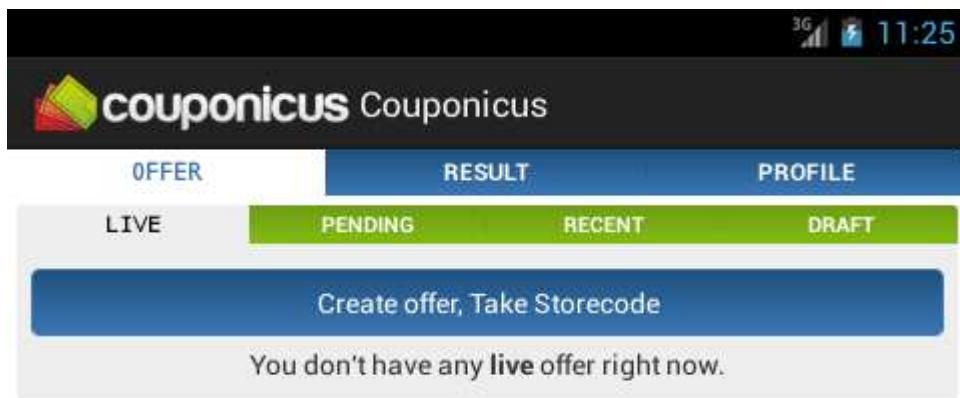
Business Name: Chatkazz

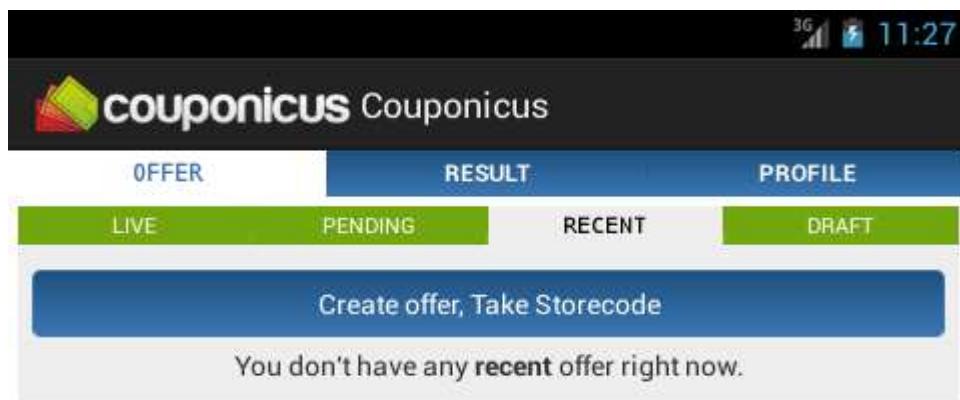
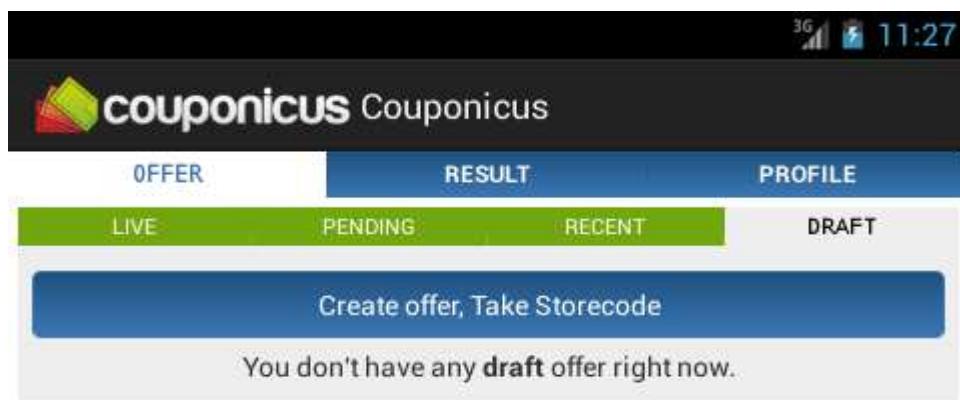
Business Category: Reataurant & Fast food

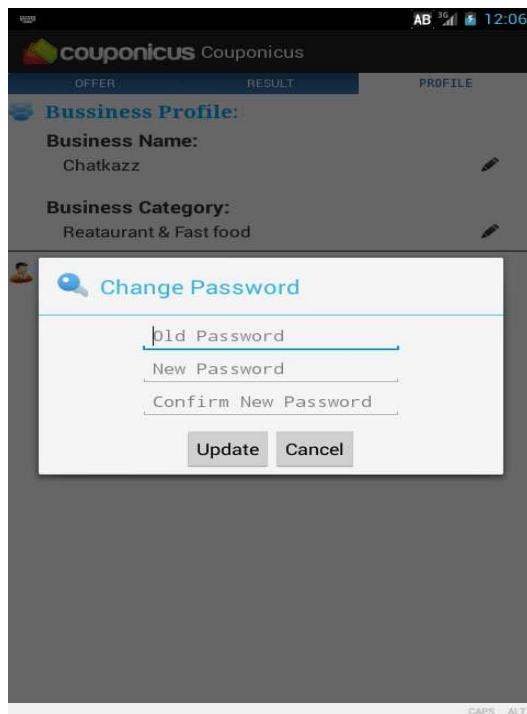
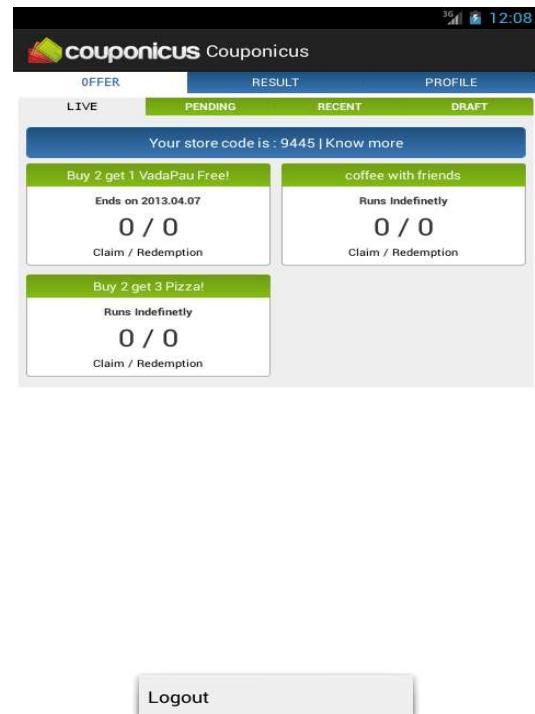
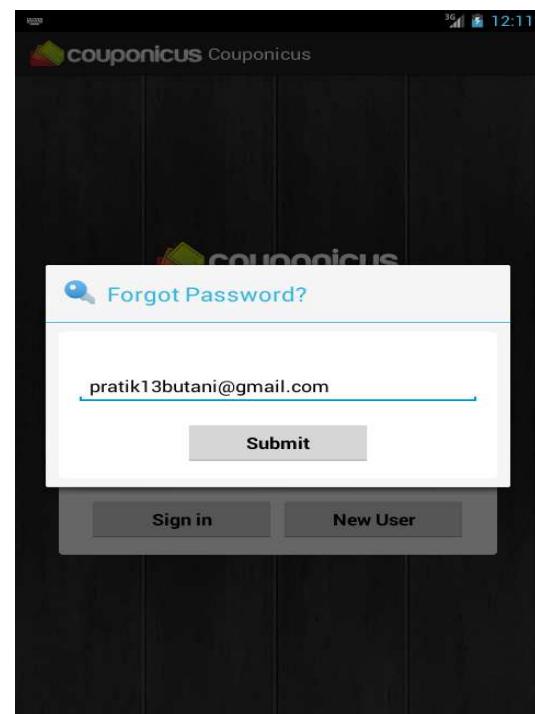
Account Detail:

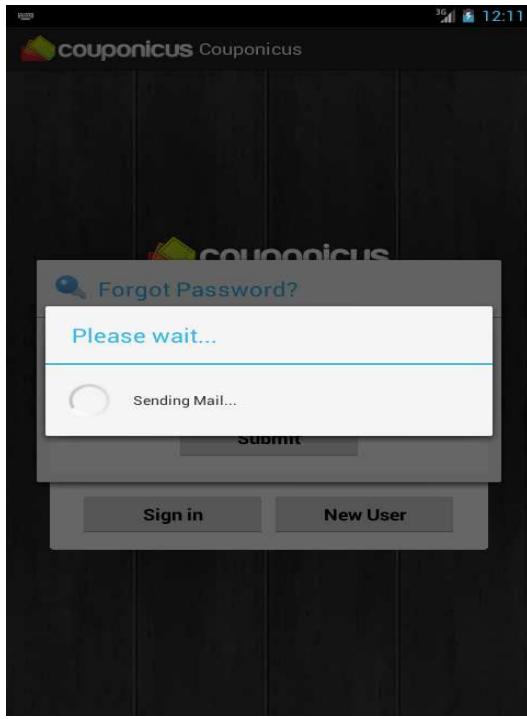
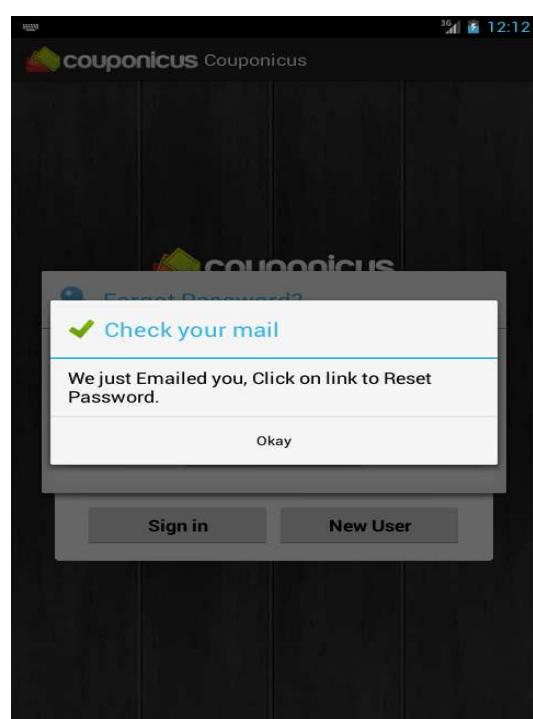
Email: pratik13butani@gmail.com

Password : Change Password

38. Profiles – Edit Business Category:**39. Profiles – Edit Email Address:****40. Live offers (if not Available):**

41. Pending offers (if not Available):**42. Recent offers (if not Available):****43. Draft offers (if not Available):**

44. Profiles – Change Password:**45. Logout in Menu Screen:****46. Forgot Password in Menu:****47. Forgot Password Screen:**

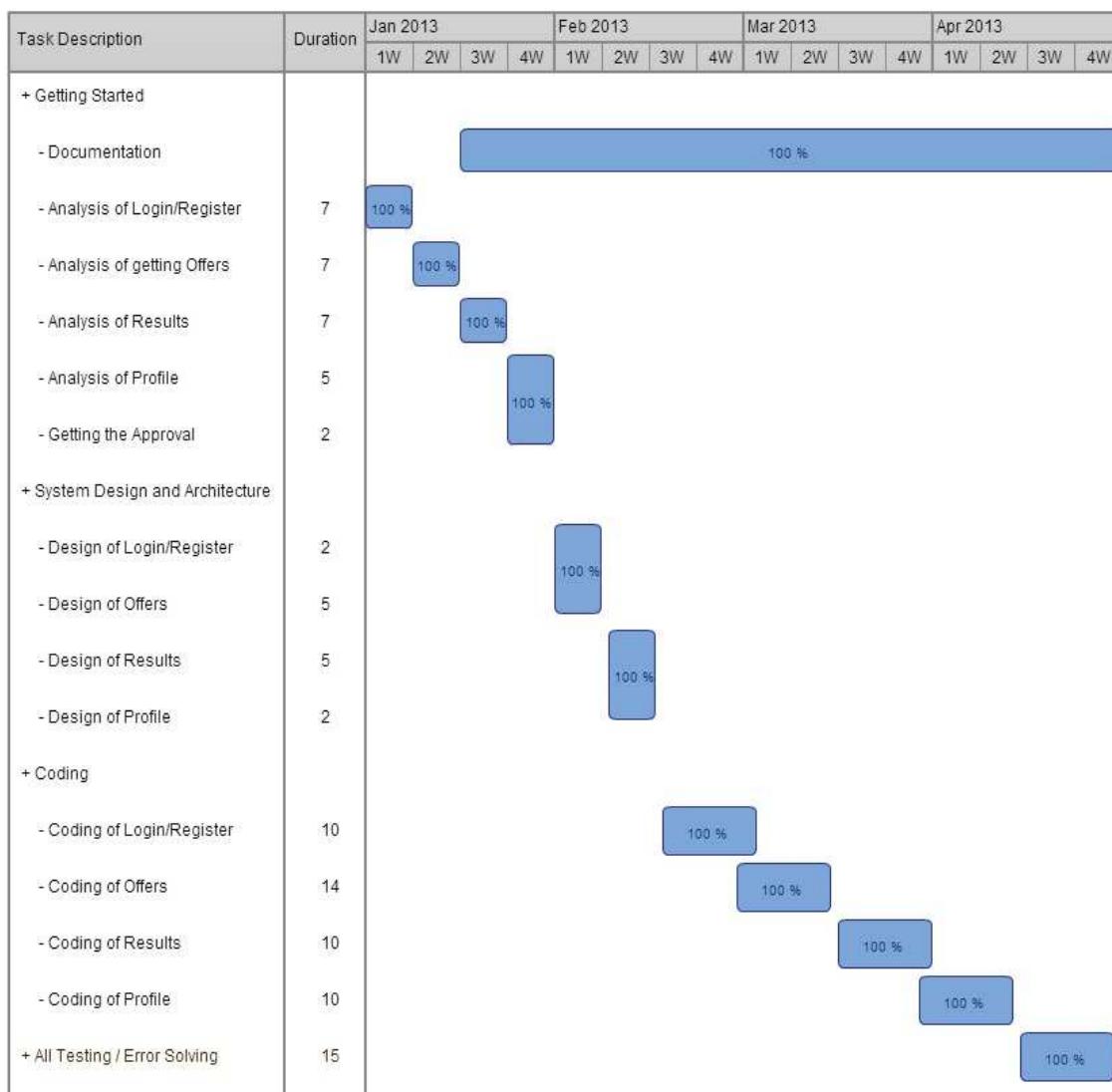
48. Processing for Sending Mail:**49. Forgot Password Successfully:**

5.2 Report Layout:

A document containing information organized in a narrative, graphic, or tabular form, prepared on ad hoc, periodic, recurring, regular, or as required basis.

Using Report layout we get many advantages which are as following:

- Increased readability
- Increased accessibility
- Structuring and organizing content is easier
- Easily understandable
- Writing with colleagues is easier to manage



5.3 Testing:

5.3.1 Testing Plan:

5.3.1.1 *The Testing Process:*

Developer tests the software process activities such as Design, Implementation, and Requirement Engineering. Because, design errors are very costly to repair once system has been started to operate, therefore, it is quite obvious to repair them at early stage of the system. Therefore, analysis is the most important process of any project.

5.3.1.2 *Requirement Traceability:*

As most interested portion is whether the system is meeting its requirements or not, for that testing should be planned so that all requirements are individually tested. Developer checked the output of certain combination of inputs, which gives desirable results, or not. Strictly stick to your requirements specifications, give you the path to get desirable results from the system.

5.3.1.3 *Tested Items:*

Our tested items are like:

- Check for GPS and Internet Connections
- Check for valid Source and Destination points entered by user
- Orientation changes handling
- Transitions between two activities and passing data
- Database connections and updates handling

5.3.1.4 *Testing Schedule:*

I have tested each procedure back-to-back so that errors and omissions can be found as early as possible. Once the system has been developed fully I have tested it on different devices, which differs in configuration.

5.3.2 Testing Strategy:

5.3.2.1 *Integration & System Testing:*

The system is first divided in units that are developed and tested for their functionalities. These units are integrated into a complete system during Integration phase and tested to check if all modules/units coordinate between each other and the system as a whole behaves as per the specifications. After successfully testing the software, it is delivered to the customer.

5.3.2.2 *Black Box Testing Strategy:*

Black Box Testing is not a type of testing; it instead is a testing strategy, which does not need any knowledge of internal design or code etc. As the name "black box" suggests, no knowledge of internal logic or code structure is required. The types of testing under this strategy are totally based/focused on the testing for requirements and functionality of the work product/software application. Black box testing is sometimes also called as "Opaque Testing", "Functional/Behavioral Testing" and "Closed Box Testing".

The base of the Black box testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming common to route the testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test the application by the developer.

In order to implement Black Box Testing Strategy, the tester is needed to be thorough with the requirement specifications of the system and as a user, should know, how the system should behave in response to the particular action. Black Box Testing strategy are: functional testing, stress testing, recovery testing, volume testing, User Acceptance Testing (also known as UAT), system testing, Sanity or Smoke testing, load testing, Usability testing, Exploratory testing, ad-hoc testing, alpha testing, beta testing etc.

5.3.2.3 *White Box Testing Strategy:*

White box testing strategy deals with the internal logic and structure of the code. White box testing is also called as glass, structural, open box or clear box testing. The tests written based on the white box testing strategy incorporate coverage of the code written, branches, paths, statements and internal logic of the code etc.

In order to implement white box testing, the tester has to deal with the code and hence is needed to possess knowledge of coding and logic i.e. internal working of the code. White box test also needs the tester to look into the code and find out which unit/statement/chunk of the code is malfunctioning.

⇒ Advantages of White box testing are

- i) As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the application effectively.
- ii) The other advantage of white box testing is that it helps in optimizing the code it helps in removing the extra lines of code, which can bring in hidden defects.

5.3.3 Testing Methods:

Software Testing involves executing an implementation of the software with test data and examining the outputs of the software and its operational behavior to check that it is performing as required.

5.3.3.1 Statistical Testing:

Statistical Testing is used to test the program's performance and reliability and to check how it works under operational conditions. Tests are designed to reflect the actual user inputs and their frequency. The stages involved in the static analysis for this system are follows.

- Control flow analysis
- Unreachable code
- Unconditional branches into loops
- Data use analysis
- Variable used before initialization
- Variables declared but never used
- Variables assigned twice but never used between assignments
- Possible array bound violations
- Parameter type mismatches
- Parameter number mismatches
- Non-usage of the results of functions
- Uncalled functions and procedures
- Storage management faults
- Insufficient memory space

5.3.3.2 Defect Testing:

Defect Testing is intended to find inconsistencies between a program and its specification. These inconsistencies are usually due to the program faults or defects.

5.3.3.3 Loop Testing:

Tester has tested the some conditions in code of application. So they test the looping in source code of application for finding miss rout or any error or wrong direction of flow in code.

5.3.3.4 Structural Testing:

Developer has done path testing to exercise every independent execution path through a component or program. If every independent path is executed then all statements in the components must have been executed at least once. The structure of our program is also checked.

5.3.3.5 *Integration Testing*

After our individual modules Developer tested out Developer go to the integrated to create a complete system. This integration process involves building the system and testing the resultant system for problems that arise from component interactions.

Developer has applied top-down strategy to validate high-level components of a system before design and implementations have been completed. Because, our development process started with high-level components and Developer worked down the component hierarchy.

5.3.3.6 *Performance Testing:*

Performance testing is designed to test the runtime performance of the system within the context of the system. These tests Developer performed as module level as Developer as system level. Individual modules Developers tested for required performance.

5.3.3.7 *Condition Testing:*

Condition testing is a test case design method that exercises the logical conditions contained in a program module. If the condition is incorrect, then as least one part of the condition is incorrect. It may include:

- String Index out of Bound error
- Null Pointer Assignment
- Input Output Connection Exceptions
- Arithmetic expression error

Developer examined the code to be tested and explicitly list each call to an external component. In the system, standards tests for GUIs have been performed, which are as follows.

- The position and related labels for all controls checked.
- All menu functions and sub functions verified for correctness.
- Validations for all inputs done.
- Each menu functions tested, whether it invokes the corresponding functionality properly.
- Pull down controls was verified for proper functionality.
- Whether the non-editable text control is disabling and it was also verified that it doesn't exceed the maximum length.

5.3.4 *Test Cases:*

The purpose of the test cases is to test the various input and see the output produce any error or not. There are different test cases according to the system. It is tested with different types of value like single value multiple value and see it can generate expected output.

- **Date related Issue:**

Bug: There are some date formats which are not supported in Android 4.0 and MySQL combination.

Solution: Direct date will be passed from Web service from MySQL.

- **Chart related Issue:**

Bug: In chart library of achartengine is gives useless output when value of results is not available.

Solution: if value is null then I have displayed message related to Result.

5.3.5 Future Enhancements:

In future, it can be made possible to:

- Create offer by promoter.
- Share offer on Facebook, Twitter and own Website.
- Claim/Redeem offer by customer.

6. About the Tools & Technology:

6.1 Develop for Android:

Android, This word means a lot in present High-Tech World. Today Smartphone are known for its operating system which is Android. Earlier there is no option for operating systems like Android in mobile, as usual there are symbian, java featured operating systems but today things had changed a lot, everyone wants a Smartphone which is functioned on Android only. In a very short span of time android created a reputed place in the market.



What is this *Android* actually? Android is a software cluster for mobile devices that includes an operating system OS, key applications and middleware. The Android SDK provides the tools and APIs required to begin developing applications on the Android platform using the Java programming language. About the design, Kernel of Android is based on Linux kernel and further furnished by Google.

The Open Handset Alliance (OHA), a group of more than 30 technology and mobile companies, was formed in 2007 to foster creativity and innovation in the mobile arena. This group includes members like Sprint, T-Mobile, Intel, Broadcom, Motorola, Samsung, Google, eBay, amongst others.

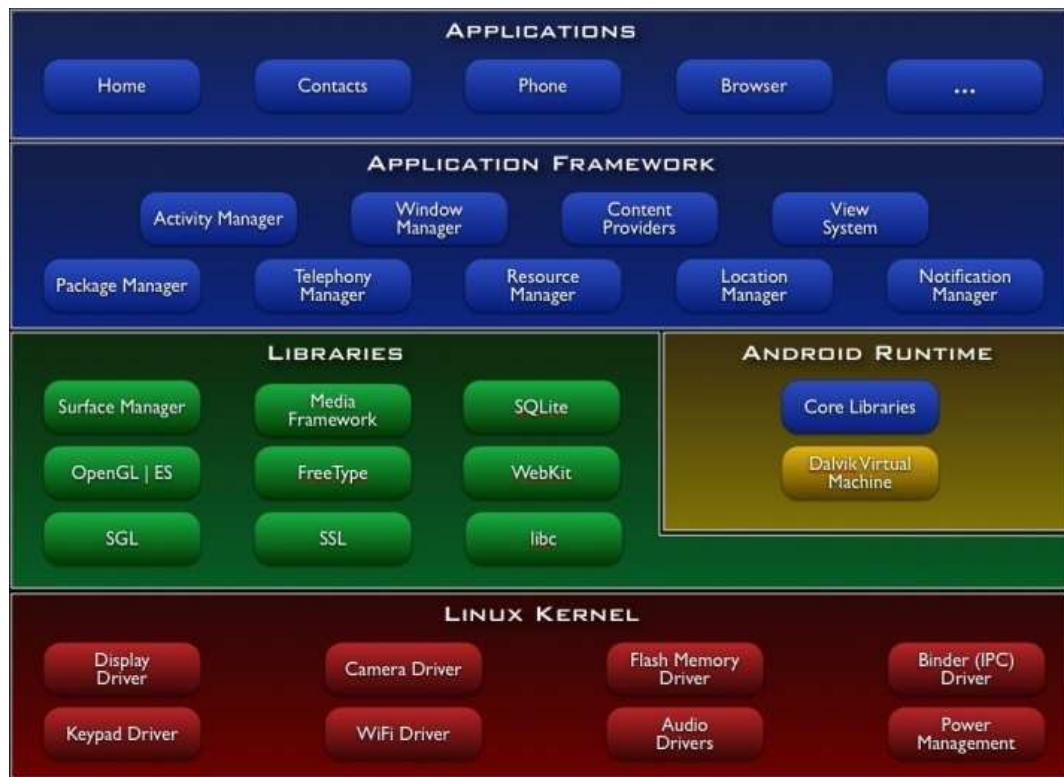
The main motive of this move was to facilitate openness in mobile software and thus give customers a diversified software base which is not only less expensive but also convenient and easy to use. The Android platform has emerged from such a move. Android is an Operating System that is designed to be used with a variety of handsets that will be launched in the second half of 2008.

Currently an SDK is hosted by Google, which is a leading player in the initiative. This SDK allows creation of various Java based programs by using a special Eclipse plug-in that interfaces with the kit. Android is not only an OS but also includes a middleware and an array of applications for the users. Some of the supported features are – a Dalvik Virtual Machine, built in browser and database support, media, camera, GPS, map and other features.

Android is a mobile operating system running on the Linux kernel. It was initially developed by Android Inc., a firm later purchased by Google, and lately by the Open Handset Alliance. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries. The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 47 hardware, software and telecom companies of a special pre-processor (called the Meta Object Compiler or MOC) to enrich the language.

6.1.1 Android Architecture:

The following diagram shows the major components of the Android operating system. Each Section is described in more detail below.



6.1.2 Applications:

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

6.1.2.1 Application Framework:

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more. Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems, including:

- A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
- Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data
- A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.
- A Notification Manager that enables all applications to display custom alerts in the status bar
- An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack

6.1.2.2 Application Component:

- **Libraries:**

Android includes a set of C/C++ libraries used by various components of the Android system.

- **Android Runtime:**

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

- **Linux Kernel:**

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

6.1.3 Development Toolkit:

6.1.3.1 *Android SDK:*

The Android SDK includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. Currently supported development platforms include x86- architecture computers running. Linux (any modern desktop Linux distribution), Mac OS X 10.4.8 or later, Windows XP or Vista. Requirements also include Java Development Kit, Apache Ant, and Python 2.2 or later. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug-in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications.

6.1.3.2 *ADT Plug-in:*

The Android Development Tools (ADT) plug-in for Eclipse adds powerful extensions to the Eclipse integrated development environment. It allows you to create and debug Android applications easier and faster. If you use Eclipse, the ADT plug-in gives you an incredible boost in developing Android applications:

- It gives you access to other Android development tools from inside the Eclipse IDE. For example, ADT lets you access the many capabilities of the DDMS tool: take screenshots, manage port-forwarding, set breakpoints, and view thread and process information directly from Eclipse.
- It provides a New Project Wizard, which helps you quickly create and set up all of the basic files you'll need for a new Android application.
- It automates and simplifies the process of building your Android application.
- It provides an Android code editor that helps you write valid XML for your Android manifest and resource files.

6.1.3.3 *Emulator:*

The Android SDK includes a mobile device emulator -- a virtual mobile device that runs on your computer. The emulator lets you prototype, develop, and test Android applications without using a physical dev. The Android emulator mimics all of the typical hardware and software features of a typical mobile device, except that it can place actual phone calls. It provides a variety of navigation and control keys, which you can "press" using your mouse or keyboard to generate events for your application. It also provides a screen in which your application is displayed, together with any other Android applications running.

To let you model and test your application more easily, the emulator supports Android Virtual Device (AVD) configurations. AVDs let you specify the Android platform that you want to run on the emulator, as well as the hardware options and emulator skin fill you want to use. Once your application is running on the emulator, it can use the services of the Android platform to invoke other applications, access the network, play audio and video, store and retrieve data, notify the user, and render graphical transitions and themes.

The emulator also includes a variety of debug capabilities, such as a console from which you can log kernel output, simulate application interrupts (such as arriving SMS messages or phone calls), and simulate latency effects and dropouts on the data channel.

6.1.3.3.1 Emulator Limitations:

- No support for placing or receiving actual phone calls.
- No support for USB connections.
- No support for camera/video capture (input).
- No support for device-attached headphones.
- No support for determining connected state.
- No support for determining battery charge level and AC charging state.
- No supports for determining SD card insert/eject.
- No support for Bluetooth.

6.1.3.4 What Androids Are Made Of

When you write a desktop application, you are "master of your own domain". You launch your main window and any child windows – like dialog boxes – that are needed. From your standpoint, you are your own world, leveraging features supported by the operating system, but largely ignorant of any other program that may be running on the computer at the same time. If you do interact with other programs, it is typically through an API, such as using JDBC (or frameworks atop it) to communicate with MySQL or another database. Android has similar concepts, but packaged differently, and structured to make phones more crash- resistant.

6.1.3.4.1 Activities:

The building block of the user interface is the activity. You can think of an activity as being the Android analogue for the window or dialog in a desktop application. While it is possible for activities to not have a user interface, most likely your "headless" code will be packaged in the form of content providers or services, described below.

6.1.3.4.2 Content Providers:

Content providers provide a level of abstraction for any data stored on the Device that is accessible by multiple applications. The Android development Model encourages you to

make your own data available to other Applications, as well as your own – building a content provider lets you do that, while maintaining complete control over how your data gets accessed.

6.1.3.4.3 Intents:

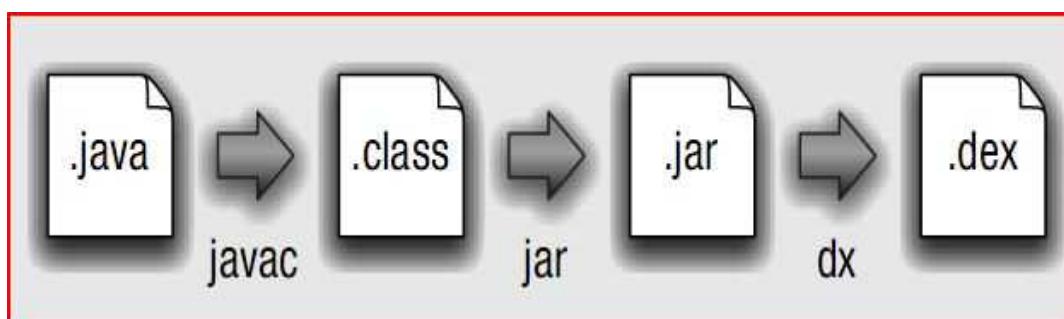
Intents are system messages, running around the inside of the device, notifying applications of various events, from hardware state changes (e.g., an SD card was inserted), to incoming data (e.g., an SMS message arrived), to application events (e.g., your activity was launched from the device's main menu).

6.1.3.4.4 Services:

Activities, content providers, and intent receivers are all short-lived and can be shut down at any time. Services, on the other hand, are designed to keep running, if needed, independent of any activity. You might use a service for checking for updates to an RSS feed, or to play back music even if the controlling activity is no longer operating.

6.1.3.5 Eclipse:

- Most people know Eclipse as an integrated development environment (IDE) for Java. Today it is the leading development environment for Java with a market share of approx. 65%.
- Eclipse is created by an Open Source community and is used in several different areas, e.g. as a development environment for Java or Android applications. Eclipse roots go back to 2001.
- The Eclipse project is governed by the Eclipse Foundation. The Eclipse Foundation is a non-profit, member supported corporation that hosts the Eclipse projects and helps to cultivate both an open source community and an ecosystem of complementary products and services.



- The Eclipse IDE can be extended with additional software components. Eclipse calls these software components "plug-ins". Several Open Source projects and companies have extended the Eclipse IDE.

- It is also possible to use Eclipse as a basis for creating general purpose applications. These applications are known as Eclipse Rich Client Platform (Eclipse RCP) applications.

6.2 About EllisLab:

6.2.1 History:

We founded EllisLab in 2002 with the launch of a blogging application called pMachine Pro. A few years later we took everything we had learned from that experience and built ExpressionEngine, a powerful Content Delivery Platform, and CodeIgniter, an agile, open-source framework. Using these tools, web professionals can build nearly any type of dynamic website or application they can dream up. Today tens of thousands of people choose EllisLab software to build their sites, applications, and business with.

6.2.2 Our Business Model:

Our business model is simple: We build software for web professionals, and enthusiastically support them. We like to have fun, but we're focused about what we do.

6.2.3 Our Customers:

A common trait shared by our customers is their need to build amazing things on the web. Our software is used by freelancers, web agencies, businesses of all sizes, Fortune 100 companies, and everyone in between.

6.2.4 What is CodeIgniter?

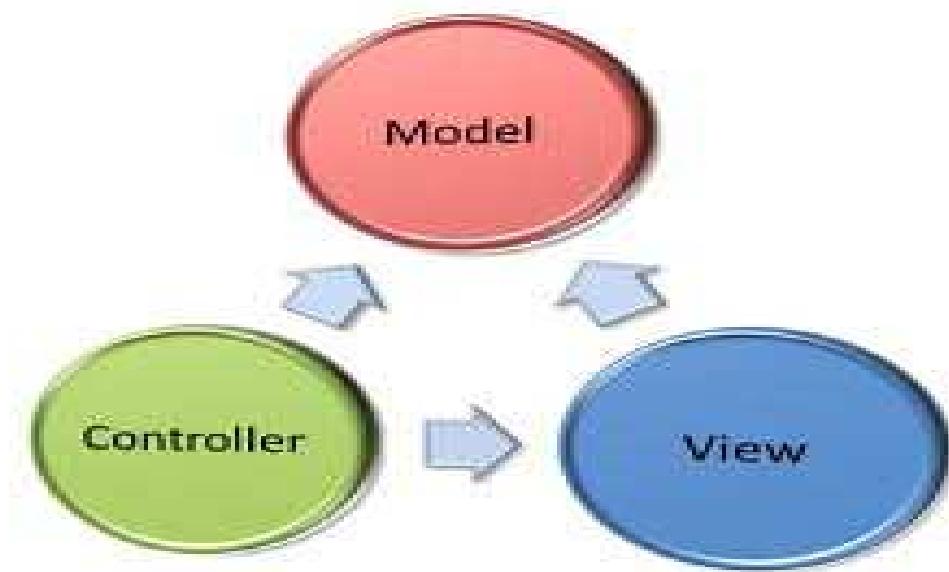
CodeIgniter is a powerful PHP framework with a very small footprint, built for PHP coders who need a simple and elegant toolkit to create full-featured web applications. If you're a developer who lives in the real world of shared hosting accounts and clients with deadlines, and if you're tired of ponderously large and thoroughly undocumented frameworks, then CodeIgniter might be a good fit.

6.2.4.1 *CodeIgniter Is Right for You if...*

- ✓ You want a framework with a small footprint.
- ✓ You need exceptional performance.
- ✓ You need clear, thorough documentation.
- ✓ You are not interested in large-scale monolithic libraries.
- ✓ You need broad compatibility with standard hosting.
- ✓ You prefer nearly zero configurations.
- ✓ You don't want to adhere to restrictive coding rules.
- ✓ You don't want to learn another template language.
- ✓ You prefer simple solutions to complexity.
- ✓ You want to spend more time away from the computer.

6.3 Model View Controller (MVC):

Model-View-Controller (MVC) is an architectural pattern used in software engineering. Successful use of the pattern isolates business logic from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other.



In MVC, the model represents the information (the data) of the application; the view corresponds to elements of the user interface such as text, checkbox items, and so forth; and the controller manages the communication of data and the business rules used to manipulate the data to and from the model.

⇒ **In simpler words:**

1. Model handles all our database logic. Using the model we connect to our database and provide an abstraction layer.
2. Controller represents all our business logic i.e. all if and else.
3. View represents our presentation logic i.e. our HTML/XML/JSON code.

6.4 MySQL:

MySQL is an open source relational database management system (RDBMS) that uses Structured Query Language (SQL), the most popular language for adding, accessing, and processing data in a database.

MySQL is noted mainly for its speed, reliability, and flexibility. It is commonly employed with most of the popular server-side scripting languages including PHP, JSP and ASP. It is a multithreaded, multi-user, SQL (Structured Query Language) relational database server (RDBMS).



MySQL is available either under the GNU General Public License (GPL) or under other licenses when the GPL is inapplicable to the intended use. MySQL is a freely available third-party database engine designed to provide fast access to stored data. Data can be stored, updated and deleted using languages such as PHP. The data can be retrieved from the database to allow the generation of dynamic WebPages.

7. Learning during Project Training:

During project training, I got lots of experience related to real life of software developer. In that duration, we learn many things with programming on live project.

Between JAN 2013 to APR 2013, I have done my application and other many task related to live project.

7.1 List of such task:

- Used of AsyncTask is very gratefully learn.
- Create demo of ActionBarShareLock which is provide menu bar like Android latest version.
- Use of Graph in android is learned from achartengine library.
- I have prepared one graph task using achartengine library which is successfully run on emulator and android 2.2 mobile but it's not working in android 4.0, this problem gives me lots of experience.

And there is many more small or big task performs during project training that gives us experience how to complete project on time and what is pressure of live application.

8. Conclusion:

This project was first experience for us in the software industries; hence it possesses very much importance in our career. I have learned basics of “Software Development Life Cycle” during our stay at “Kevalam Software”. I came across important documents of projects and get idea about importance of documentation in software industries.

From this project I came to know how to work with Surviving technologies with in deadline and work with team structure. I came to know what to do and what not to do to make project unbeaten. It also gives me benefits to understand how real worlds Project carried out in IT Firm.

I worked with some of the Team Leaders who were having lots of experience in the same field. I got idea about their responsibilities.

On the whole, the project has made me learn so many new as well as important things. The purpose of final semester training is fulfilled with this project. And above all, the things which I have learnt will be useful for all the upcoming projects.

9. Role Statement:

My role as a trainee was full of learning new things, new technology, new language, new culture, and new targets and of course a totally new definition. I got to learn many new things such as how to enhance programming speed of the project along with saving other important resources, how to achieve targets, etc.

During the phase of training, all my experiences were fortunately very good.

10. References:

1. <http://www.stackoverflow.com>
2. <http://www.developer.android.com>
3. <http://www.vogella.com>
4. <https://www.github.com>
5. <http://www.achartengine.org>