A Customer Acquisition Tool

For

Realtor using Social Media

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By
Smarth Madan
Shaunak Khedkar

Dec, 2012

APPROVED FOR THE COLLEGE OF ENGINEERING

Prof. Rakesh Ranjan, Project Advisor

Dan Harkey, Director, MS Software Engineering

Dr. Sigurd Meldal, Chair, Computer Engineering Department

ABSTRACT

A customer acquisition platform for Realtors using social media

By: Smarth Madan, Shaunak Khedkar

Real estate websites that are available in market today focus on consumers providing them search listings for properties. Real estate agents (Realtors) either have their own websites or they rely on bigger real estate companies for advertisements as wells as customer retention. These real estate platform match customer preferences with MLS data and pass the listings to consumers. There is a lack of software platform that utilizes social media techniques for customer acquisition and retention. Our project aims at solving the problem of advertising the realtor to a large crowd using social media and to manage the customer data in a smart way to convert them into potential buyers and sellers of property through realtor. The application will focus on creating a personalized page for a realtor that will give him information of his customers and potential contacts who could become customer. It will have a dashboard, which will give information about the latest happenings in the market, graphs to display the progress for the month and many more features. Social media integration with Facebook, Twitter will keep the realtors' customer involved in the process of becoming potential buyers without spamming their email accounts or unwanted phone calls. The integration with social media aims at making it easy for the customers to see the new property listings, provide feedback on it, schedule an appointment, invite for open houses, and make offers and for first time visitors a way to connect with the realtor. The demand for such an application is huge with innumerable technology inclined realtors entering the market every day.

Acknowledgments

We would like to thank our Project Advisor Prof. Rakesh Ranjan for his guidance throughout this year. We are highly indebted to him for all his support and time in completing this project. We would also like to express our thanks to Prof. Jerry Gao and Prof. Dan Harkey for their constant motivation and guidance.

Table of Contents

List of Figures

List of Tables

Chapter 1. Project Overview

Introduction

Project goals and objectives

Real estate agents (realtor) today have to be smart and fast in this competitive world to excel and make good profits. We aim at building a web application to help realtor to cope up with their competitors and grab as many customers as they can making use of Social Media for publicity. This tool will have access to social website accounts of Realtor and the realtor can manage his content on Facebook and Twitter through this tool itself. Also he would be able to manage customers who are following the Realtor's Facebook page and also interact with them using app feature of Facebook. Along customer and social content manger, the tool also aims at connecting to MLS feeds and performing search on it based on customer requirements. This would help save a lot of time for realtor and also he would be able to reach out to larger number of people using social media in their day-to-day life.

Problem and motivation

Social networking websites today are no longer mere websites for sharing and for people with same interests. They have a huge possibility of being used to get valuable business value from it. A lot of business models are present in market which relies on social networking for functioning and making money also. Our project here aims to use the social network and attract new customers for a real estate agent. The traditional ways used by realtors were spamming mails and irritating the customers by unwanted phone

calls. But with the social networking platform, the realtor can aim at a very large number potential customer without using any of the traditional methods.

According to the latest survey, Facebook has 845 million users with 200 million users added in the last year itself [1]. With the Facebook claiming the degree of separation has reduced from 6 to 4.74, people all around the globe are even more closely related. 77% consumers interact with brands on their Facebook account by reading posts and updates. Realtor can definitely use such kind of platform to publicize their availability and hence improving business profits. Some of the latest statistics are given below"

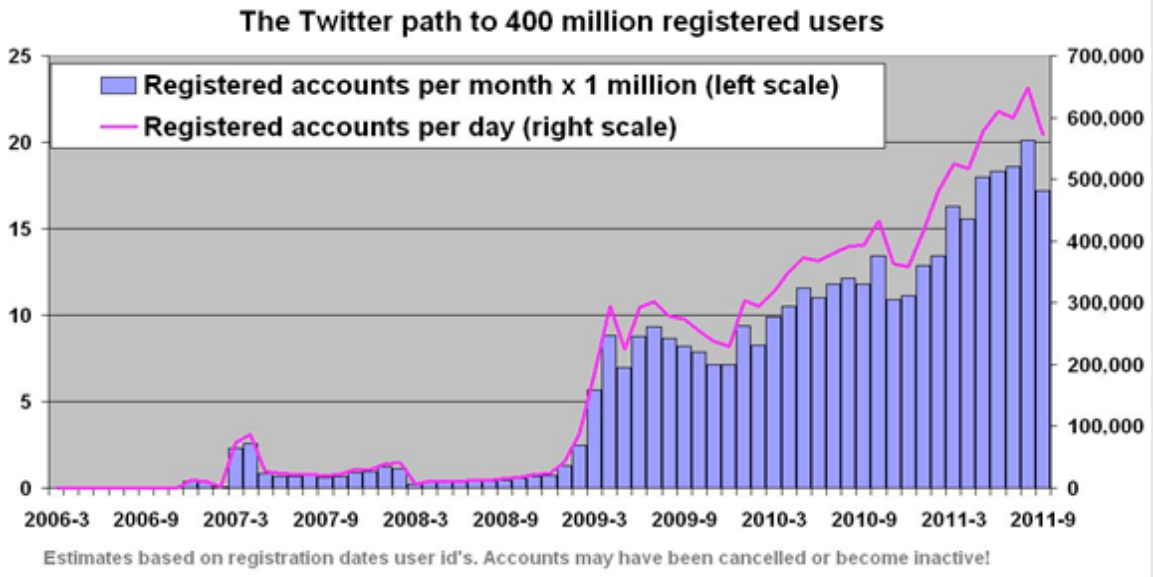


Figure 1: Twitter active users [2]

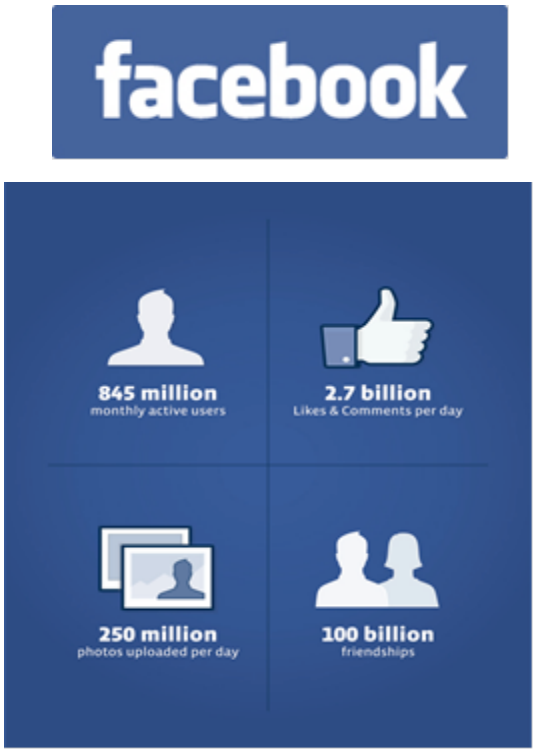


Figure 2: Facebook active users[4]

Project application and impact

The main application area of our project would be for a real estate agent to manage all their data through a web based application. This application would be fully integrated with the social websites Facebook and twitter. The realtor would be able to manage his social content such as posts as well as invitations through this application. Another aspect of this application is to connect the MLS listing system, so as to find the right customer with right property. Hence the application would be one stop point for the needs of a realtor. This would save a lot of time for the realtor and also improve his contacts with potential customers, which would eventually improve business profits.

Areas of Study and Academic Contribution

The project primarily concentrates on developing a web based application that essentially tries to integrate with social-media like Facebook, twitter and possibly LinkedIn. Thus studying the FB and Twitter API's was of utmost importance. All the social networking websites expose their API's as REST web-services due to which researching and using REST web-services was one of the major tasks while implementing the project.

The application is developed on spring social framework which is an extension of Spring Framework to provide integration with Social media. Spring social was extensively studied to ensure that it supports all the necessary functionalities the project

aims at establishing. Analyzing the framework for its support to scalability was another important area of research.

Current State of the Art:

With the increase of popularity of social networking platforms, we also see a lot of platforms coming up to manage your social content on internet. Every brand wants to reach out to as many people possible and are using social media platform for that. This means a single brand has to manage data across various websites like Facebook, twitter, LinkedIn, foursquare, etc. Following is a statistics as per January 2012 for the top 10 social-networking websites:

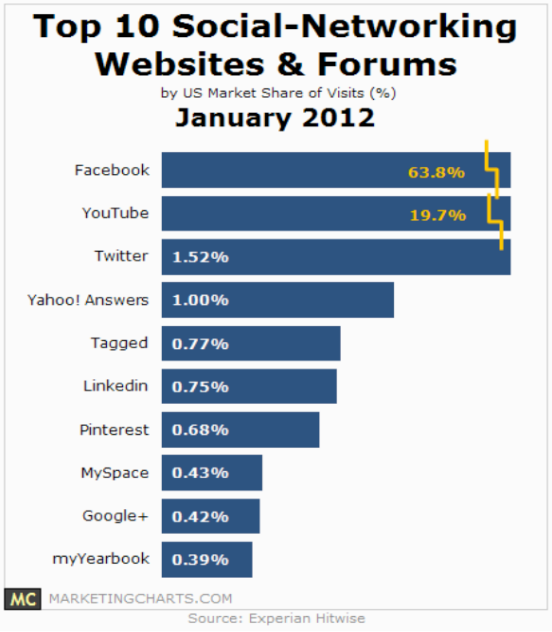


Figure 3: Top 10 social networking websites[2]

Following are some of the products present in the market currently which focus on Social media management for realtors along with a comparison chart to differentiate our application from the existing ones:

a) **Shoutlet[8]**:

This application is an enterprise marketing platform for publishing, engaging and measuring social activity. It currently supports publish content on Facebook, twitter and YouTube only. The company is based in Madison, WI. It has features of email marketing and mobile also.

Figure 4: Shoutlet[8]

b) Social Bios[9]

This application is a social search platform and part of Move, Inc. It links an individual, or company's website or social page and performs useful search based on the visitor of the page. This includes, no. of common friends, status updates, etc. You can add Facebook, Twitter, LinkedIn, foursquare and gmail accounts for performing searches.

Figure 5:socialbios[9]

c) Market Snapshot[6]

This application focuses on giving you the market idea of the neighborhood real estate.

It tells you some of the curious questions a customer might have like how is the market, when should I sell my house, is it good time to invest, etc. This application is useful to real estate agents as they would be able to provide data for particular neighborhood in no time.

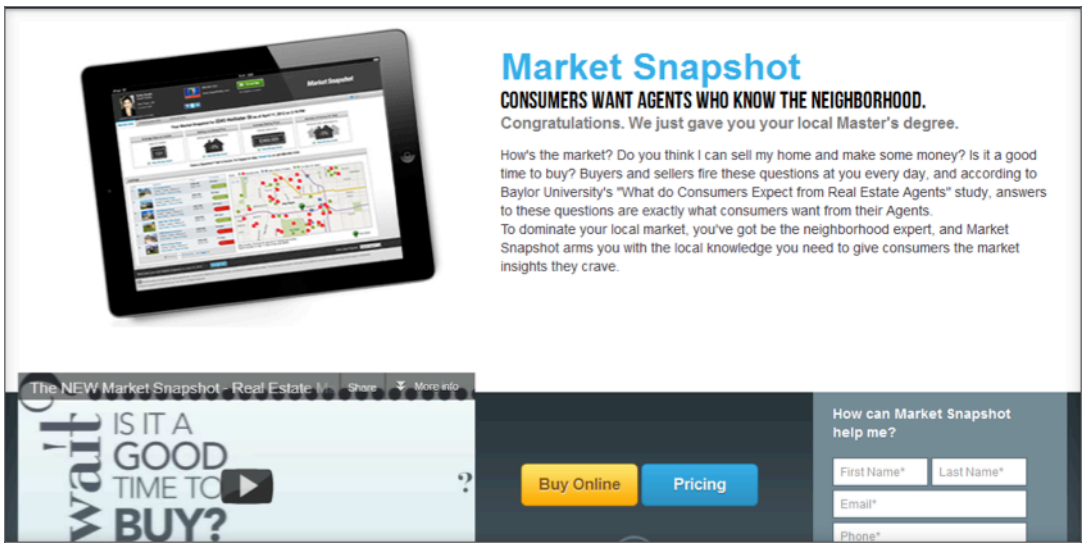


Figure 6:Market Snapshot[6]

d) Web Real estate marketing[11]

This is tool for real estate agents to start their own blogs. This way they can reach out to many people on internet and prospective customers would contact the realtor based on his blog. This tool helps you to organize your blog and publish it on social media platform. Also it gives suggestion on web searches.

Figure 7:Web real estate Marketing[11]

Comparison:

	Social analytics	Publish content	Integration with various websites	Customer interaction	Real estate Listings	Real estate Analytics
Shoutlet	✓	✓	Facebook, Twitter, Youtube	✓	✗	✗
SocialBios	✓	✗	Facebook, Twitter, Foursquare, gmail, LinkedIn	✗	✗	✗
Market Snapshot	✗	✓	Facebook	✗	✗	✓
Web Real estate marketing	✗	✓	Facebook, Twitter, Wordpress, Youtube	✗	✗	✗

Table 1: Comparison of social tools present in the market

From the above comparison carried out, there is not tool in the market which would manage the customer data, along with social content. Also the same tool which would perform analytics and also integrate with MLS listings.

Chapter 2. Project Architecture

Introduction

Our project is based on JEE platform and we are making use of Spring framework. Spring framework leverages features such as authentication, security, MVC architecture that we are automatically inheriting in our system. For data persistence, we are using MySQL database, as it is most widely used open source database. Also all languages have good integration with MySQL, which won't be of any issue to us if we decide to switch to different framework or language in future. The three tier of our system are:

1. Client – Our Client (i.e. Realtor) can view the tool running on web server through his browser. We will be rendering JSP pages which would be making use of HTML5, Jquery javascript framework, and CSS3 to give a rich and interactive UI.
2. Server – We will be running our server on Amazon EC2 micro cloud. Our application will be deployed and running on Apache tomcat 7 container. The client will be sending REST based calls to the server and server will handle them using Spring framework MVC architecture and return appropriate view for the calls.
3. Database – We shall be using MySQL as persistence model for storing information regarding Customers, MLS listings and also other tables, which are required for running of our app.

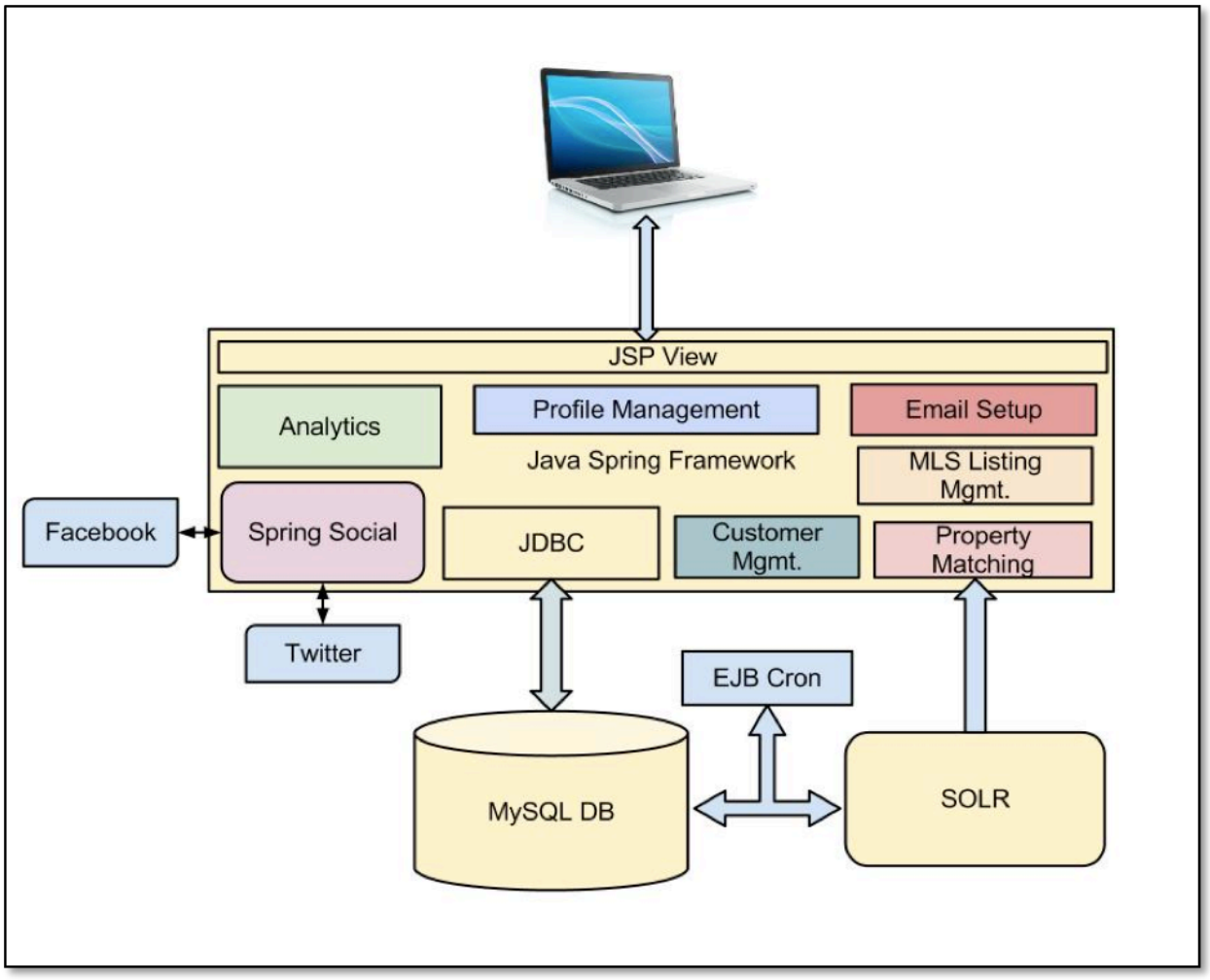


Figure 8: System Architecture

Architecture Subsystems

Following are the Subsystems of our architecture:

1. **Spring Social** – This is the most important module of our system. We are making use of Spring Social plugin, which is an open source plugin, made available springsource.org for connecting to various social media platforms available. Although it leverages connections to various platforms such as LinkedIn, Facebook, twitter, etc but we are making use of Twitter and Facebook only keeping the nature of our

project. Since we are focusing on publicity of the realtor, Facebook and twitter are perfect platform to reach out to large amount of people.

2. **Analytics** – We want the tool to also perform analytics on social media such as Facebook and keep the realtor informed on metrics such as no. of likes per month, no. of new visits to his page, etc. This would help him to keep a track on how useful this tool can be and also keep him engaged.
3. **Profile Management** – This module focuses on providing features for the realtor related to his personal profile. This would include his basic information, social media account details and email preferences. Once the realtor has setup these setting, he can use full features of the tool without re-entering credentials every time he/she wants to perform any operation requiring external authentication.
4. **Email Setup** – This module is responsible for setup an email client for the realtor that will automatically trigger an email send request when there are potential match for MLS listings for a particular customer requirements. The email will be sent as if it is a personal email from the realtor's account to the customer's email account. This would keep the customer informed about listings which he in particular could be interested in.
5. **MLS Listing Mgmt**. – This module will manage the MLS listings as well as tagging feature for the realtor. Tagging is a feature that will help the realtor to associate specific keywords as tags to a property listing. These tags will also play an important role in matching the listings to customer requirements. For our project MLS listings is currently stored in our MySQL database but ideally these listings are available to real estate agents from MLS website. Also special subscriptions are available which can be bought as per monthly basis.

6. **Customer Mgmt** – This module is responsible for managing the customers for the realtor. A realtor can search customers, add new customer or update the information of a customer. Also customers have priority associated to them. If the customer contacts the realtor himself, the priority of that customer is high. Over a period of time if the communication does not take place, the priority can be reduced by the system.
7. **Property Matching** – This feature is a special algorithm that would come up with listings matching a customer's requirement. This matching would be based on various factors such as keyword search from customer's description of his dream house, zip code matching and certain parameters that we can retrieve from the customer's Facebook profile. This module is the heart of our system without which the system is not of much importance.
8. **JDBC** – This is a helper module to connect to MySQL database and facilitate data transfer to and fro the database.
9. **SOLR** – Solr is an open source text search server, which can perform search on very huge amount of data, and return matching records quickly. Its used as data mining tool and it is used mainly because of its high efficiency. We are using Solr to index data of MLS listing as search matching data of customer requirements as quickly as possible.

Chapter 3. Technology Descriptions

An attempt has been made to use the latest technologies at all the 3 tiers-Client, middle and Data. The choice of technologies used is based on the requirements, the architecture of the system and their flexibility to adapt to changes. The technologies used at each level have been described as follows:

Client Technologies:

Customer acquisition through social media is web based tool that runs in a browser. The browser's supported for this application in Google Chrome, Safari and Firefox. The client has been developed on JSP's that run on server. JSP's form the base on top of that we have used latest features of HTML 5, CSS 3 and Javascript.

We have used 'Bootstrap' for improving and enhancing the UI. Bootstrap is a UI framework that provides a bunch of UI templates to choose from. It provides a way to have a beautiful, esthetic yet professional and most importantly a consistent look to your website. It is easy to use and there is a lot of documentation and examples to guide.

The web based tool displays graphs to show the analytics. For the graphs 'FusionCharts' have been used. FusionCharts is an easy way to convert your data into more readable and easy to understand graphs. It uses Flash to display the graphs and requires data in the form of XML files or JSON objects. It is very easy to use and integrate with the system.

The system also sends formatted emails to the customer's Gmail ID for which it uses HTML 5, CSS 3 and uses the email templates.

Middle-Tier Technologies:

The middle tier is the heart of the Application. This is where all the business logic goes. To ensure correctness and quality the technology chosen for middle-tier had to be one that is sound and provides a strong, flexible framework to build upon. Spring-Social Framework provides one such platform. Spring social is essentially the same old Spring framework that uses principles of Dependency Injection. Spring-social framework provides a way to build flexible and loosely coupled web applications that allow you to connect with the social networking SaaS providers like Facebook and Twitter. Spring Social makes it easier to connect with Facebook and the framework already provides sharing of data. It provides most of the Graph API functions. The support to Twitter is also good but lacks a little when it comes to LinkedIn as LinkedIn hardly provides any data publicly. Spring social here uses Constructor based injection which makes the communication between controllers, helpers and POJO's extremely easy and keeps them loosely couples.

For displaying maps the Fusion charts require data in form XML documents, so the utility classes create XML files. For matching the customers with Properties we have integrated Apache Solr with the spring framework. Apache Solr is used for efficient searching of properties. A nightly Cron job runs that puts the Data from MySQL into Apache solr so as to keep it updated. The application is build using Maven and the war file is deployed in Tomcat container (Apache Tomcat 7).

Data-Tier Technologies

At the data tier level MySQL is used as the Relational Database. MySQL was chosen for ease of use and ease in integrating it with Spring Social framework. The relational database that has been created has some of the following entities as tables:

- Customer
- Realtor
- C_Requirements (Customer Requirements)
- MLS_Listings (For properties)
- Tags (Tags added to various properties)
- Userconnection (To manage connections/sessions)
- Zip_codes

Apart from that at the data tier level we have used Apache solr for efficient searching of properties for customers. The data to Apache Solr is made available as XML files, which get populated by a cron job. The indexing is done for better search results.

Chapter 4. Project Design

Our project is based on Spring MVC architecture.

Client Design

Our client is based on JSP(JavaServer pages). As the name suggests these pages are compiled on the server side and the created markup is rendered on the client side. The controller provides model which contains the data to be shown in the view. This model is passed to JSP page and they utilize it using special syntax like "${<model_name>}". Also it is possible to do computation and scripting using delimiters. Some of the most commonly used are: <%... %>, <%=…%> and <%@...%>.

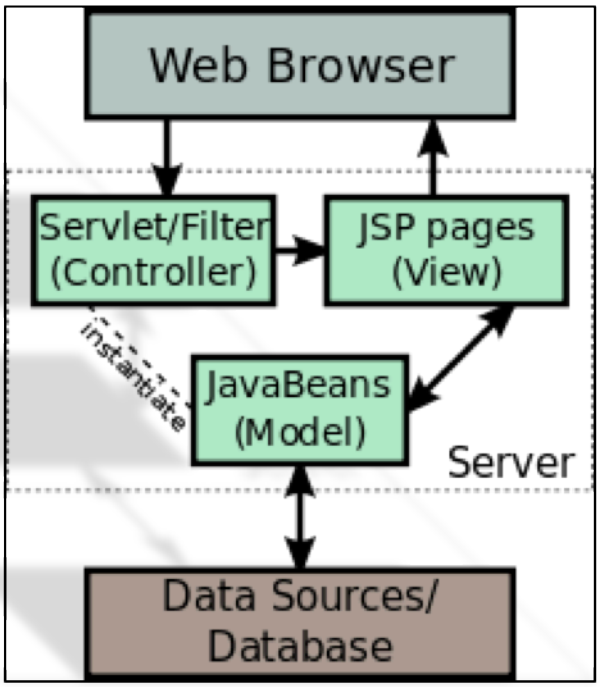


Figure 9: JSP view[12]

Our client design also include rich html5 user interface with CSS3 styling. To havea glimpse of the UI please refers to following screenshots.

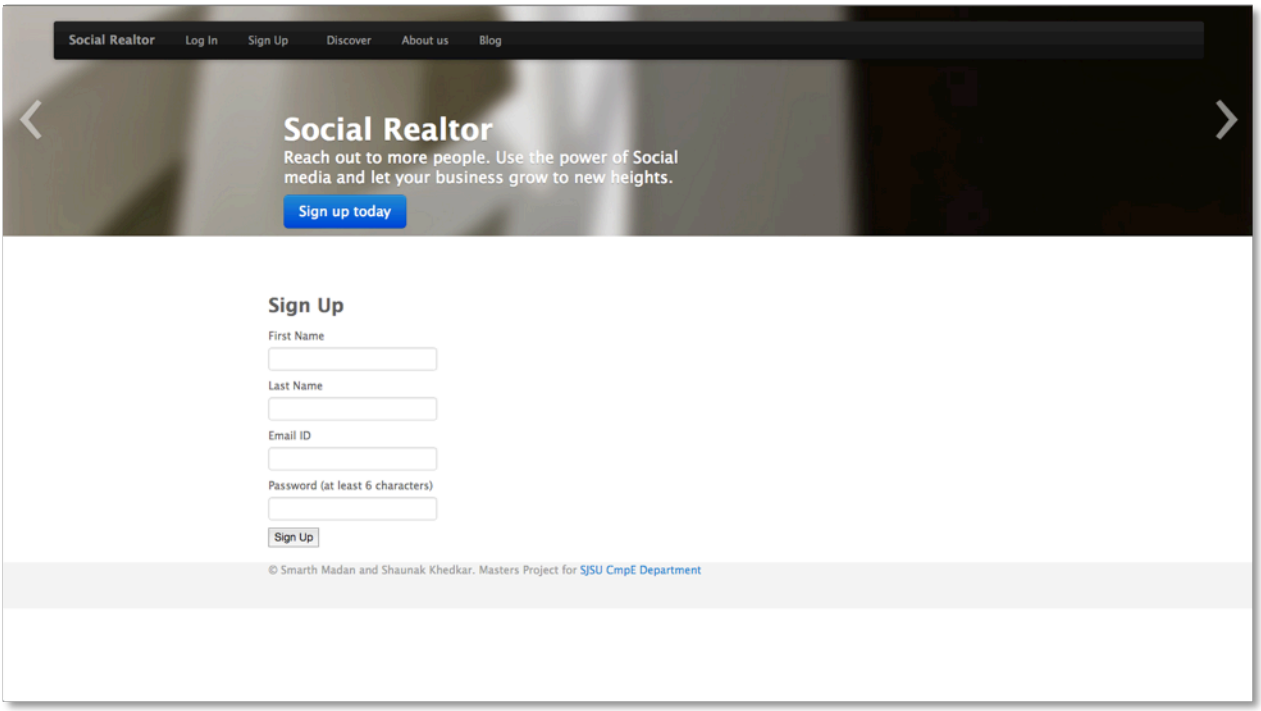


Figure 11: Screenshot of sign up page

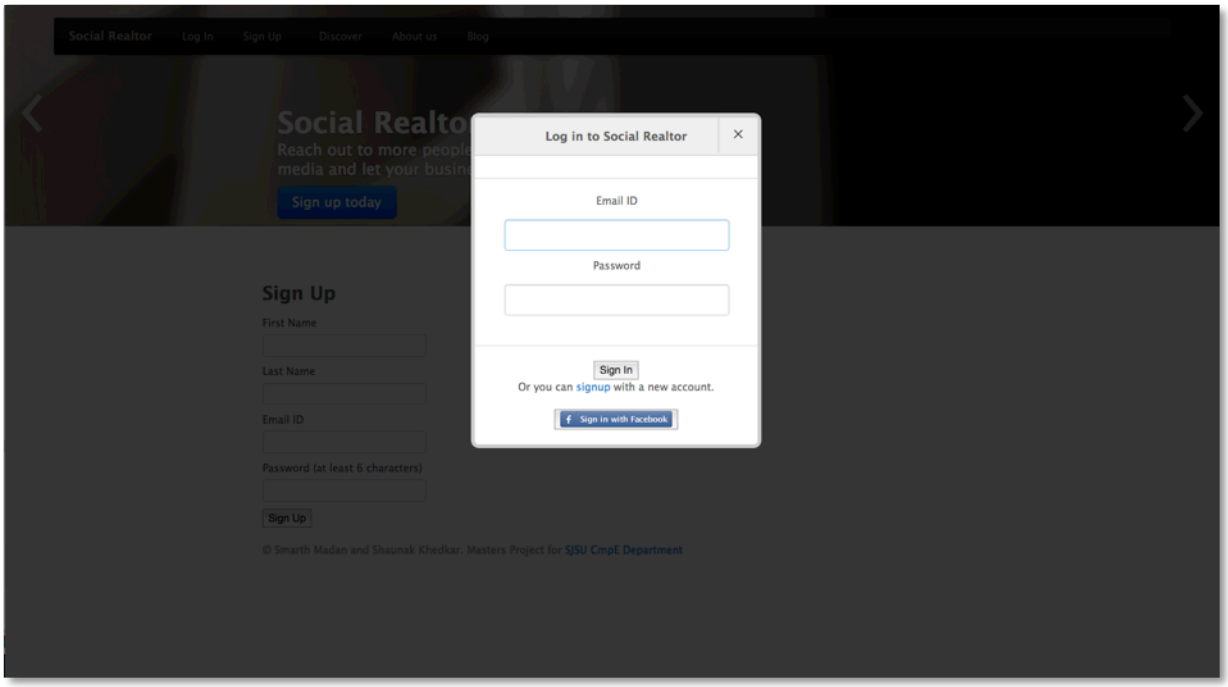


Figure 10: Screenshot of Signin page

Middle-Tier Design

Our Middle-tier components are basically Spring based java controller classes. These controller are mapped to particular end points urls with certain rules and parameters. Whenever a request is sent by client on url, the corresponding method in the controller class is executed.

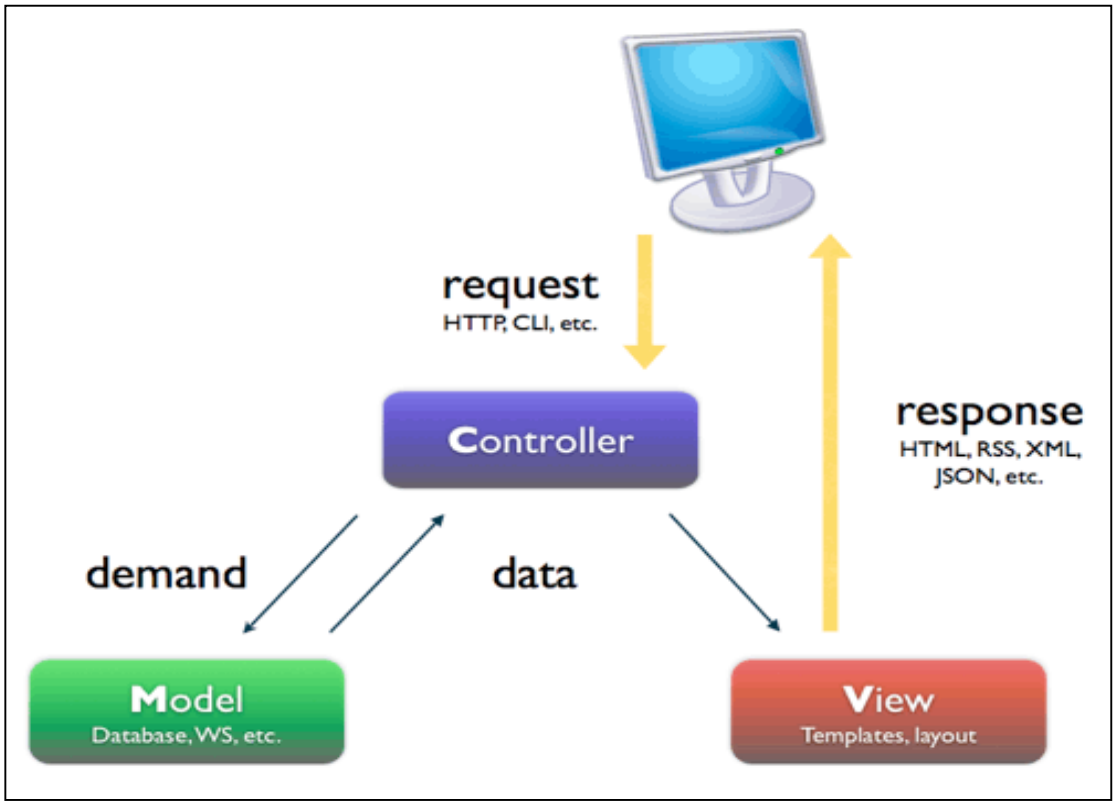


Figure 12: MVC architecture[13]

Following are few of the use cases in our application explained with the help of UML diagrams:-

1. **Customer Acquisition –**

This use case is the most important use-case of our project where the potential customer visits the Facebook page tab of realtor's page and fills out a short form of his requirements along with his details. Once he does that, the realtor gets his information on the tool from where he can query the MLS listing and find suitable property listings for that customer. On finding suitable matches, Realtor can also email the list to the customer.

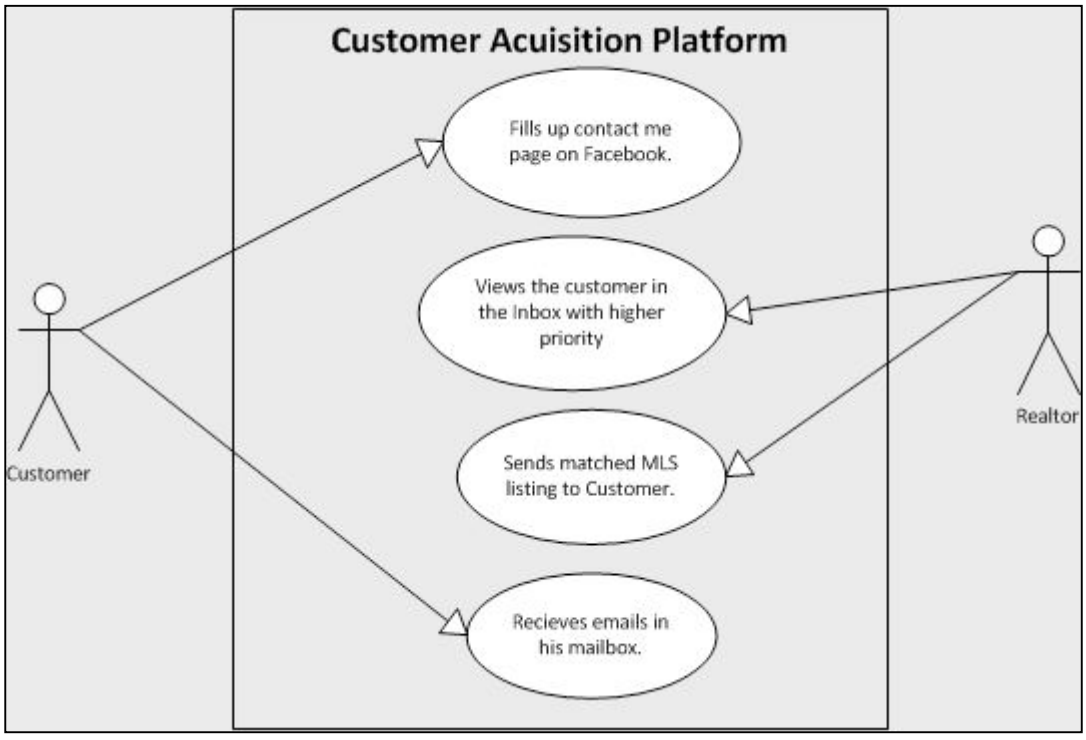


Figure 13: Customer acquisition use case

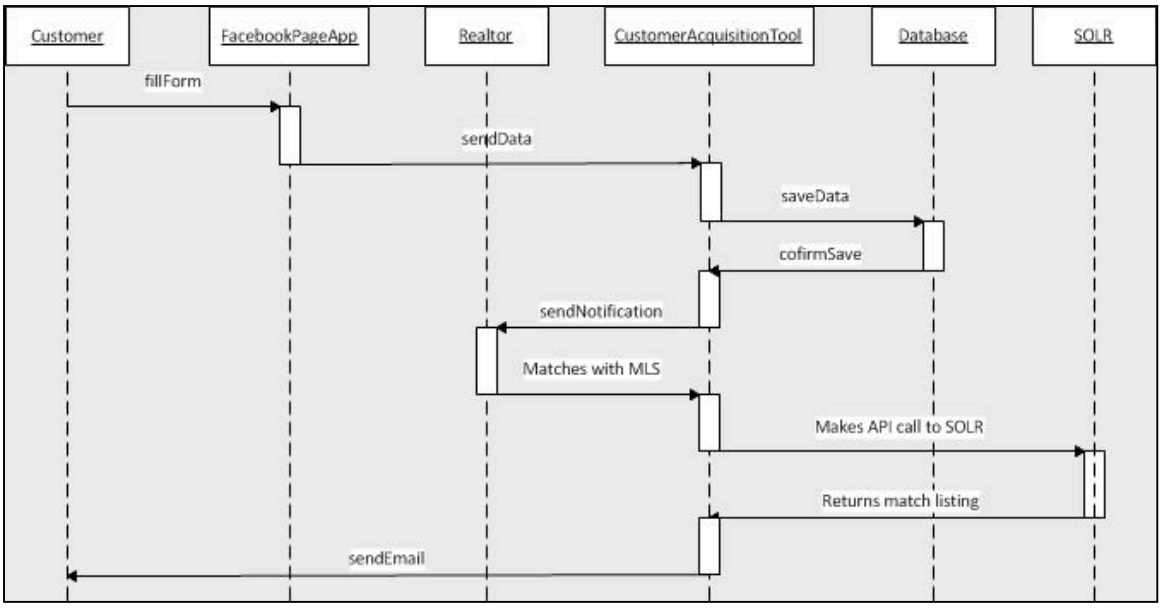


Figure 14: Customer acquisition sequence diaggram

2. Analytics -

This use case showcases the ability to check analytics of Facebook page by the realtor. The realtor can login to the tool and check various insights related to his Facebook page such as no. of likes, no. of unique users, no of new fans per week or month. Whenever a New customer visits the Facebook page or likes the page, Facebook keeps a track of such counts and provides a large amount of insights.

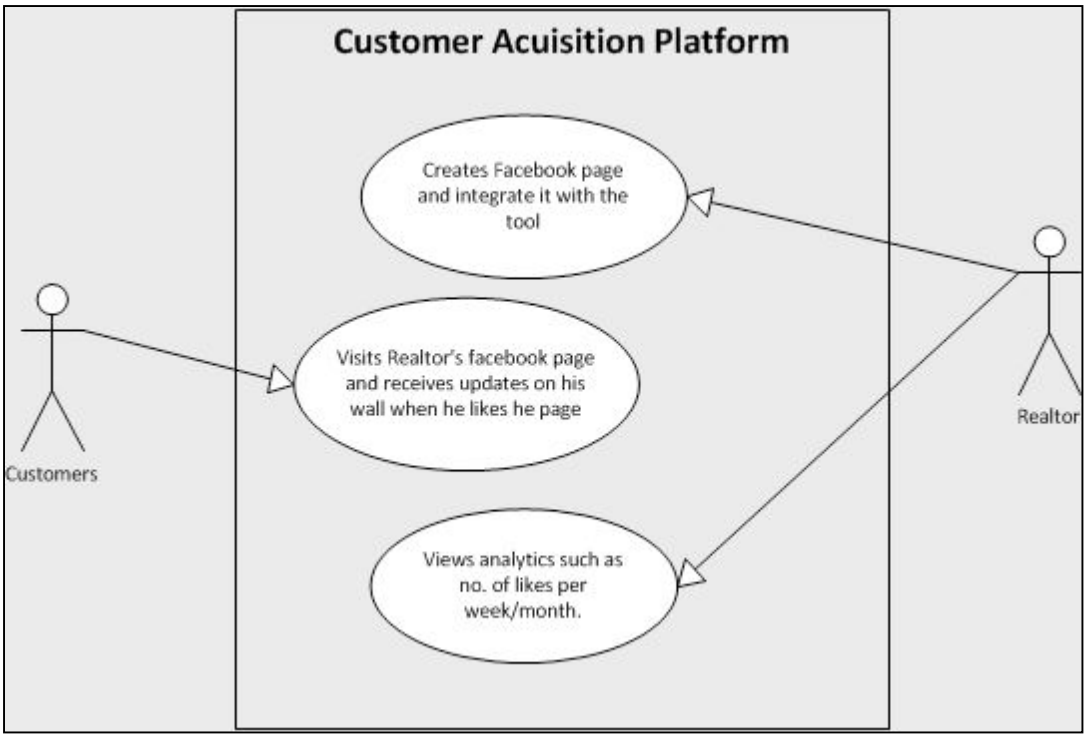


Figure 15: Analytics use case

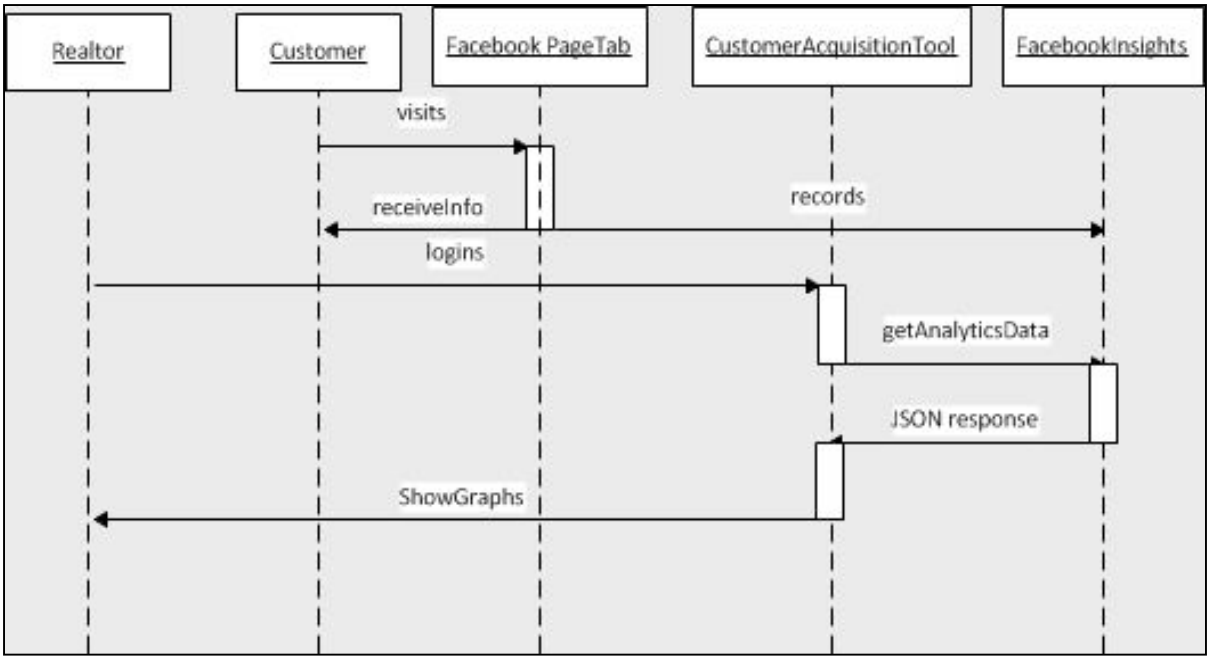


Figure 16: Analytics sequence diagram

Following are the class diagrams of few of our important classes: -

1. **Customer Controller**

Customer controller is an important class, which is responsible for handling all the requests coming from UI related to customer. Features such as Adding new customer, viewing all the customers, viewing top 5 customers, editing customer information and matching MLS listings will be handled by this controller. As we can see from class diagram shown below, customer controller Class uses help of CustomerHelper.java for all interaction with the MySQL database. Row Mapper classes are used to map the result set of MySQL query to POJO(Plain Old Java classes). Once the data is mapped to java classes, it is easier to use them and pass them as objects.

SolrHelper.java class is responsible of matching the customer requirements to MLS listings. SolrHelper.java utilizes the MySQL database to query all the tags associated to the property and also SOLR java API to query SOLR server directly.

FacebookPageController.java is used to handle all the requests, which are related to Facebook. It makes use of the Spring social showcase framework to make graph API call. The realtor must be logged in to make use of graph API for facebook.

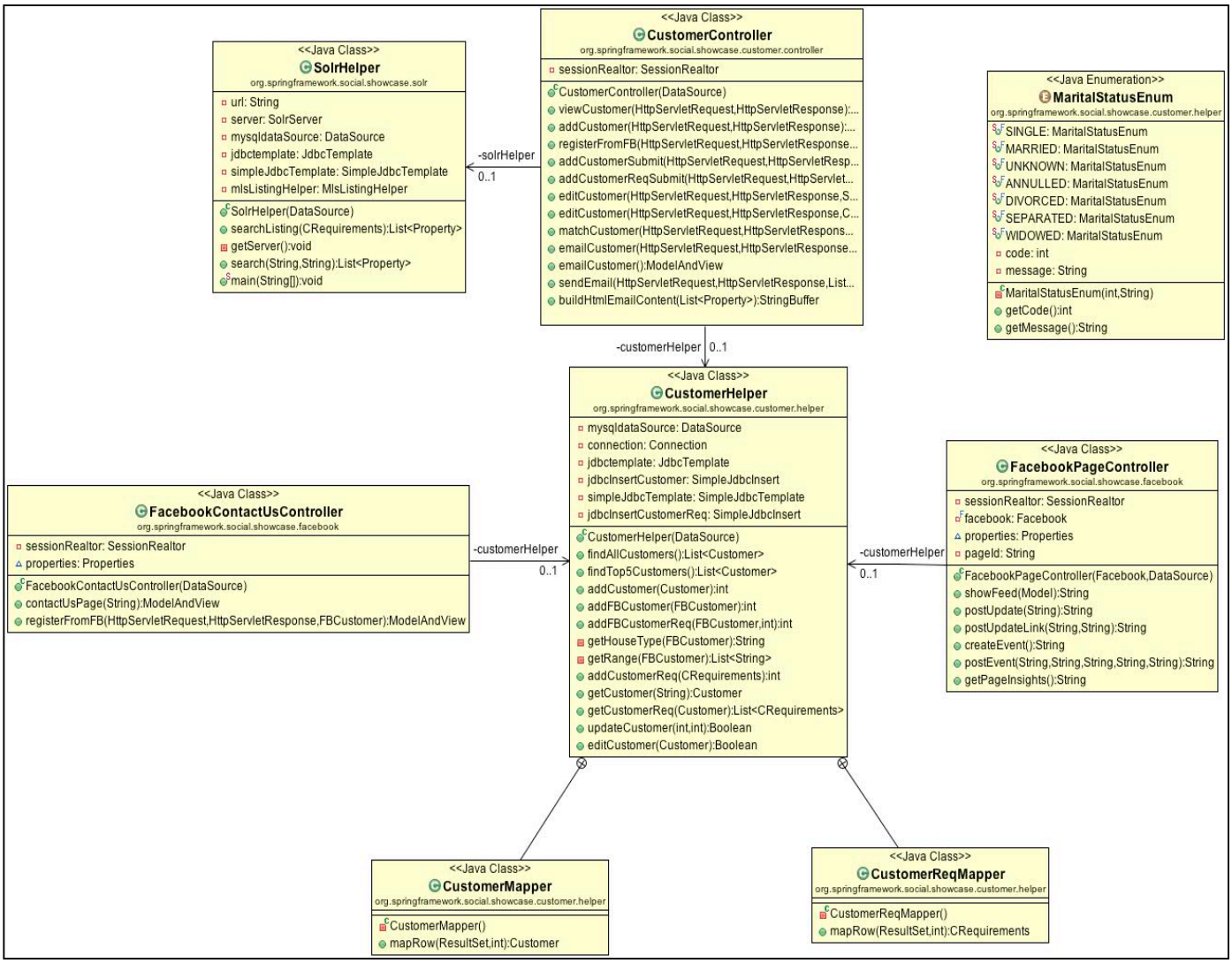


Figure 17: Customer Class diagram

2. Property Controller

PropertyListingController.java is responsible for handling requests related to ML listing. As we are faking our data for MLS currently, this class makes use of MLSListingHelper.java to contact MySQL database. TagMapper class is used to map each row returned from MySQL to a java object. These tags are associated to particular property and help in returning more customized result on searching properties for a customer. PropertyMapper class similarly maps a row in property table to its corresponding Property.java object.

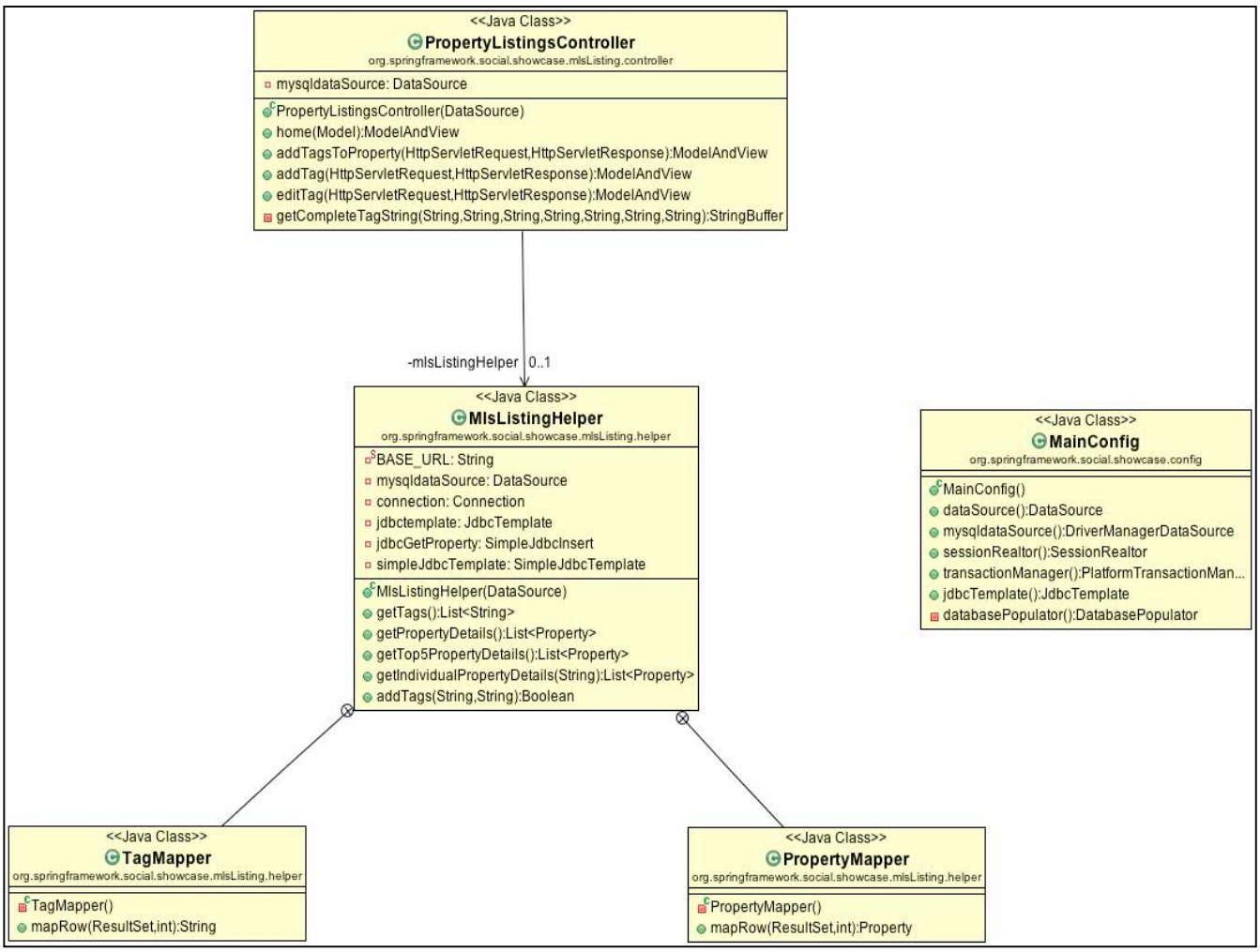


Figure 18: Properties Class diagram

Data-Tier Design

Below given is the ER diagram of our Database, which is based on MySQL server:

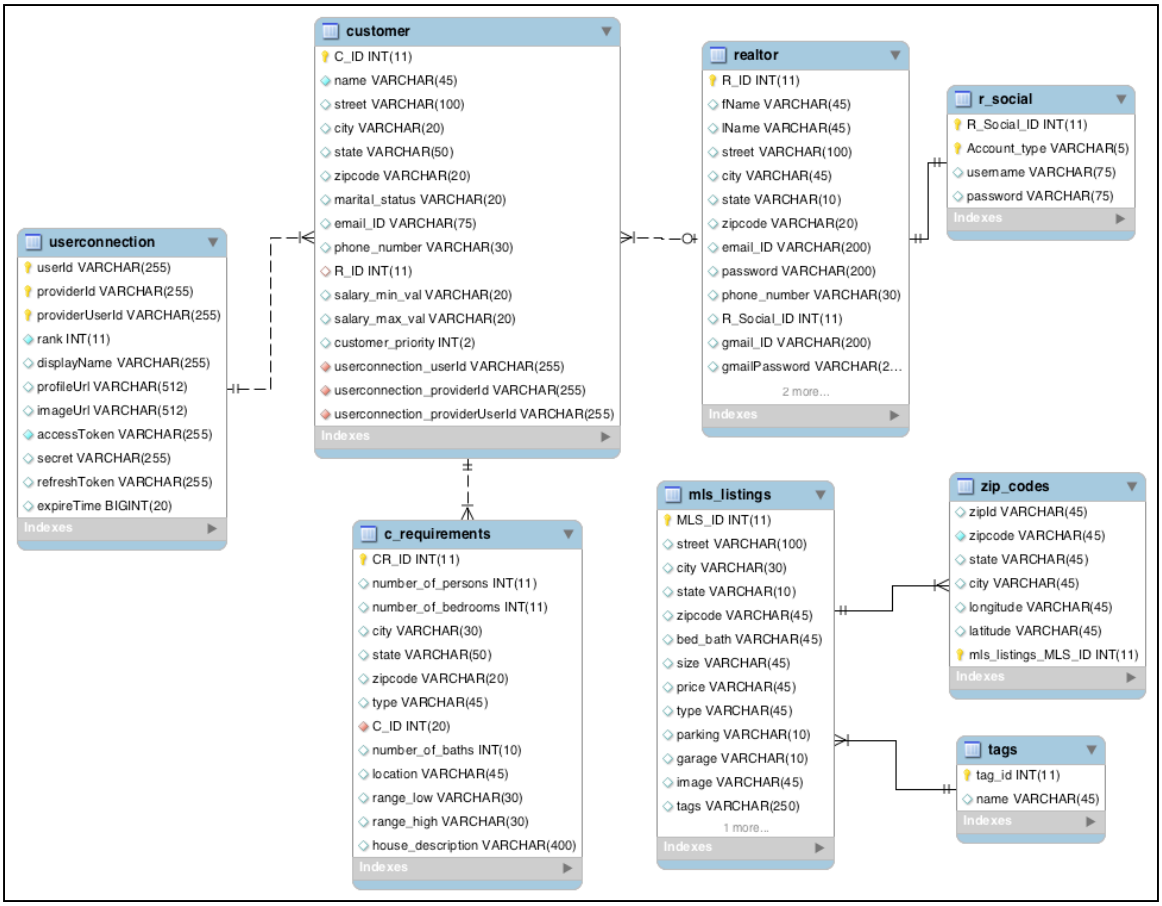


Figure 19: MySQL ER Diagram

Customer : This table holds the data for all the customers in the system for all the realtors signed up for the tool. A customer for one realtor is differentiated from other realtor using the Realtor ID, which it is associated to. A realtor can only view, edit or add customers with his own unique ID and there is no possibility for accidentally accessing other realtor's customers.

C_requirements : This table hold information regarding the customer's requirements for the type of property he is looking for such city, state, range, and more description.

User.

UserConnections : This table holds value for the realtor's social connections. If the realtor has associated his/her twitter or Facebook account, there would be an entry present in this table. Also every time there is a call to be made to fetch some data from social media, the access token stored in this table it used to make the API calls for authentication purpose.

MLS_Listings : This table contains fake data of MLS. Since MLS listing subscription is only provided to Real Estate agents with proper certification, we decided to tackle this problem by faking the data but keeping it as realistic as possible by going through all the property websites.

Realtor : This table contains profile data for realtors. Each realtor has a unique row and ID associated to it. This realtor ID is also associated to his customers. Realtors' social accounts are managed in different table called "**R_social**".

Chapter 5. Project Implementation

The customer acquisition tool has been primarily coded in Java language. Along with Java; HTML 5, CSS, Javascript, SQL and XML are the other languages which have been used in this project. HTML 5, CSS, Javascript have been used for frontend development while the middle tier is all Java and XML and the backend uses SQL. Following are some of the important pieces of programs from the project.

Features

Home: The homepage is where it all starts. The home page primarily concentrates on showing up the most important things to the customer. It displays the top 5 customers for the realtor. The top 5 customers are being chosen for the realtor based on the priority assigned to the customer and also the date when the customer was included into the realtor's tool. The priority is again based on the users Facebook activity whether he has requested for an appointment and some other factors like his date of joining.

Similarly it also displays the top 5 real estate properties for the customer. Again the top 5 properties are chosen from the criteria of the ones that have been listed the latest. The home page also displays the graph for the analytics. The analytics are typically based on the data fetched from Facebook. Like for example, the graph shows the number of posts on the page for a period of one week. It also redirects to the analytics page if the realtor wants to see more of the analytics.

The other most important part of home page is that if the user is not already logged into Facebook, it exquisitely asks the user/realtor to login to as a lot of data is being coupled from Facebook. Some of the important pieces of code are described below.

Client Implementation

Some of the important pages, Javascript, css and xml files are discussed below. The client has been decorated with Bootstrap. Bootstrap is UI framework that gives beautiful, aesthetic yet professional and consistent look.

Page.jsp

This forms the base page. All the other JSP pages are loaded as Tiles in this page. The menu.jsp has been included in this page that provides all the dropdowns.

```
<html lang="en">
<head>
<script type="text/javascript" src="<c:url
value="/resources/datetimepicker.js" />"> </script>
<script type="text/javascript" src="<c:url
value="/resources/js/FusionCharts.js" />"> </script>
<link rel="stylesheet" href="<c:url
value="/resources/messages/messages.css" />" type="text/css"
media="screen" />
<link rel="stylesheet" href="<c:url
value="/resources/bootstrap.min.css"/>“ media="screen">
</head>

<body>
  <div class="container"> <tiles:insertAttribute name="content" /></div>

</body>
```

As highlighted above the two javascript files:

- **Datetimepicker.js** is used for popping up a calendar for date selection
- **Fusioncharts.js** is used for rendering graphs onto the page. Fusioncharts also uses flash to load the graphs in the webpage.

Further the 2 CSS files:

- **Messages.css** is used for formatting the messages, texts
- **Bootsrap.min.css** is used for the overall decoration of the website. It is the CSS file used from Bootstrap which is an open source framework for building UI.

```
The <div class="container"><tiles:insertAttribute name="content" /></div>
```

is the place where other jsp's are loaded as tiles**.**

- **Tiles.xml:** This is the file where mapping for views is done.

```
<tiles-definitions>
<definition name="analytics/analytics" extends="page">
<put-attribute name="content" value="/WEB-
INF/views/analytics/analytics.jsp"/>
</definition>
</tiles-definitions>
```

name='analytics/analytics' is the path that is set in controller and here it is mapped to a view like analytics.jsp

- **Data.xml**: The data for maps is provided as an XML file.

```
<chart> caption='Weekly Sales Summary'
  xAxisName='Week' yAxisName='Amount' numberPrefix='$'>
   <set label='Week 1' value='14400' />
   <set label='Week 2' value='19600' />
   <set label='Week 3' value='24000' />
   <set label='Week 4' value='15700' />
</chart>
```

Home.jsp

This is home page of our application.

```
<div class="row">
<!-- <div class="span12"></div> -->
<div class="span12" style="width:100%">
<div class="row">
  <div class="span6"><h1>Welcome, <c:out
value="${account.firstName}"/>!</h1></div>
  <div class="span6"><c:if test="${not empty image.imageUrl}"> <img
src="${image.imageUrl}" height="100" width="100" style="float:right">
</c:if> </div>
</div>

<table style="width:100%">
       <tr ><td>
              <b> Your top 5 Customers : </b><hr />
              <table style="float:left" class="formatHTML5" >
                     <tr style="background-color:grey"><td><b>C
ID</b></td><td><b>Name</b></td><td><b>Email ID</b></td></tr>
                     <c:forEach items="${top5CustomersList}" var="customer">
                     <tr >
                            <td ><c:out value="${customer.c_id}" /></td>
                            <td ><c:out value="${customer.name}" /></td>
                            <td ><c:out value="${customer.email_ID}" /></td>
                     </tr>
                     </c:forEach>
              </table>
<tr><td><b> Analysis: <hr /> </b></td></tr>
<tr><td> <div id="chartContainer">FusionCharts XT will load
here!</div><center><a href="<c:url value="/analytics"/>" > Click here for
More Analytics..</a></center></td></tr>
```

Middle-Tier Implementation:

Some of the important classes are discussed below

- **SocialConfig.java:** This class obtains the connections with the various social media

```
@Bean
@Scope(value="singleton", proxyMode=ScopedProxyMode.INTERFACES)
public ConnectionFactoryLocator connectionFactoryLocator() {

ConnectionFactoryRegistry registry = new ConnectionFactoryRegistry ();

registry.addConnectionFactory(new
TwitterConnectionFactory(environment.getProperty("twitter.consumerKey"),

     environment.getProperty("twitter.consumerSecret")));

registry.addConnectionFactory(new
FacebookConnectionFactory(environment.getProperty("facebook.clientId"),

     environment.getProperty("facebook.clientSecret"))); }.
```

- **HomeController.java:** The controller that handles home feature.

```
@Controller
public class HomeController {

   @Autowired
   private SessionRealtor sessionRealtor;

   @Inject
   public HomeController(Provider<ConnectionRepository>
connectionRepositoryProvider, AccountRepository
accountRepository,DataSource mysqldataSource,Facebook facebook) {
          this.facebook = facebook;
          this.mysqldataSource = mysqldataSource;
          mlsListingHelper = new MlsListingHelper(mysqldataSource);
          this.connectionRepositoryProvider = connectionRepositoryProvider;
          this.accountRepository = accountRepository;
          customerHelper = new CustomerHelper(mysqldataSource);
          facebookHelper = new FacebookHelper(facebook);
   }

   @RequestMapping("/")
   public String home(HttpServletRequest request, Principal currentUser,
Model model) {
```

```
            Account account =
    accountRepository.findAccountByUsername(currentUser.getName());

            model.addAttribute(account);
            sessionRealtor.setRealtorId(account.getR_ID());
            Map<String, List<Connection<?>>> connections =
    getConnectionRepository().findAllConnections();
            model.addAttribute("connectionsToProviders", connections);
            try{
                  model.addAttribute("image",
    getConnectionRepository().findAllConnections().getFirst("facebook"));
            }catch(Exception e){

            }

            model.addAttribute("graphJsonData", createdJsonSTring);
            //return "home";
            if(connections.get("facebook").isEmpty())
                  return "forward:/connect/facebook";
            else
                  return "home";

       }
```

Modules

Profile:

The "Profile" feature is essentially the feature to manage the realtor's personal data. It allows for editing the data if any information is incorrect or missing. It also has the feature to register realtor's Facebook page. All it requires the page id for the page. The Face page ID is found in Facebook page> update info. This is data is pretty important if he has to have sort of meaningful interaction with the Social media.

This feature also has a way to setup your gmail ID. Setting up your gmail ID is important because currently it the only way to support email functionality.

My Profile Page:

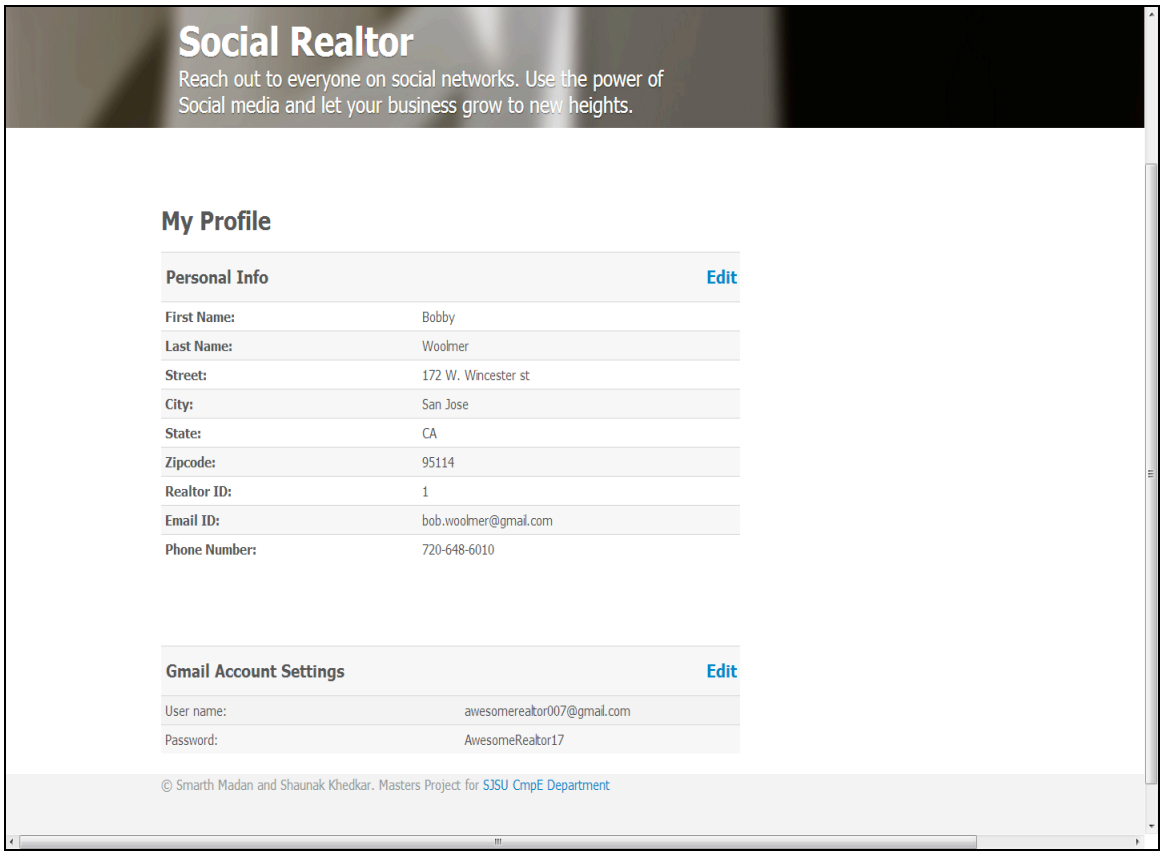


Figure 20: Profile page

Customer:

The customer feature is one of the most important features. It displays all the customers and their details. The customer details can be edited. Further we can add more customers. Along with the customer details like his name, address and contact details, the tool also allows gathers information like Salary range, the customers expectation of the house like number of bedrooms, baths, location, range of amount for the house.

The other important feature of the customer is the matching of properties with the customer. Based on the criteria provided by the customer the properties are matched. The properties are also searched based upon the tags that have been added to a property. To make the matching efficient, the data has been stored as XML files in Apache Solr. Here we have created indexes which make the searching really fast. Following are some of the code snippets:

```
public class SolrHelper {

	private String url = "http://localhost:8983/solr";

	public SolrHelper(DataSource mysqldataSource) {
		this.mlsListingHelper = new MlsListingHelper(mysqldataSource);
		getServer();
	}

	private void getServer(){
		try {
			server = new CommonsHttpSolrServer(url);
			((CommonsHttpSolrServer) server).setSoTimeout(1000;
			((CommonsHttpSolrServer)
			((CommonsHttpSolrServer)
server).setMaxTotalConnections(100);
		} catch (MalformedURLException e) {
		System.out.println("Could not get the Solr Server!!!!");
		}
	}
private String tagsAndCity(List<String> tags, String city){
		String solrQuery ="";
```

```
            if(!tags.isEmpty()){
                  solrQuery = "!tags:(";
                  for(String tag : tags){
                        solrQuery = solrQuery + "*" + tag+ "*" +" OR ";
                  }
                  solrQuery = solrQuery.substring(0, solrQuery.length()-4);
                  solrQuery = solrQuery + ") ";
                  solrQuery = solrQuery + "AND ";
            }
            solrQuery = solrQuery + "city:("+city+")";
            return solrQuery;
      }
}
```

Apache Solr runs on port 8983. As shown in the above code we create solr queries to match the properties.

Once the properties are matched for a particular customer, these properties can be emailed to him. For emailing the application uses Gmail's SMTP server, for this reason the user will have to provide his Gmail ID and password. This username and password can be stored under My profile> Personal Info > Gmail Account Settings.

MLS Listings:

This feature is at the heart of our application. The MLS listings are the properties that are available. The details of the properties such as address, size, and number of bedrooms and so on are displayed. This feature allows the realtor to post a particular property to his Facebook Page. Along with the property details the user can also provide a message that he can post to FB. The message could be about the property that he would want the people to know about. For posting the Property to FB the spring social framework's Page

Operations object as shown below:

```
facebook.pageOperations().postPhoto(pageId, albumId, resource, caption);
```

Apart from posting the property to FB the other important sub-feature is adding tags to a property. The tags functionality allows the realtor to tag certain attributes to particular property. Some of the tags are safe, beach, skyline, mountain, university, student, upscale and many more. The tags can be added from the dropdowns and also the exisying tags can be edited or removed. How these tags help? Well these tags come handy when the application searches for properties for a particular customer. These tags are keywords that the algorithm matches with the data provided by the Realtor's customer while signing up. It also analyses the description/expectations provided by the customer by matching the text with these tagging keywords.

Social connections:

The social connections feature is used to connect, maintain or disconnect with the social media like Facebook and Twitter. The Facebook and Twitter functionalities do not come up on the menu unless you are logged into Facebook and Twitter respectively. The logging in functionality is pretty much provided by the spring social framework. For logging into Facebook or twitter the consumer-Key and consumer-Secret are required as shown below:

```
ConnectionFactoryRegistry registry = new ConnectionFactoryRegistry();
            registry.addConnectionFactory(new
TwitterConnectionFactory(environment.getProperty("twitter.consumerKey"),
                        environment.getProperty("twitter.consumerSecret")));
            registry.addConnectionFactory(new
FacebookConnectionFactory("349748648445657","ebb75941f20396e69aae76b441fca746"
```

```
));
```

Facebook:

As the name suggests the Facebook feature provides a way to connect with Facebook. The Facebook Page feature provides a way to post Links and Messages to Realtor's Facebook page. Along with the ability to post it also shows up all the previous posts. Along with the posts it shows the number of Likes and Comments done on any particular post. This way the realtor knows which property is most talked about or liked and so he can concentrate on those to strike a deal. Along with that he could also know which some of the unpopular properties are, so he can concentrate on ways to promote those and fine ways to find a right match of customers to that property.

The other Facebook feature is Facebook page Events. This feature allows the realtor create events on Facebook, thus making it easier for him to arrange open houses and other such events. Creating events directly from the tool makes his work easier as he has all the data available at a mouse click.

Twitter:

Twitter is another platform this application tries to explore so that the Realtor can reach a much larger audience without actually annoying them with email and phone calls. The Twitter feature allows you to take a peep into your timeline. But the powerful thing is he can reach his followers by tweeting about the properties from the tool.

It also provides a way to search his or his followers tweet. So if at any later point if the Realtor wants to know about whom all have re-tweeted on the original tweet about a property and get in touch with them.

Analytics:

The analytics feature is yet another important feature. This page shows the realtor about his social media activity in terms of graph. This page provides a deep insight into the realtors FB activity. It shows the number of posts for a particular week, the number of likes and so on.

One of the major challenges we faced with analytics was we did not have enough data so that Facebook could provide insights about our page. So we created a lot of posts, likes with the help of the FB friends. Facebook provides insights for a page only if the numbers of likes are 30 or above. The other important issue faced was finding the REST APIs that could give us some meaningful data as there are bunch of insights that Facebook provides. Thus finding the ones that made sense in the context of the application was a cumbersome task.

The next issue faced with analytics was that the Spring social framework does not support Insights operations. And making your own rest calls directly to the FB API's meant a lot of work and would have ended into complex pieces of code. Accessing Facebook API does require access tokens and some other permission which is readily provided by the framework. But it could not be reused as that functionality was available as jar files. So research on this led us to github[15] where somebody already had faced the issue and had created the insights by writing code above the framework. We built that application and saw that code for insights was packed into same jars. So we took those jars and replaced with ours. Thus we could get the insights but these jar files were referred externally from pom.xml so the generated war file did not work on EC2. Thus we had to manually add these jars into the war and then it work on Amazon EC2 server.

To come up with the graphs we used Fusion charts. It uses flash to display and the data is provided to it either as XML files or through JSON. We have used both the methods. A sample piece of code is shown below:

```
<script type="text/javascript"><!--
      var myChart = new FusionCharts( "<c:url
value="/resources/js/Column3D.swf" />",
      "myChartId", "400", "300", "0", "1" );
        var jsonString = "<c:out value="${graphJsonData}" />";
      var jsonStringModified = jsonString.replace(/&#039;/g,'"');
      myChart.setJSONData (jsonStringModified);
     myChart.render("chartContainer");
 </script>


<div id="chartContainer">FusionCharts XT will load here!</div>
```

The script generates the graph and then gets displayed in the appropriate “Div”. Some analytics are shown below:

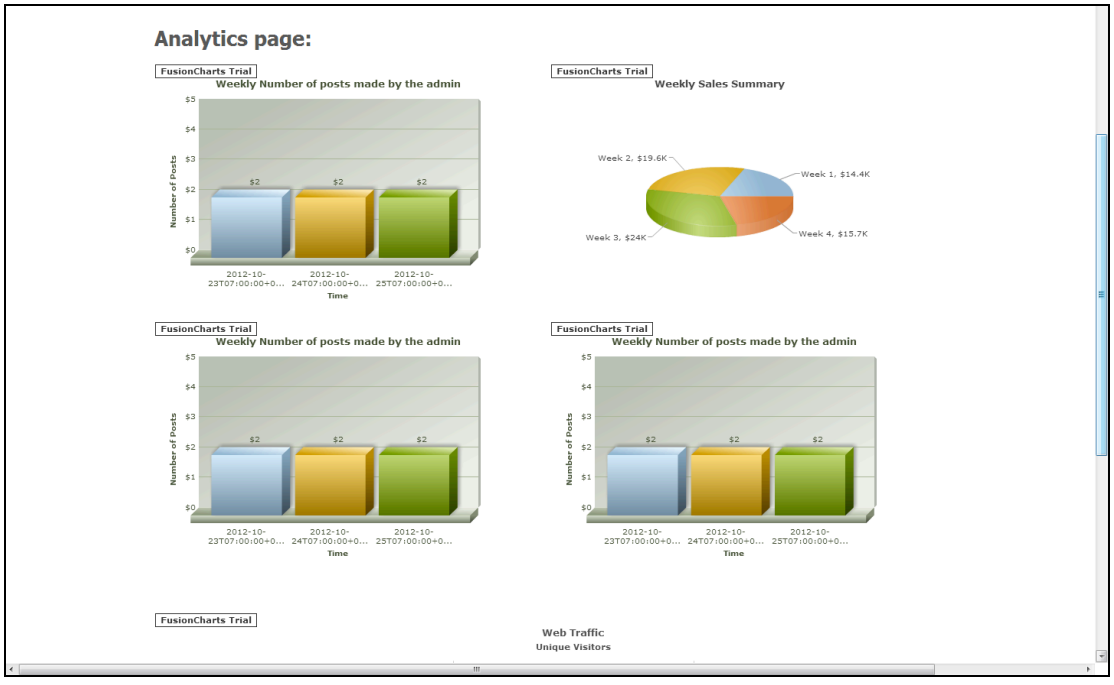


Figure 21: Analytics page 1

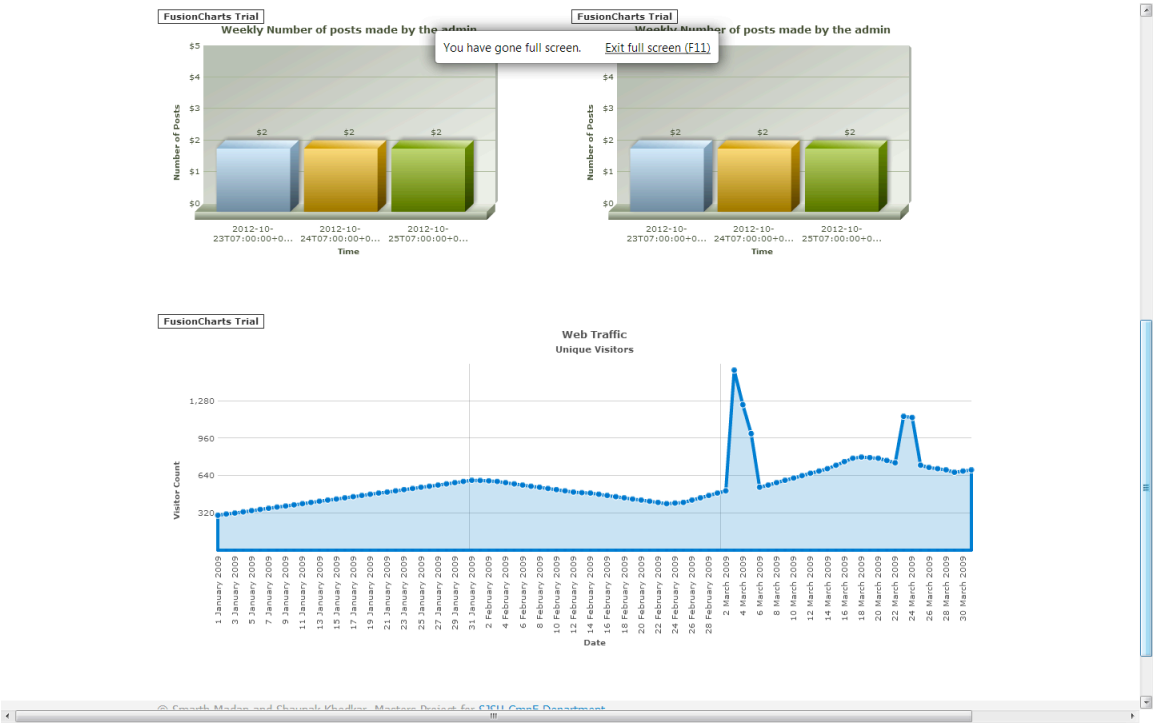


Figure 22 : Analytics page 2

Appointments:

Appointments feature shows up all the request appointments that the realtor has received when users signed up with the realtor from Facebook. Thus he has all the relevant data at his hands. He can either directly call the customer about the meeting or he can email the customer from the tool itself.

News:

The news feature provides latest insights into Real Estate markets. Currently it just uses RSS feeds to pull in data from some other sites. But it gathers data from 2-3 news pages thus providing a consolidated data to the realtor.

Chapter 6. Performance and Benchmarks

The customer acquisition tool is a web-based tool. The tool has been integrated with social media thus exposing it to a large number of audiences. Hence the web-based tool is required to respond to a large a number of requests. Thus performance testing is a very important part of this project. Performance testing is a way to test how the tool behaves under a particular workload. This is the way to examine the scalability and reliability.

Junit Testing:

Each controller class has a Juint test which makes use of Easy Mock framework for unit testing. These unit test are run every time we build the war file for the project. In case of any failures, test result is visible at the time of build and war file creation.

Selenium Web-flow Tests:

Black-box testing for the flow of the tool was tested through Selenium test cases. Selenium IDE 1.9.0 has capability to be used as a plugin with the Firefox browser. The record functionality on the IDE tool is used to record the flow on the browser as the user interacts with it. Along the way there is an option to specify checks such as validate if any element is present or not or verify the text is showing up properly. For our testing we created test cases for all the menu items present in our tool<Picture>. Once we have defined the test cases we can run them after any changes we do in our code to determine if our changes has made the flow to break at any place or not.

To explain one of the flow in our application where the Realtor login into the application. The flow of the login is being shown in figure 24.

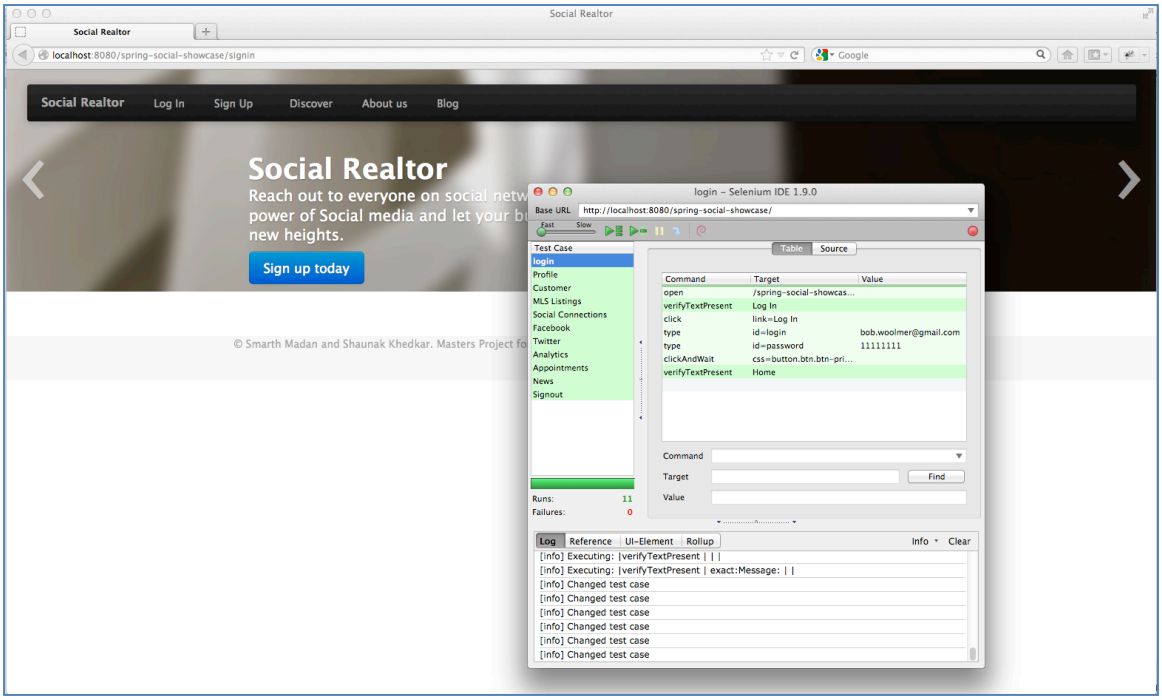


Figure 24 : Selenium test case

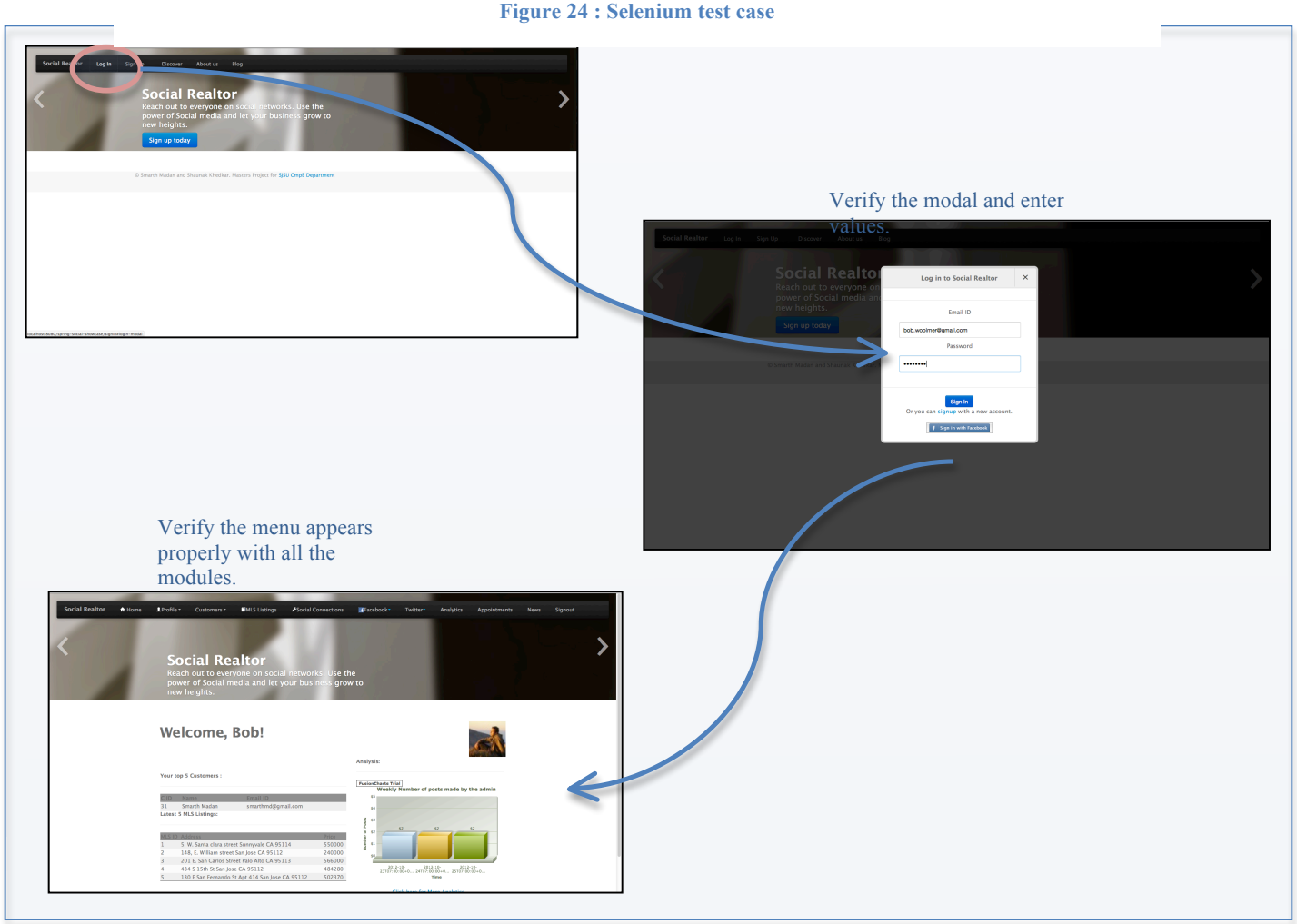


Figure 23 : Selenium test flow

Following performance tests are performed on this project.

1. **Load testing:** Load testing is testing the software under expected load. The load is typically a concurrent number of users executing a certain number of transactions. For this purpose we have used Jmeter.
2. **Stress testing:** Stress testing on the other hand is checking the upper limit of the software under which it performs satisfactorily. Again for this purpose we use Jmeter.

Jmeter:

Jmeter is an open source application used to do performance testing. It can be used to test static as well as dynamic resources like files, servlets, java objects and queries. It can test many servers like http and https , database via JDBC and mail service SMTP server. Its fully multi-threaded environment provides for making concurrent calls. However Jmeter cannot test Javascript used in the HTML pages.

We used the plugin loadosophia, which can be used inside the Jmeter. It creates a pdf report on running the jmeter test cases. When running the normal get request to our application running on amazon ec2 micro instance with 300 users we got the following results:

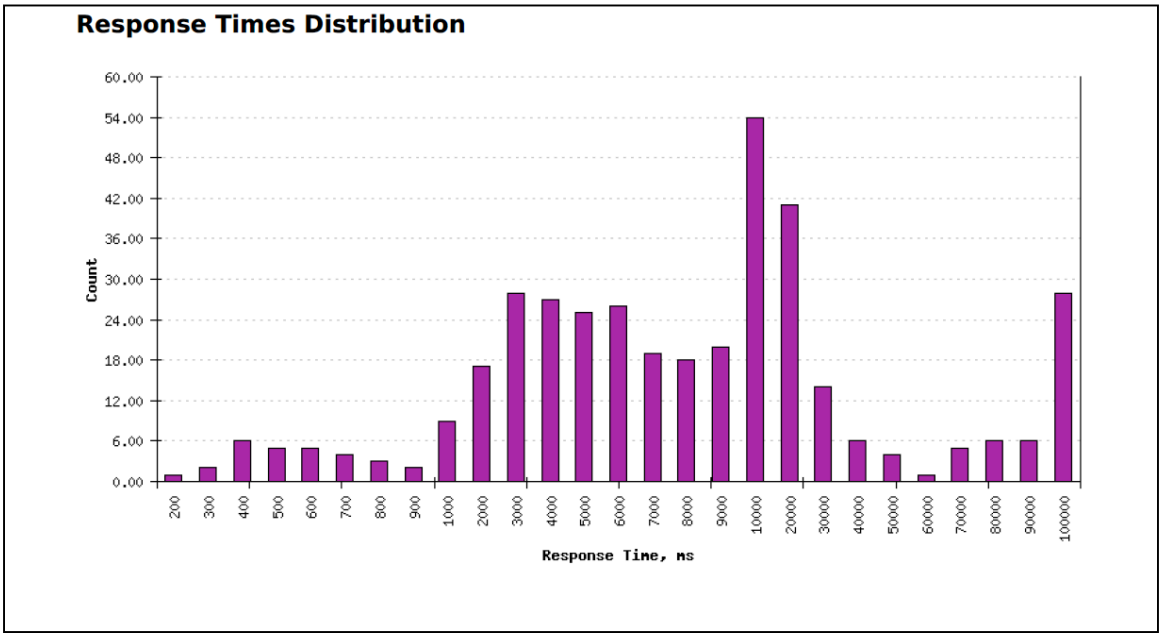


Figure 25: Jmeter Response time distribution

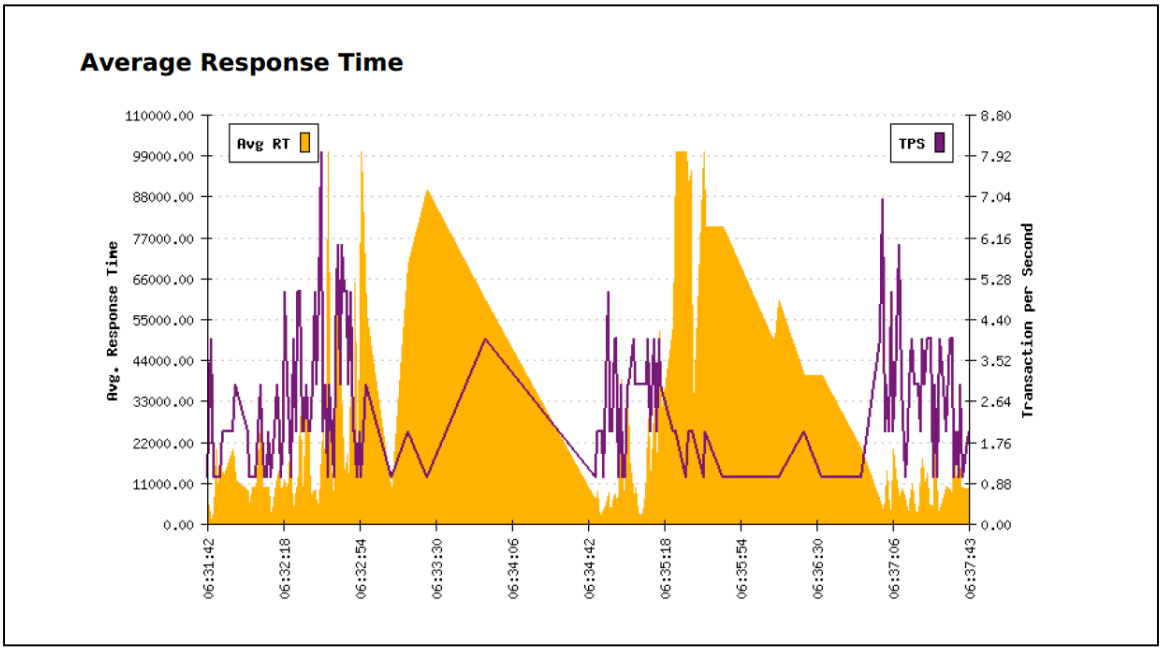


Figure 26: Jmeter - Average response time

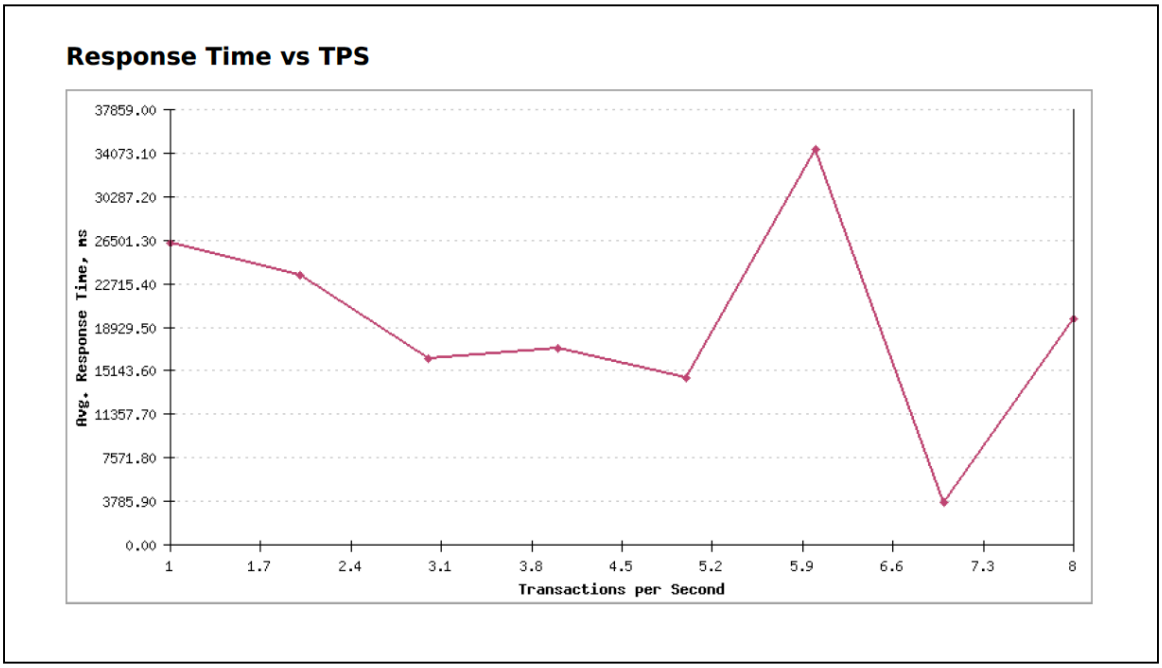


Figure 27: Jmeter: Response Time vs TPS

As we can see from the test results above with a few requests having a long response time, most of the requests have response time below one second. Also the Average response time for the 300 requests that were made is around 19,000 ms.

Chapter 7. Deployment, Operations, Maintenance

Amazon AWS

Our application runs on Amazon cloud web service on an EC2 instance. The micro instance of the Amazon is free of cost for a limited number of hours per month. It is easy, reliable and quick to deploy and run our application on the Amazon. Following are the steps for running the application on Amazon:

1. Make sure you build the project and the war file is ready to be deployed. Eclipse Maven plugin takes care of this.
2. Create an account on Amazon AWS and log into the personalized console to view all the services available to you.
3. Create an instance for EC2(Elastic Cloud Compute) with the choice of Linux, windows or Mac OS and hardware configurations. You would also need to set the security groups allowing some people access to the instance you created based on their roles. Although there is a service called Amazon Beanstalk that can be used to deploy the war file but we chose to deploy the tomcat server ourselves on the

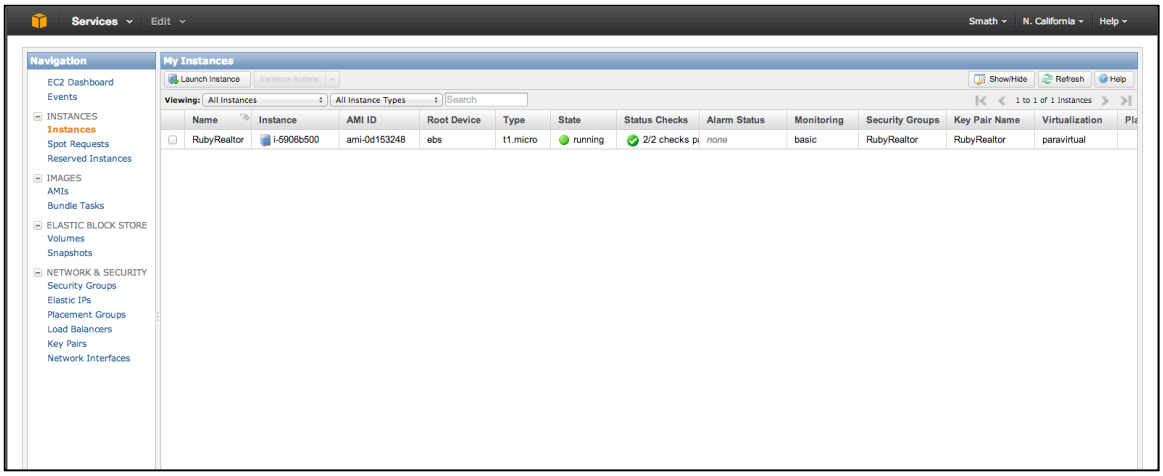


Figure 28: Amazon EC2 instance status

instance and run the application through it. The running instance on EC2 is shown in the figure.

4. SSH into the instance using the key pair provided by Amazon. Once you have logged into the server, install the components that would be used to make the application run:
    a. Tomcat 7
    b. MySQL server
    c. Solr Server
5. Using tomcat7 manager GUI you can deploy the war file as shown in the below screen shots:

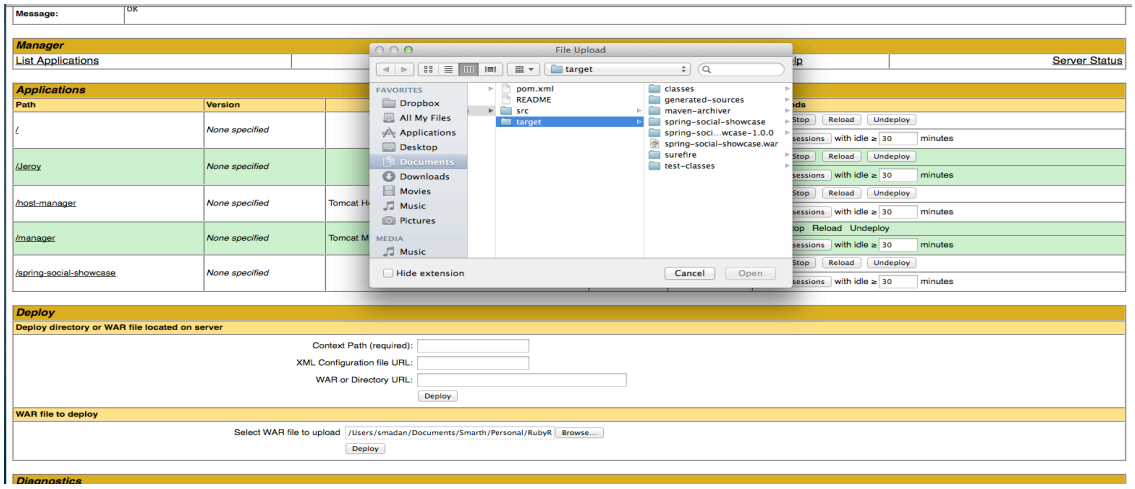


Figure 29: Apache tomcat 7 deploy screen

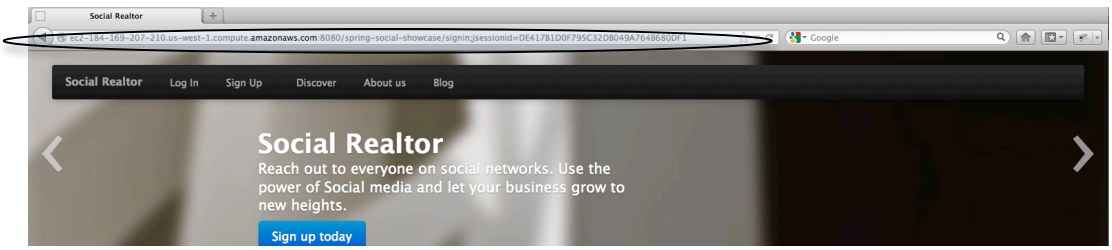


Figure 30: Deployed application

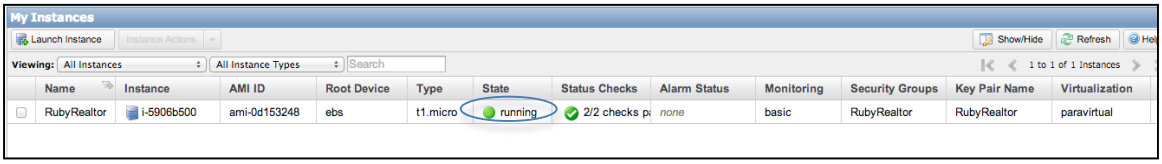
6. Run the MySQL scripts to create the database for the application to run.
7. Import the MLS data into the Solr for Matching algorithm to pick it up.

Operations:

EC2 status

For the operations and normal working of the website, we can monitor a lot of things.

Starting with the EC2 instance that can be monitored from the AWS console:



	Name	Instance	AMI ID	Root Device	Type	State	Status Checks	Alarm Status	Monitoring	Security Groups	Key Pair Name	Virtualization
☐	RubyRealtor	i-5906b500	ami-0d153248	ebs	t1.micro	● running	✓ 2/2 checks pa	*none*	basic	RubyRealtor	RubyRealtor	paravirtual

Figure 31: EC2 running instance

Tomcat Logs

For viewing the logs for tomcat7, we can ssh into the ec2 instance and navigate to folder "/var/log/tomcat7" and see the file catalina.out :

```
ubuntu@ip-10-161-49-21:~$
ubuntu@ip-10-161-49-21:~$ tail -f /var/log/tomcat7/catalina.out
DEBUG: org.springframework.web.servlet.resource.ResourceHttpRequestHandler - Found matching resource: ServletContext resource [/resources/images/slide-03.jpg]
DEBUG: org.springframework.web.servlet.resource.ResourceHttpRequestHandler - Determined media type 'image/jpeg' for ServletContext resource [/resources/images/slide-03.jpg]
DEBUG: org.springframework.web.servlet.DispatcherServlet - Null ModelAndView returned to DispatcherServlet with name 'appServlet': assuming HandlerAdapter completed request handling
DEBUG: org.springframework.web.servlet.DispatcherServlet - Successfully completed request
DEBUG: org.springframework.web.servlet.DispatcherServlet - Null ModelAndView returned to DispatcherServlet with name 'appServlet': assuming HandlerAdapter completed request handling
DEBUG: org.springframework.web.servlet.DispatcherServlet - Successfully completed request
DEBUG: org.springframework.web.servlet.DispatcherServlet - Null ModelAndView returned to DispatcherServlet with name 'appServlet': assuming HandlerAdapter completed request handling
DEBUG: org.springframework.web.servlet.DispatcherServlet - Successfully completed request
DEBUG: org.springframework.web.servlet.DispatcherServlet - Null ModelAndView returned to DispatcherServlet with name 'appServlet': assuming HandlerAdapter completed request handling
DEBUG: org.springframework.web.servlet.DispatcherServlet - Successfully completed request
```

Figure 32: Tomcat7 logs

MySQL Workbench

For managing the MySQL, you can easily do so using MySQL Workbench. It's a free tool and can be used to create and manage the databases with easy to use GUI:

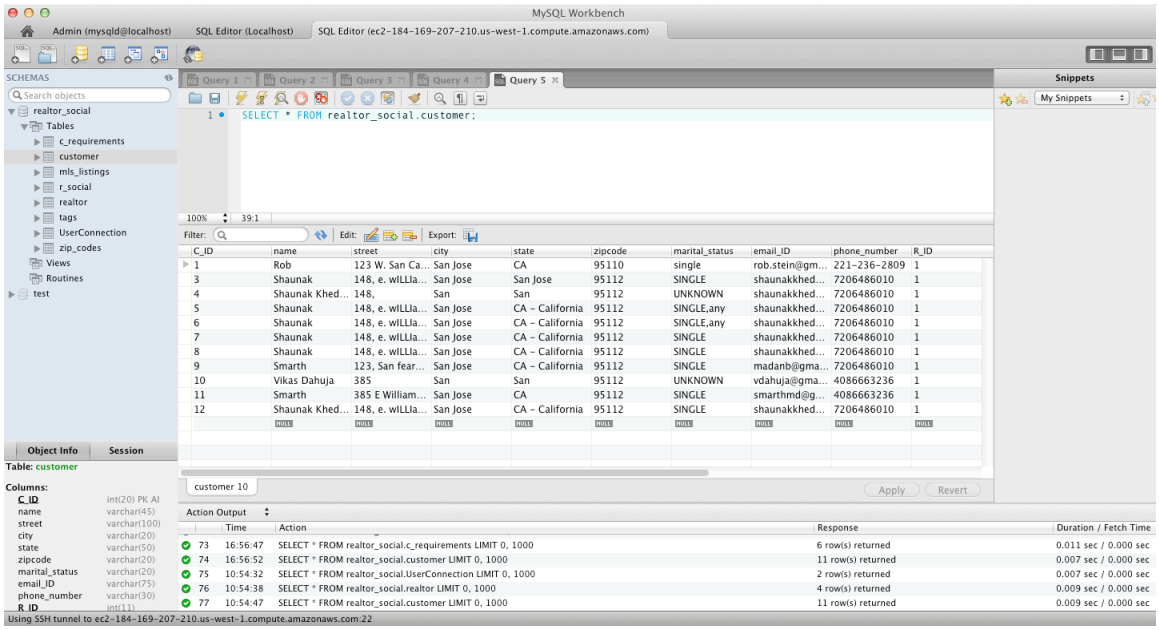


Figure 33: Workbench query result

Single Database Approach

We decided to go with single database for the complete application instead of different database for each realtor that log in. We found quite a few advantages of this. Firstly any Software update or bug fixes became very easy as it had to be done only once. Updating any data also became easier which overall increased the maintainability of the database. Reporting different data for a particular realtor became fast. But this approach has its cons to. Data integrity is a big issue we need to face. Data of one realtor cannot t be shown to the other realtor. Updating data should not have table level lock, which could

jeopardize multi-tenant architecture. We have done extensive testing to make sure these issue do not arrive.

SAAS architecture

Our application that is deployed on Amazon cloud can be treated as a SaaS based architecture as the Realtor are using our software as a service through the internet. We as the service provider are using Amazon's Infrastructure such as server machines, load balancer and all the hardware through IaaS(Infrastructure as a service) model. This model of ours support **multiple Realtors.** As shown in the architecture diagram below for Amazon Web Service, our website is hosted on Amazon's EC2 instances which is automatically scalable by Amazon depending on the load. We pay Amazon by an hourly rate for the infrastructure they provide us.

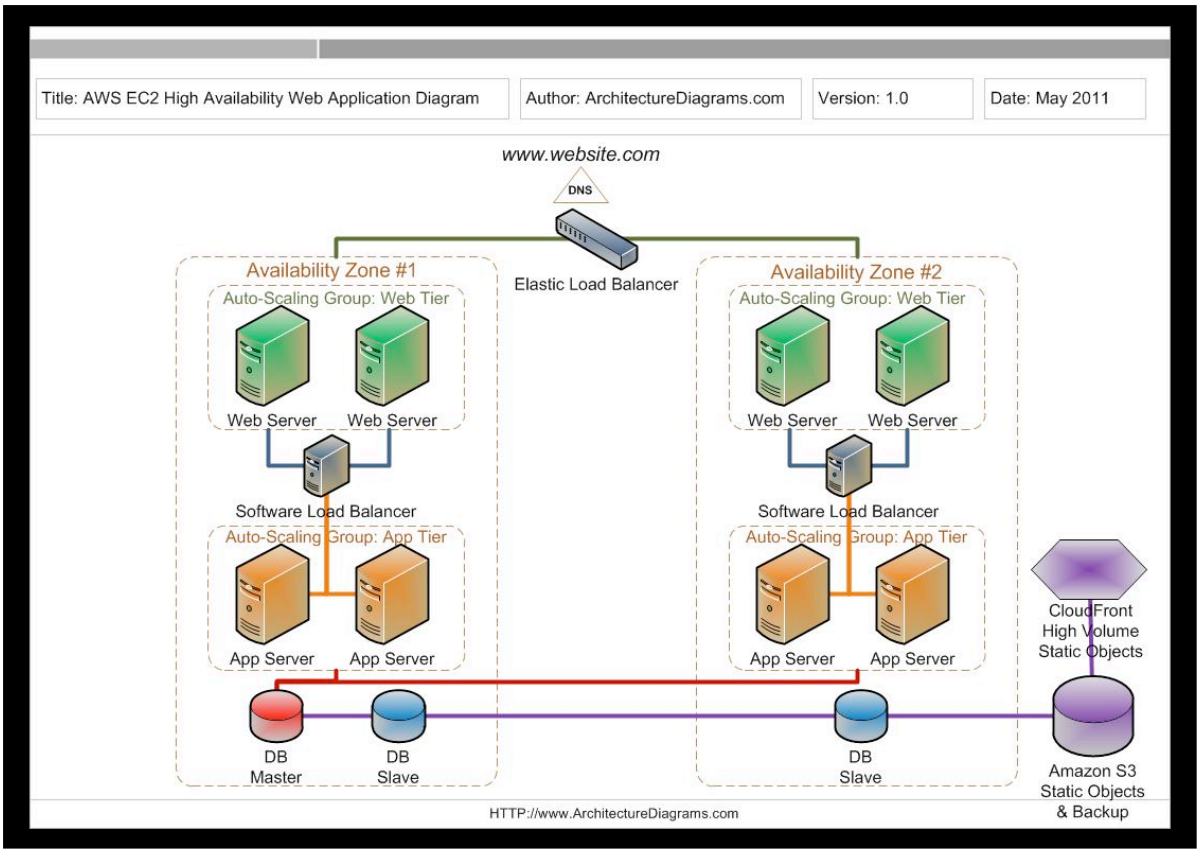


Figure 34: Amazon EC2 Architecture[14]

Github

All our code was shared between our teammates using a centralized repository in github. It is easy to use and good GUI for seeing all the commits and comparing two versions of a file. Following is a screenshot of how the commit log looks like on github :

Also following is a screenshot on how the file comparison becomes easy on github:

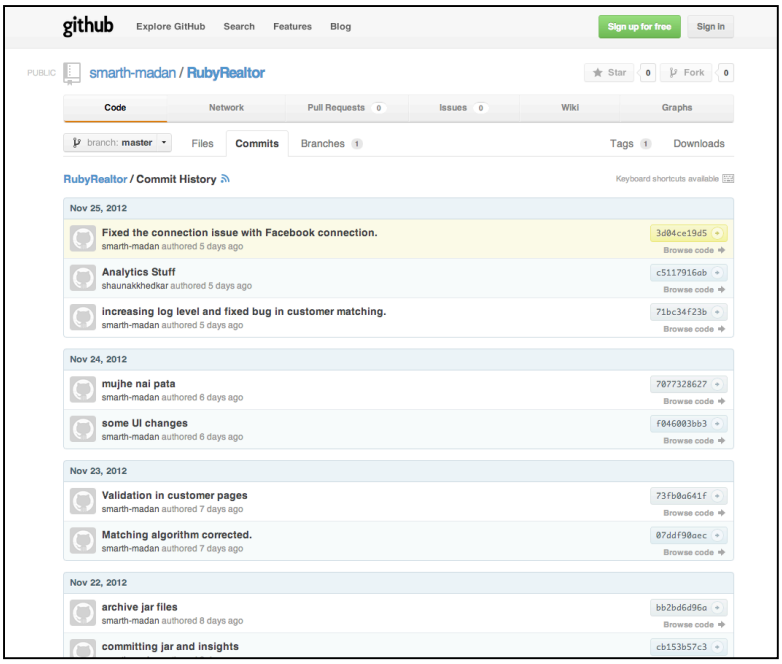


Figure 35: Github commit log

Figure 36: Github file comparison

Chapter 8. Summary, Conclusions, and Recommendations

Summary

Our project here is focused on developing solution for realtor for all his social attention he needs to get noticed along with smart engine to do searching in MLS listings for his customers. The tool will also manage few of the workflow processes such as conducting open house and getting market trends, which will empower the realtor to rise above through his competitors.

Conclusions

This application has been a huge learning opportunity for us about the happenings in Real estate market as well as emerging social trends. The application has huge market demand and at the end of this project we are quite confident that this tool could be a potential start up idea. Not many tools are available in the market which focus on providing solution that we plan to provide.

Recommendations for Further work

Although the project is fully working but there is always scope for future enhancements. Following are a list of future enhancements that could be possible to make the tool more effective:

1. **Predictive Analysis**: Some kind of data mining could help us dig into more details about the customer's requirements. Such as if the customer recently became a Father, he could be looking for a bigger house now.
2. **Semantics**: Currently when a user types in some description of his house, they are matched with some particular keywords. Semantics will help us identify more keywords with similar meaning and a more accurate property search result.

3. **MLS integration**: Since we do not have the license real estate agent, we were not having subscription to MLS data. One of the nice to have feature would be to integrate the tool with live MLS data and real time analysis.
4. **Social Media analysis**: Analysis of comments posted by other on Facebook could be implemented. This would give more insights about positive or negative comments people are giving for the entire page or a particular post.

References

[1] Digital & Social Media Statistics – February 2012. (2012, February 25). Retrieved April 1, 2012, from http://laurenproctor32.com: http://laurenproctor32.com/blog/digitalsocial-media-statistics-february-2012/

[2] Top 10 Social Networking Websites of 2012. (2012, Feburary 14). Retrieved April 2, 2012, from roymorejon.com: http://roymorejon.com/top-10-social-networking-websitesof-2012/

[3] An estimate of active Twitter accounts per ultimo February 2012. (n.d.). Retrieved April 2, 2012, from http://twopcharts.com/twitter500million.php

[4] Ebersman, D. A. (2012, February 1). Retrieved April 4, 2012, from http://www.sec.gov: http://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm

[5] Graph API. (n.d.). Retrieved April 2012, from https://developers.facebook.com: https://developers.facebook.com/docs/reference/api/

[6] Market Snapshot. (n.d.). Retrieved April 2012, from http://www.topproducer.com/products/market-snapshot.aspx

[7] REST API Resources. (n.d.). Retrieved April 2012, from https://dev.twitter.com: https://dev.twitter.com/docs/api

[8] Shoutlet. (n.d.). Retrieved 2012 April, from http://shoutlet.com/

[9] Social Bios. (n.d.). Retrieved April 2012, from http://socialbios.com/

[10] Spring MVC. (n.d.). Retrieved April 5, 2012, from Vaannila: http://www.vaannila.com/spring/spring-mvc-tutorial-1.html

[11] Web Real Estate Marketing. (n.d.). Retrieved from http://www.webrealestatetools.com/

[12] JavaServer Pages(n.d). Retrieved November 2012 from http://en.wikipedia.org/wiki/JavaServer_Pages

[13] MVC for Dummies(n.d) Retrieved November 2012 from http://www.endlick.com/mvc-for-dummies/

[14] Amazon AWS EC2 high availability Web Application Diagram(n. d) Retrieved November 2012 from http://architecturediagrams.com/tag/amazon-ec2

[15] spring-social-facebook Retrieved November 2012 from https://github.com/jachzen

[16] LoadosophiaUploader (n.d) Retrieved November 2012 from http://code.google.com/p/jmeter-plugins/wiki/LoadosophiaUploader

[17] Apache Solr(n.d) Retrieved November 2012 from http://lucene.apache.org/solr/

Appendices

Appendix A. Description of CDROM Contents

The CD contains data from our Github repository in the following structure:

a) Docs – This folder contains all the documents for our project. These includes project report, project proposal, DB creation scripts, Jmeter test cases, Selenium test cases and Sprint planning document.

b) Ruby-Realtor – This folder contains the Java Source code for our project. The same code could be imported on eclipse for future enhancements.

c) WAR – This folder consist of war file which can be deployed on the tomcat 7 server.

d) Solr – This folder consist the executable for the SOLR server. Any user of the tool has to go to the example directory and execute following command to start the server:

   "Java- jar start.jar"

   To import data from the MySQL server, the user has to run the following command on the browser :

   "ec2-184-169-207-210.us-west-1.compute.amazonaws.com:8983/solr/dataimport?command=full-import"