Project Report

on

Real Estate Management

a project in J2EE

To be submitted in partial fulfilment for the award of degree of

Bachelor of Technology

(Computer Science & Engineering)

at

AMITY UNIVERSITY

Submitted To:

Ms. Shilpi Barman Sharma

Submitted By:

Anand Yati

A2305207057

Roll No. # 456, 7CS1

B.Tech (CS&E) 2007-11

AMITY SCHOOL OF ENGINEERING & TECHNOLOGY

AUUP, Noida

ACKNOWLEDGEMENT

It is my pleasure to acknowledge in front of you the summer internship report on the project "**Real Estate Project**". First and foremost I would like to express my profound sense of gratitude to **Mohd. Imran** Trainer/Developer Hewlett Packard for providing crucial support and able guidance which was critical in development of this project, and without which I would not have been able to complete the project.

I would like to thank my faculty mentor **Ms. Shilpi Barman Sharma** for useful suggestions, feedback and help. I express my sincere gratitude to her for her able criticism and encouragement. I would also like to express my gratitude to **Mr. Kaiser Singh** sir H.O.D CS&E, for providing me crucial help at different stages and for his able guidance.

Last but not least I would like to thank my team members and my parents for their constant support, help and encouragement.

ANAND YATI

ABSTRACT

Our main objective was to develop a professional property selling website in Java. We were given a set of minimum requirements and features which our website needs to have and which form the core of essential functionalities we want to deliver from our website.

The **functionalities** which were required by our website and which are indeed delivered by us are as follows:

1. We have developed our website in **STRUTS framework.**
2. Our website implements user's **security via session tracking.**
3. Clients are provided with the option of **uploading their requirements** in advanced and quick search modules of our website.
4. Non-registered users can sign up and **create new accounts** and registered users can **modify their existing account**.
5. An **admin account** is **used to validate** any new accounts created by user.
6. Clients are able to **upload their company's image** which is **displayed** to them **in** their **account information** section.
7. **Appearance** of website is **professional** and **industrial nomenclature** is strictly followed in coding.
8. **Database Access Object Layer Architecture** is used in database connectivity.
9. There are three types of users of our website: Normal user, Agent, Builder. All of these users have different functionalities attached with them which will be discussed later in this report.
10. Agent and builder can upload additional details about their company and their work area.
11. Html coding and number of html pages are kept to minimum.

Apart from fulfilling the minimum the criteria and requirements we have added additional features too in our website which enhance its usability, security and overall functionality. These **additional features** are as follows:

1. We have used **Struts Tiles framework** for developing user interface of our website.
2. We have used **Struts Validator framework** to filter and check user input before entering it into our database.
3. We have provided space to put **ads** on our website.

4. We have **disabled page traversing via url rewriting** which enhances our website's security further.
5. We have used **JSP standard tag library components** to display messages at certain locations.
6. Even non-registered users can access **quick search** option of our website which provides generic search results in comparison to **advanced search** which provide specific result.
7. All of our web-pages are in Struts.

Blending of such rich technologies and framework has dramatically and drastically boosted up the re-usability, understanding and functionality of our website. Our website has been repeatedly checked by various users including esteemed trainers & developers of industry. This has resulted in fixing of several issues and updating of our website, in order to create this final bug free version.

We have tested our site on various internet browsers like IE 8.0 beta, Mozilla Firefox, Google Chrome to check its compatibility and view, and we have found it operating efficiently over all these browsers. We have also done a comprehensive testing on data being entered into our website and have eliminated any of the possible errors which may occur or which occurred.

Our website serves all basic needs of any person using it to make his property deals. We have covered all requirements and have not only implemented them but also have implemented them in best and most efficient possible way. We have always kept in mind the industry nomenclature and guidelines during complete tenure of development of this website. It was really a privilege for us to develop a website for world's largest IT company Hewlett Packard and under the guidance of industry's best experts and developers.

So we present in front of u **www.squarefeet.com** our own property dealing website to cater all your needs related to property dealing. We have studied & gone through a number of available property dealing websites to understand the functionalities they offer. Analysis of existing work in our field gave us novel ideas and created a deep understanding of the product and its minimum required quality level. We have benchmarked our websites with several leading websites in the same field in terms of security, ease of use, functionality and appearance. We have created a simple yet a very powerful product which is ready to be used by industry.

The motto of our development team was not only to develop this product but it was to create a milestone and a benchmark in this field. We faced a lot of challenges as there were already so many websites in this field and we wanted to be different and better, but we succeeded to create a masterpiece of work with our innovative ideas, hard work and excellent guidance from our mentors.

TABLE OF CONTENTS

INTRODUCTION

1.1.Introduction

We were assigned the task to build a professional website for property selling, renting and purchasing. We named our website as www.SquareFeet.com. The name of our website is derived from its utility. We chose this name as it would be easier for the clients to remember the basic unit Sq. Ft. in which area of any property is measured. When we say "building a professional website" then it means a lot more things than its mere appearance. It involves the functionality, methodology and framework of development combined with a professional look. The features which were required in our website as per company requirements/guidelines are:

1. Website should be in STRUTS Framework.
2. It should provide security to persons using it via session tracking or cookies.
3. Non-registered users can sign up and create new accounts and registered users can modify their existing account.
4. An admin account is used to validate any new accounts created by user.
5. Clients should be able to upload their requirements and property details onto the website.
6. Clients should be able to search properties as per their requirements.
7. Clients should be able to upload their company's image which is displayed to them in their account information section.
8. Appearance of website should be professional and industrial nomenclature should be followed in coding.
9. Database Access Object Layer Architecture should be used in database connectivity.
10. There should be three types of users of our website: Normal user, Agent, Builder. All of these users have different functionalities attached with them which will be discussed later in this report.
11. Agent and builder should upload additional details about their company and their work area.
12. Html coding and number of html pages should be kept to minimum.

As per company guidelines these were the minimum features which we were bound to complete in order to complete our project. We have also provided space for ads which can be used by client to post ads of his properties onto our site. Further in this report I will discuss about the various JAVA Technologies we used in developing our website. I will answer to all possible questions like: What, Why & How. I will discuss about what we made, how we made and why we made it in this way. I will be discussing about the significance and functionality of every component we used.

Further I will be discussing about the lifecycle model and designing approach followed by us while working on this project. DFD and use-case diagrams of our modules and project as a whole are attached with this report. In most of the cases the software developed by students are not up to the industry standards but we are proud to develop a website which is built on the latest norms followed by industry in building websites similar to us. We don't believe in merely completing our work. Our motto is to create a benchmark in the work assigned to us.

Our website provides a very user friendly environment and can be used easily by the clients for their use. Simplicity and powerful functionality are two attributes on which remained our primary objectives throughout the development process. We have developed a product which can easily be modified to accomplish new needs and to suit our desires. The coding is done with an idea in mind that anyone who wants to add or update the functionality can easily understand our present work. The industry nomenclature followed is a big step in boosting the readability of code.

We have accomplished not only the minimum requirements specified for our project but also have added additional functionalities where it was needed. We have given our best efforts to make the code error free and our code has been verified by industry's leading trainers.

ABOUT COMPANY

2.1.About Company

HP is a technology company that operates in more than 170 countries around the world. They explore how technology and services can help people and companies address their problems and challenges, and realize their possibilities, aspirations and dreams. HP applies new thinking and ideas to create more simple, valuable and trusted experiences with technology, continuously improving the way our customers live and work.

No other company offers as complete a technology product portfolio as HP. They provide infrastructure and business offerings that span from handheld devices to some of the world's most powerful supercomputer installations. HP offers consumers a wide range of products and services from digital photography to digital entertainment and from computing to home printing. This comprehensive portfolio helps us match the right products, services and solutions to our customers' specific needs.

2.2.Fast Facts about HP:

- HP was founded in 1939.
- Corporate headquarters are in Palo Alto, Calif.
- Mark Hurd is president and CEO.
- HP is the world's largest IT company, with revenue totaling $114.6 billion for fiscal 2009.
- HP's 2009 Fortune 500 ranking: No. 9.
- HP serves more than one billion customers in more than 170 countries on six continents.
- HP has approximately 304,000 employees worldwide.
- HP ship **48 million** PC units annually.
- **One** out of every **three** servers shipped worldwide is from HP.
- HP Software makes calls possible for more than **300 million** mobile phone customers around the globe.
- HP helps **50 million** customers store and share over 4 billion photos online.
- HP supports the top **200 banks** and more than 130 of the world's major stock exchanges.

2.3.Contribution

- HP strives to be an economic, intellectual and social asset to each country and community in which we do business.
- Key areas of contribution are electronic waste, raising standards in our global supply chain and increasing access to information technology.

2.4.Technology Leadership

HP's three business groups drive industry leadership in core technology areas:

- The Personal Systems Group: business and consumer PCs, mobile computing devices and workstations.
- The Imaging and Printing Group: inkjet, LaserJet and commercial printing, printing supplies.
- Enterprise Business: business products including storage and servers, enterprise services and software.

2.5.Growth

HP is focused on three technology shifts that have the power to transform our customers' lives and businesses:

- Information explosion
- Digital transformation
- Environmental sustainability

MATERIALS AND METHODS USED

3.1.Materials and Methods Used

We have used a number of a number of software resources, JAVA technologies and frameworks which we are mentioned here. Our code is fully forward compatible with all the new versions of the software we have used.

3.2.Java Technologies Used :

1. Struts Website Designing Framework.
2. Struts Tiles framework.
3. Struts Validator framework.
4. JSP standard tag library components.
5. Database Access Object Layer Architecture.
6. Other J2EE Tools :
   - ❖ Servlets
   - ❖ JSP
   - ❖ Type 4(pure java) Database Drivers.

3.3.Software Specifications:

1. JDK 1.5
2. NetBeans IDE 6.5
3. MySQL Server 5.0 & MySQL GUI Tools 5.0
4. Macromedia Dreamweaver 8
5. Adobe PhotoShop CS4
6. Web Browsers(Internet Explorer 8, Mozilla Firefox 3.6, Google Chrome 5.0)
7. Server Used : Apache Tomcat 6.0.18

3.4.Minimum System Requirements for developing this project :

1. 800 MHz Intel Pentium III processor (or equivalent) or later.
2. Windows 2000 or Windows XP or later.
3. 512 Mb RAM.
4. 1024x768 16 bit display(32 bit recommended).
5. 1.5 GB Free hard disk space.

3.5.Method

We divided our project into three generic phases which was further divided into modules to make our work systematic, organized and efficient. The two main phases of our website development were:

1. Design Phase
2. Coding Phase
3. Testing Phase

Each of these phase are dependent on each other and are followed in the same order i.e. design phase follows the coding phase and so does the testing phase. This is so because only after the designing phase we come to know that what are the components we need to code for and only when complete coding is done, we get a working product to test.

The Designing phase is further divided into these modules or phases:

1. Selection of a CSS (Cascading Style Sheet) for our website.
2. Creating logo, basic appearance and set a name for our Website.
3. Creating simple HTML pages.
4. Converting these HTML pages into Struts.
5. Creating database for the inputs taken from user and to store data on which our website works.

The Coding phase is further divided into these modules or phases:

1. Creating form beans for HTML pages where input is taken from user.
2. Creating action servlets for implementing the logic on the inputs taken from user.
3. Creating a DAO (Data Access Object) Layer Architecture for database connectivity.
4. Breaking the CSS into components like header and footer so that they can be used as tiles. Implementing Struts Tiles Framework.
5. Applying these tiles to our web-pages.
6. Implementing the searching algorithm.
7. Creating dynamic web-pages.
8. Creating a Admin account.
9. Attaching Struts Validator Framework to the designed web-pages.
10. Disabling the page traversing via URL rewriting.

Testing Phase:

1. Alpha Testing: Done by our team after complete coding.
2. Beta Testing: Done by industry developers and trainers.

Now I am going to explain each and every phase of our project in details so as to throw light on underlying details. This will create a deep understanding of methodology followed by us in development of this website.

Designing Phase:

1. Selection of CSS: We needed to select a appropriate CSS for our website. A CSS or cascading style sheet provides a basic layout of a website and also with the basic components of the website. In our case we needed a CSS which demonstrated the features required for our website. We browed through all the possible CSS available on net related to property dealing and finally short-listed a few of them. Then with the advice of our team members and our industry guide we selected a specific CSS and modified and tailored it to fit our specific needs. The CSS provided us with following features :
    - Background.
    - Basic set of buttons.
    - Basic layout of website.
    - Space for ads.

2. Creating a logo, basic appearance and set a name for our website: A logo is what we call trademark of our business. It is the thing which gets imprinted in the minds of customers and clients using a website. It showcases the category and business standards of our website and hence our business. Hence it was a challenge for us to select a good and suiting logo for our website. Which showcases all the requirements mentioned above, and hence it took us really a long time to arrive at our final logo which we used in our website.

   Setting the basic appearance for our website was yet another phase which was taken out with great care. As this phase decided the overall appearance, positioning of controls and buttons over web-pages and many more things. The appearance of any website is the first thing which catches the attention of its users. We made sure to use soothing colours and positioning the user controls in appropriate locations. We kept in mind the ease of use from the point of view of our user and hence made every effort to simplify the structure and appearance of our website.

We took proper care of alignment of various controls over the web-pages, so that our website can leave a pleasant view in the mind of its viewer. We know and understand this fact very well that appearance of a website is also as important as delivering the required functionality. It is so because what will be the use of an efficient website which no one uses because of its unpleasant or complex user interface.

Finally one of the biggest challenges of this module was deciding the name for our website. We kept in mind following requirements for choosing the name for our website:

- The name should sound as a brand.
- It should be related to the category for which we are designing our website.
- It should reflect our professionalism.
- It should be easy to remember.
- It should be simple but attractive.

So we selected the name www.squarefeet.com for our website. It solved all of our requirement issues and fitted our needs.

3. Creating simple HTML pages : We created all our static HTML pages in Macromedia Dreamweaver 8. It simplified our designing issues to a great extent but still we had to write a lot of code in several pages. We did not merged our CSS here with our html pages. In-fact we just created the controls i.e. buttons, textboxes, checkboxes, combo-boxes etc. and later merged it with our CSS in coding phase. We kept in mind the industry nomenclature while designing these pages and named every HTML control with a noun.

4. Converting these HTML pages into STRUTS: As we needed to use STRUTS framework so the task of using it starts with the elementary phase of our designing. We had to change all of our html pages into STRUTS HTML pages. Although this was a tedious task for us as we can't afford to miss even a single tag or write it in a wrong way, but we managed to clear through it with ease. We used the tag library: "/WEB-INF/struts-html.tld" with a prefix name "html" which we attached to all the relevant places like:
   - Code declaring the HTML controls like buttons, text-fields etc.
   - HTML tags.

This procedure converted all our simple HTML pages to STRUTS HTML pages. The aim of this conversion was:

- Our basic requirement of using and applying STRUTS FRAMEWORK to our website.
- As we wanted to use STRUTS VALIDATION FRAMEWORK and STRUTS TILES FRAMEWORK.

5. Creation of database: After designing the web-pages for our website we came to know the various inputs we are taking from user. Depending upon these inputs we needed a storage mechanism to store the data passed by the user which can later be retrieved easily. We used MY SQL 5.0 to create and maintain our database. We created various tables and fields in them to store the data supplied by our user. We tried and also succeeded in creating minimum number of tables in our database.

Coding Phase:

1. Creating form beans for STRUTS HTML pages: We created form beans for fetching the data from the view part of struts architecture. These form beans are simple java pages which are stored with a .java extension. View part here is the HTML page on which user is entering data. These form beans contains the setter and getter methods for all the input fields filled by the user. These setter and getter fields gets the data from the HTML form and pass it on to ACTION SERVLET to perform required logical operations.

2. Creating Action Servlets: Each and every form bean is related with an action servlet which contains the logic to be implemented on the data fetched by form bean. This action servlet applies the business logic and forwards the control of program to appropriate servlet for further processing(if necessary). Action servlet and form beans are two faces of the same coin and are essential counterparts. These action servlets contain all the java code for storing, retrieving and forwarding control to appropriate JSP page. These servlets also contain the code for implementation of our business logic.

3. Creating a DAO Layer Architecture : It consist of three components :
   - Connection class: To create connection with database.
   - Interface: To declare a method which executes query passed by the developer.

- Java class: Which implements the above interface and defines the methods in it, this class is called and the query we need to pass is passed as the argument in its method.

DAO layer architecture provides an easy way of passing queries to database and reduces the redundant code i.e. which we needed to write everytime we made a connection to database on any of the JSP or Servlets. Apart from this DAO layer architecture also fulfils the industry standards of designing any project. It simplifies and optimize the code written to make and use database connectivity.

4. Implementing Struts Tiles Framework: We broke our CSS into small components like header, footer etc. and then inserted them into various jsps. These jsps were then called with our normal STRUTS HTML pages to combine and give a complete view. We used tiles in place of manual CSS because in latter case we need to attach the CSS coding with each of our html page but with the help of tiles we just need to call the code of CSS at appropriate place and our job is done. Hence saving a large amount redundant code generated.

   Tiles enables developers to develop web applications by assembling reusable tiles. The steps to create tiles are as follows :

   - Add Tiles Tag Library Descriptor(TLD) file to web.xml
   - Create layout JSPs.
   - Develop web pages using layouts.
   - Repackage, run and test application.

   We also need to edit tiles-defs.xml file in order to customize and configure tiles to fulfil our requirements and needs.

5. Applying these tiles to our web-pages: We attached our tiles to the JSP pages containing our html codes. These tiles were attached by calling the JSPs which contained these tiles at appropriate locations. The tiles attached with JSPs reduces our burden of re-writing same set of codes like that of CSS and other background options which are common in our all web-pages. We simply need to call the JSP containing the code instead of writing the whole code again.

6. Implementing the searching algorithm: We have used two types of searches in our project that are :

   - Quick Search: Which provides a generic result to user as it compares very less fields with the fields present in our database.

- Advanced Search: It extensively compares a large number of fields and selects the records with maximum number of matching fields and displays it to the user. This search results in a very specific and targeted result.

7. Creating dynamic web-pages: After completing the designing of all the static web pages and attaching with them their respective form beans and action servlets we created the dynamic web pages. These web pages holds the result which is generated according to user's query. These web-pages display the results to user and are often linked with activities like retrieval of a specific data set from database. In our project we have created these dynamic web-pages in JSP.

8. Creating a Admin account: The admin account was a feature which was told to us as a very important feature and also as a specification we needed to complete in order to increase the security of our website. An admin account is an account which has supreme authority over all the other normal accounts on that website.

   In our case we have given following privileges to our admin :

   - Any new signed up person is first verified by the admin to activate that person's account. Without activation that person cannot login into her account and hence can't use the features of our website.
   - Admin can delete any newly signed up person's account.
   - Admin has a secure login via same login page that is for users, but the difference is that no one can access admin account details and can neither change it. This account is directly created into the database with a unique key which directs admin to the page containing his specific controls.

9. Implementing Struts Validator Framework: Validator framework provides the functionality to validate the form data. We used the Validator framework to validate the SIGN UP details of any user as these details are crucial for identification of any user. Hence we applied our validation rules on various fields in which user has to fill his data. This data is checked with our validation rules for its validity as soon as the submit button is pressed. The form data is entered into the database only if it satisfies all the validation rules or else an appropriate error message is displayed to user.

10. Disabling page traversing via URL rewriting : We have observed in case of many websites that once you are logged in to the site then you can open any of the pages of the website just by rewriting in the URL of that website. This poses a very severe security risks because in this way any logged in person may get access to secure data or account information of other persons. Hence we saved

the user data in session object and allowed him to view the pages having session id of his own. We ensured that every session object stores only the data of the user logged in and we cannot change the data in this object .Which means we cannot change our session in between(without relogin) and hence cannot visit the page of some other user(as that user has different data in session object.)

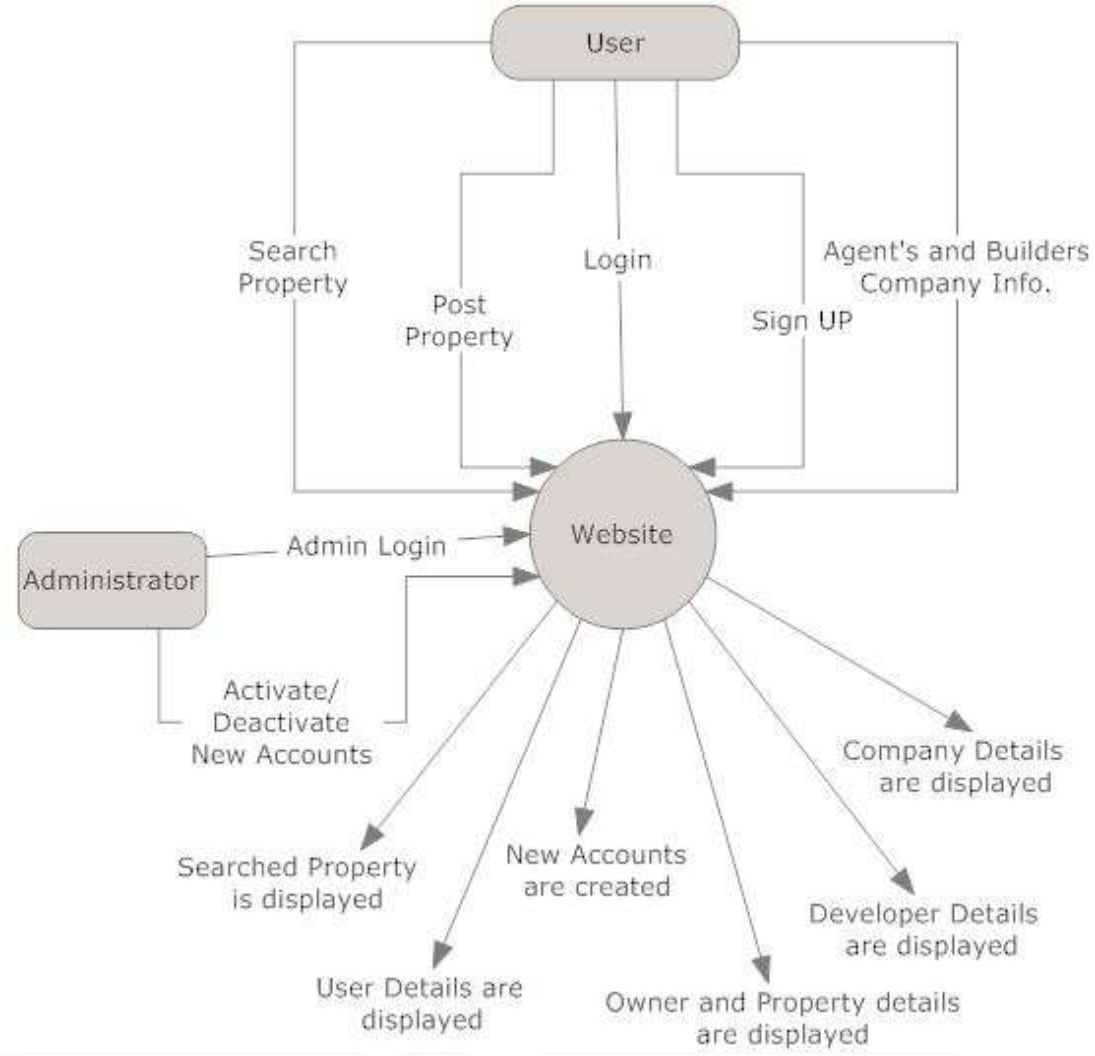
Testing Phase:

1. Alpha Testing: Alpha testing is done at the developers end in order to find an rectify any errors or abnormal behaviour of code in certain situations. In our case being the developer we did the alpha testing for our code i.e. project. We thoroughly tested each and every functionality of our website and entered various test data to ensure the correct working and integrity of our site. We rectified all the errors if found in our testing phase and hence improved the stability of our website.

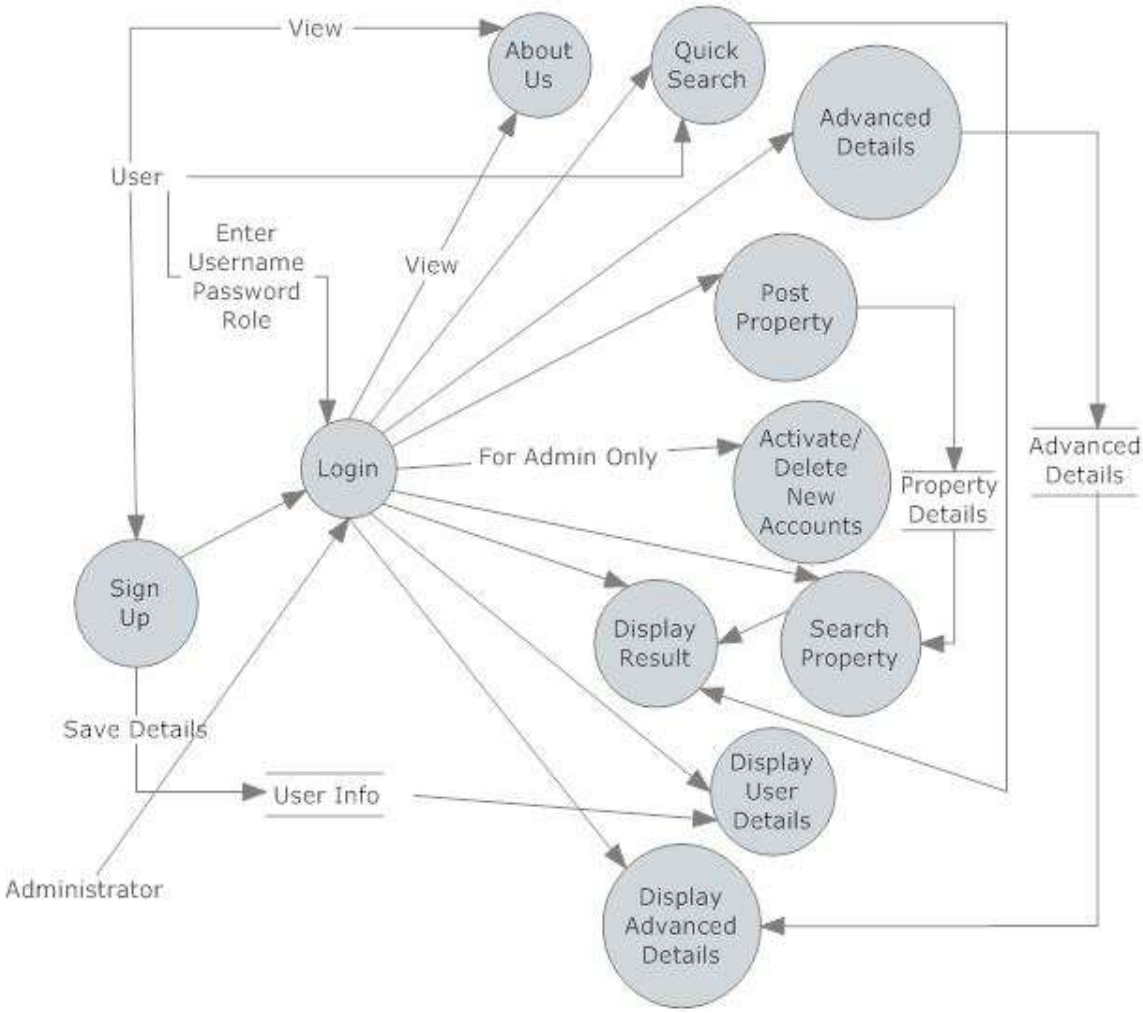
2. Beta Testing: Beta testing is done at the client's end which in our case was trainers and industry developers working for Hewlett Packard. They tested our code thoroughly and checked for all the functionalities applied. They also tested our code for redundancy and ensured that our code satisfies all minimum specifications in best and most efficient way. This testing made our code even more perfect after optimizing on the points pointed out by our industry guides.

3.6. Data Flow Diagram

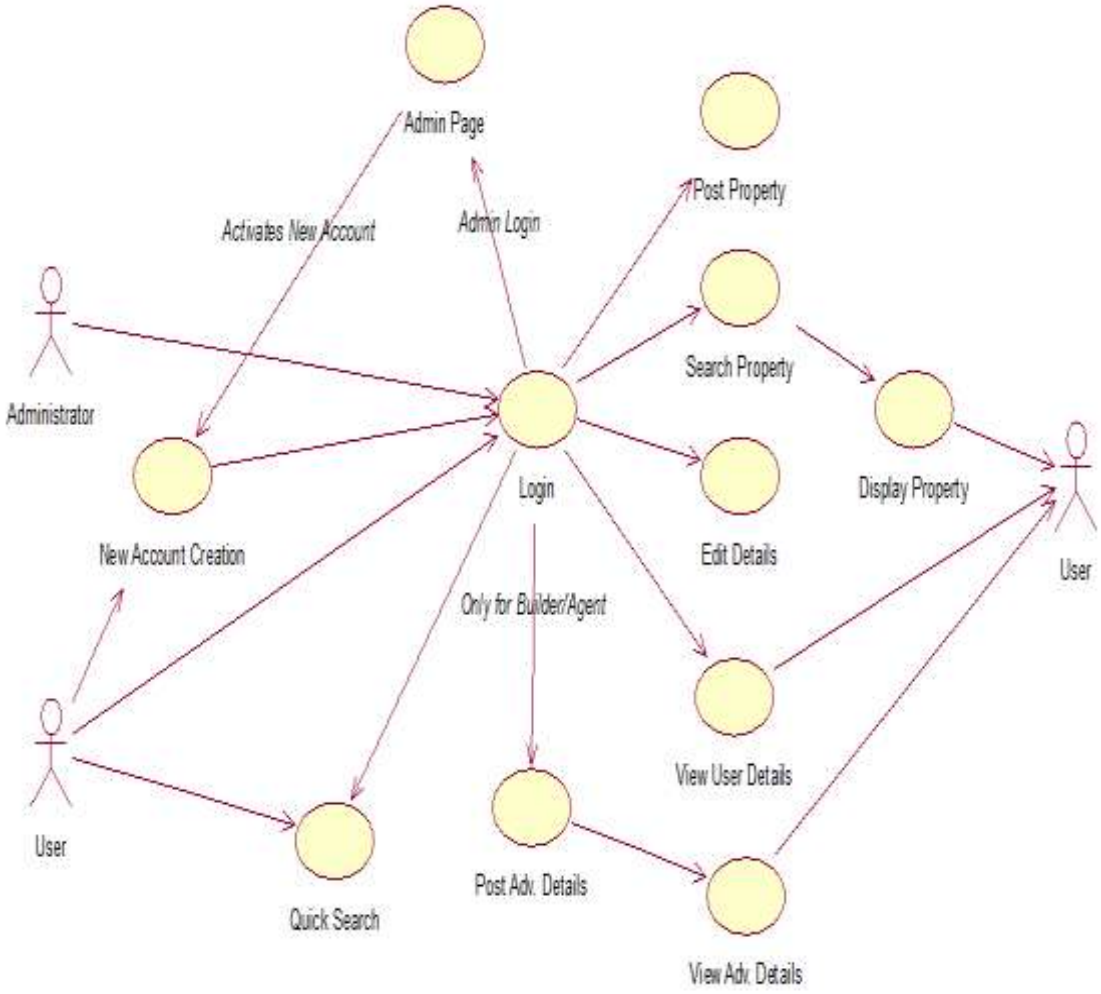
3.6.1. Context Level DFD (Level 0 DFD)



3.6.2. Level 1 DFD



3.7. Use Case Diagram



PROJECT AND TECHNOLOGY

4.1.Project and Technology

In this section I am going to discuss about the problem central to our project, define and discuss about various technologies and platforms used for the development of this project. The problem statement provided to us by the company was: "Develop a website on property dealing which is professional, simple and rich in functionality."

The technological specifications which needed to be fulfilled by us are as follows:

1. Follow the STRUTS 2 Architecture.
2. Use NetBeans IDE 6.5
3. Use Apache Tomcat Server 6.0.18
4. Use MY SQL 5.0 database.
5. Follow Data Access Object Layer Architecture for database connectivity.

The coding specifications which needed to be fulfilled by us are as follows:

1. Clients are provided with the option of **uploading their requirements** in advanced and quick search modules of our website.
2. Non-registered users can sign up and **create new accounts** and registered users can **modify their existing account**.
3. An **admin account** is **used to validate** any new accounts created by user.
4. Clients are able to **upload their company's image** which is **displayed** to them **in** their **account information** section.
5. There are three types of users of our website: Normal user, Agent, Builder. All of these users have different functionalities attached with them which will be discussed later in this report.
6. Agent and builder can upload additional details about their company and their work area.

4.2.Technologies Used :

Struts: When Java servlets were first invented, many programmers quickly realized that they were a Good Thing. They were faster and more powerful that standard CGI, portable, and infinitely extensible.

But writing HTML to send to the browser in endless `println()` statements was tiresome and problematic. The answer to that was JavaServer Pages, which turned Servlet writing inside-out. Now developers could easily mix HTML with Java code, and have all the advantages of servlets. The sky was the limit!

Java web applications quickly became "JSP-centric". This in-and-of itself was not a Bad Thing, but it did little to resolve flow control issues and other problems endemic to web applications.

Another model was clearly needed.

Many clever developers realized that JavaServer Pages AND servlets could be used **together** to deploy web applications. The servlets could help with the control-flow, and the JSPs could focus on the nasty business of writing HTML. In due course, using JSPs and servlets together became known as Model 2(meaning, presumably, that using JSPs alone was Model 1).

Of course, there is nothing new under the Sun ... and many have been quick to point out that JSP's Model 2 follows the classic Model-View-Controllerdesign pattern abstracted from the venerable Smalltalk MVC framework. Java Web developers now tend to use the terms Model 2 and MVC interchangeably. In this guide, we use the MVC paradigm to describe the Struts architecture, which might be best termed a Model 2/MVC design.

The Struts project was launched in May 2000 by Craig R. McClanahan to provide a standard MVC framework to the Java community. In July 2001, Struts 1.0 was released, and IOHO, Java Model 2 development will never be quite the same.

The Model View Controller(MVC) Design Pattern

In the MVC design pattern, application flow is mediated by a central Controller. The Controller delegates requests - in our case, HTTP requests - to an appropriate handler. The handlers are tied to a Model, and each handler acts as an adapter between the request and the Model. The Model represents, or encapsulates, an application's business logic or state. Control is usually then forwarded back through the Controller to the appropriate View. The forwarding can be determined by consulting a set of mappings, usually loaded from a database or configuration file. This provides a loose coupling between the View and Model, which can make applications significantly easier to create and maintain.

The Model: System State and Business Logic JavaBeans

The *Model* portion of an MVC-based system can be often be divided into two major subsystems -- the **internal state** of the system and the **actions** that can be taken to change that state

Many applications represent the internal state of the system as a set of one or more JavaBeans. The bean properties represent the details of the system' state. Depending on your application's complexity, these beans may be self contained (and know how to persist their own state), or they may be facades that know how to retrieve the system's state from another component. This component may be a database, a search engine, an Entity Enterprise JavaBean, a LDAP server, or something else entirely.

Large-scale applications will often represent the set of possible business operations as methods that can be called on the bean or beans maintaining the state information. For example, you might have a shopping cart bean, stored in session scope for each current user, with properties that represent the current set of items that the user has decided to purchase. This bean might also have a `checkOut()` method that authorizes the user's credit card and sends the order to the warehouse to be picked and shipped.

Other systems will represent the available operations separately, perhaps as Session Enterprise JavaBeans (Session EJBs).

In a smaller scale application, on the other hand, the available operations might be embedded within the `Action` classes that are part of the Struts control layer. This can be useful when the logic is very simple or where reuse of the business logic in other environments is not contemplated.

The Struts framework architecture is flexible enough to support most any approach to accessing the Model, but it is **strongly** recommend that you separate the business logic ("how it's done") from the role that `Action` classes play ("what to do").

The View: JSP Pages and Presentation Components

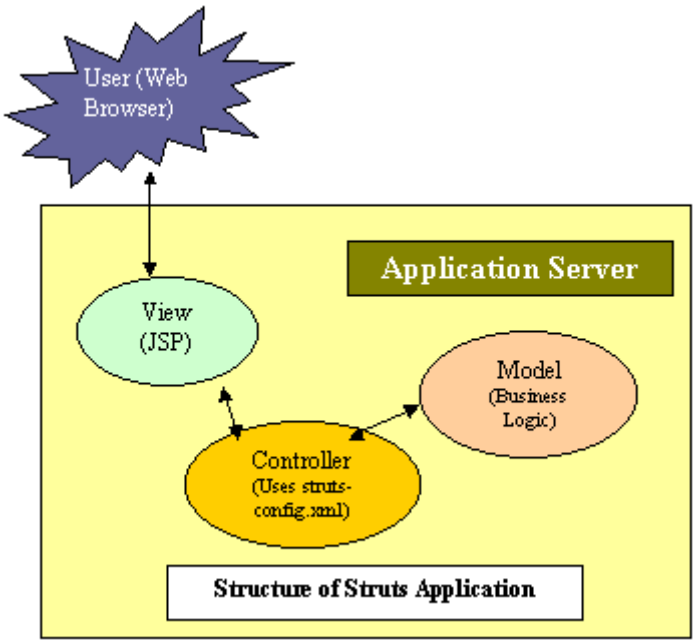
The *View* portion of a Struts-based application is most often constructed using JavaServer Pages (JSP) technology. JSP pages can contain static HTML (or XML) text called "template text", plus the ability to insert dynamic content based on the interpretation (at page request time) of special action tags. The JSP environment includes a set of standard action tags, such as `<jsp:useBean>` whose purpose is described in the JavaServer Pages Specification. In addition to the built-in actions, there is a standard facility to define your own tags, which are organized into "custom tag libraries."

Struts includes a set of custom tag libraries that facilitate creating user interfaces that are fully internationalized and interact gracefully with `ActionForm` beans. ActionForms capture and validate whatever input is required by the application.

The Controller: ActionServlet and ActionMapping

The *Controller* portion of the application is focused on receiving requests from the client (typically a user running a web browser), deciding what business logic function is to be performed, and then delegating responsibility for producing the next phase of the user interface to an appropriate View component. In Struts, the primary component of the Controller is a servlet of class `ActionServlet`. This servlet is configured by defining a set of `ActionMappings`. An ActionMapping defines a `path` that is matched against the request URI of the incoming request and usually specifies the fully qualified class name of an Action class. All Actions are subclassed from `org.apache.struts.action.Action`. Actions encapsulate calls to business logic classes, interpret the outcome, and ultimately dispatch control to the appropriate View component to create the response.

Struts also supports the ability to use `ActionMapping` classes that have additional properties beyond the standard ones required to operate the framework. This allows us to store additional information specific to your application and still utilize the remaining features of the framework. In addition, Struts lets us define logical "names" to which control should be forwarded so that an action method can ask for the "Main Menu" page (for example), without knowing the location of the corresponding JSP page. These features greatly assist us in separating the control logic (what to do) with the view logic (how it's rendered).



Struts Validator Framework

Struts Framework provides the functionality to validate the form data. It can be use to validate the data on the users browser as well as on the server side. Struts Framework emits the java scripts and it can be used to validate the form data on the client browser. Server side validation of the form can be accomplished by sub classing your From Bean with **DynaValidatorForm** class.

The Validator framework was developed by David Winterfeldt as third-party add-on to Struts. Now the Validator framework is a part of Jakarta Commons project and it can be used with or without Struts. The Validator framework comes integrated with the Struts Framework and can be used without doing any extra settings.

Using Validator Framework

Validator uses the XML file to pick up the validation rules to be applied to an form. In XML validation requirements are defined applied to a form. In case we need special validation rules not provided by the validator framework, we can plug in our own custom validations into Validator.

The Validator Framework uses two XML configuration files **validator-rules.xml** and **validation.xml**. The **validator-rules.xml** defines the standard validation routines, these are reusable and used in**validation.xml**. to define the form specific validations. The **validation.xml** defines the validations applied to a form bean.

Structure of validator-rule.xml

The **validation-rules.xml** is provided with the Validator Framework and it declares and assigns the logical names to the validation routines. It also contains the client-side javascript code for each validation routine. The validation routines are java methods plugged into the system to perform specific validations.

Following table contains the details of the elements in this file:

Element	Attributes and Description
form-validation	This is the root node. It contains nested elements for all of the other configuration settings.
Global	The validator details specified within this, are global and are accessed by all forms.
Validator	The validator element defines what validators objects can be used with the fields referenced by the formset elements. The attributes are: • **name**: Contains a logical name for the validation routine • **classname**: Name of the Form Bean class that extends the subclass of ActionForm class • **method**: Name of the method of the Form Bean class • **methodParams**: parameters passed to the method • **msg**:Validator uses Struts' Resource Bundle mechanism for externalizing error messages. Instead of having hard-coded error messages in the framework, Validator allows you to specify a key to a message in the ApplicationResources.properties file that should be returned if a validation fails. Each validation routine in the validator-rules.xml file specifies an error message key as value for this attribute. • **depends**: If validation is required, the value here is specified as 'required' for this attribute. • **jsFunctionName**: Name of the javascript function is specified here.
Javascript	Contains the code of the javascript function used for client-side validation. Starting in Struts 1.2.0 the default javascript definitions have been consolidated to commons-validator. The default can be overridden by supplying a <javascript> element with a CDATA section, just as in struts 1.1.

The Validator plug-in (validator-rules.xml) is supplied with a predefined set of commonly used validation rules such as Required, Minimum Length, Maximum length, Date Validation, Email Addressvalidation and more. This basic set of rules can also be extended with custom validators if required.

Structure of validation.xml

This **validation.xml** configuration file defines which validation routines that is used to validate Form Beans. You can define validation logic for any number of Form Beans in this configuration file. Inside that definition, you specify the validations you want to apply to the Form Bean's fields. The definitions in this file use the logical names of Form Beans from the struts-config.xml file along with the logical names of validation routines from the validator-rules.xml file to tie the two together.

Element	**Attributes and Description**
form-validation	This is the root node. It contains nested elements for all of the other configuration settings
Global	The constant details are specified in <constant> element within this element.
Constant	Constant properties are specified within this element for pattern matching.
constant-name	Name of the constant property is specified here
constant-value	Value of the constant property is specified here.
Formset	This element contains multiple <form> elements
Form	This element contains the form details. The attributes are: **name**:Contains the form name. Validator uses this logical name to map the validations to a Form Bean defined in the struts-config.xml file
Field	This element is inside the form element, and it defines the validations to apply to specified Form Bean fields. The attributes are: • **property**: Contains the name of a field in the specified Form Bean • **depends**: Specifies the logical names of validation routines from the validator-rules.xml file that should be applied to the field.
Arg	A key for the error message to be thrown incase the validation fails, is specified here
Var	Contains the variable names and their values as nested elements within this element.

var-name	The name of the criteria against which a field is validated is specified here as a variable
var-value	The value of the field is specified here

Struts Tiles

Struts Tiles component. Tiles is a templating system. (Include on steroids.) It can be used to create a common look and feel for a web application. Tiles can also be used to create reusable view components. Tiles builds on the "include" feature provided by the Java Server Pages specification to provide a full-featured, robust framework for assembling presentation pages from component parts. Each part ("Tile") can be reused as often as needed throughout your application. This reduces the amount of markup that needs to be maintained and makes it easier to change the look and feel of a website.

Overview of Tiles Features

- Screen definitions
  - Create a screen by assembling ***Tiles*** , e.g. header, footer, menu, body
  - Definitions can take place:
    - in a centralized XML file
    - directly in JSP pages
    - in Struts Actions
  - Definitions provide an inheritance mechanism: a definition can extend another one and override some (or all) of its parameters
- Layouts
  - Define common page layouts and reuse them across your web site
  - Define menu layouts and pass lists of items and links
  - Define a portal layout, use it by passing list of ***Tiles*** (pages) to show
  - Reuse existing layouts, or define your own ones
- Dynamic page building
  - Tiles are gathered dynamically during page reload. It is possible to change any attributes: layout, list of Tiles in portal, list of menu items, ...
- Reuse of ***Tiles*** /Components
  - If well defined, a ***Tile*** can be reused in different locations
  - Dynamic attributes are used to parameterize ***Tiles***
  - It is possible to define libraries of reusable ***Tiles*** .
  - Build a page by assembling predefined components, give them appropriate parameters
- Internationalization (I18N)
  - It is possible to load different Tiles according to the user's Locale
  - A mechanism similar to Java properties files is used for definition files: you can have one definition file per Locale, the appropriate definition is loaded according to the current Locale
- Multi-channels

- It is possible to load different Tiles according to a key stored e.g. in session context
- The key could hold e.g. user privileges, browser type, ...
- A mechanism similar to Java properties files is used for definition files: you can have one definition file per key, the appropriate definition is loaded according to the key

Apache Tomcat

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process.

Apache Tomcat is developed in an open and participatory environment and released under the Apache License version 2. Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. **Apache Tomcat** (or **Jakarta Tomcat** or simply Tomcat) is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

Data Access Object (DAO) layer

The Data Access Object (DAO) layer is an essential part of good software architecture. A DAO is an interface to some kind of database. It provides some specific operations without revealing or exposing the details of the database.

In the web and software development process, this object design pattern is applicable to most programming languages, software and almost all types of databases. But, Data Access Objects are always associated with JAVA EE applications.

Web & software development companies develop certain business applications that always need to access to data from an object or relational databases. The JAVA platform is the best platform that offers many techniques or methods for accessing the data. A web and software development firm uses the oldest and the most mature technique: the JDBC API or Java Database Connectivity API. The JDBC API provides the developer an access to the data.

Advantages of Data Access Objects:

The data Access Object design pattern provides a simple and rigorous technique. It separates object persistence and data access logic from an application which do not know anything about each other and which can be assumed to evolve independently.

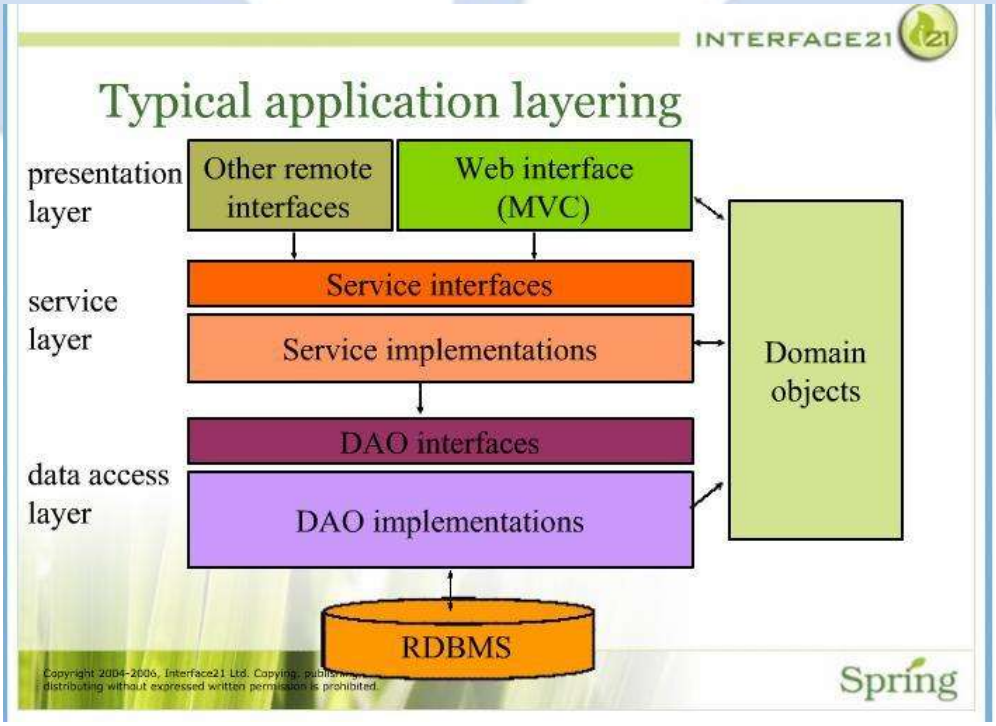
Data Access Objects provides ample flexibility to alter an application's database or persistence mechanism over time without the need to re-program the logic of the application that interacts with the Data Access Object Tier.

The Data Access Object design pattern can be executed in a number of ways. This ranges from an outright simple interface that separates data access from application logic to commercial products and frameworks.

Data Access Object coding model (or paradigms) require some excellent programming skills and web development in India hire the best experts and use traditional as well as modern technologies to accomplish it. An ideal web development India uses technology like Java persistence technologies to ensure the design pattern is implemented. It also employs other technologies such as EJB CMP that come inbuilt into an application server. The technology can be used in a JEE application server. Applying the Data Access Object (DAO) pattern throughout the applications enabled us to separate low-level data access logic from business logic. We built DAO classes that provide CRUD (create, read, update, delete) operations for each data source.

The DAO pattern is one of the standard J2EE design patterns. Developers use this pattern to separate low-level data access operations from high-level business logic. A typical DAO implementation has the following components:

1. A DAO factory class
2. A DAO interface
3. A concrete class that implements the DAO interface
4. Data transfer objects (sometimes called value objects)



NetBeans IDE 6.5

The NetBeans IDE is an award-winning Integrated Development Environment available for Windows, Mac, Linux, and Solaris. The NetBeans project consists of an open-source IDEand an application platform which enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy, and C/C++. It is supported by a vibrant developer community and offers a diverse selection of third-party plugins

Supported Technologies:

NetBeans IDE 6.5 supports the following technologies and has been tested with the following application servers. If you plan to use versions other than those explicitly listed, please note that you may encounter unpredictable IDE behavior as a result of the technologies being external interfaces which the project team has little or no control over.

- Java EE 5 and J2EE 1.4
- JavaFX 1.1
- Struts 1.2.9
- Spring 2.5
- Hibernate 3.2.5
- Java API for RESTful Web Services (JAX-RS) 1.0
- Java API for XML-based RPC (JAX-RPC) 1.6
- PHP 5.2
- Ruby 1.8
- JRuby 1.1.4
- Rails 2.1
- Groovy 1.5
- Grails 1.0
- VCS
- CVS: 1.11.x, 1.12.x
- Subversion: 1.3.x, 1.4.x, 1.5.x
- Mercurial: 1.0.x
- ClearCase V7.0

Tested application servers:

- Sun Java System Application Server 9.0 (GlassFish V1)
- Sun Java System Application Server 9.1 (GlassFish V2)
- Sun GlassFish Enterprise Server v3 Prelude
- Sun Java System Application Server PE 8.2
- Tomcat 5.5
- Tomcat 6.0.18
- JBoss 4.0.4
- WebSphere 6.0
- Websphere 6.1
- WebLogic 9.2
- WebLogic 10

System Requirements

Minimum Hardware Configurations:

- **Microsoft Windows XP Professional SP3:**
  - **Processor:** 800MHz Intel Pentium III or equivalent
  - **Memory:** 512 MB
  - **Disk space:** 750 MB of free disk space

- **Microsoft Windows Vista:**
  - **Processor:** 800MHz Intel Pentium III or equivalent
  - **Memory:** 512 MB
  - **Disk space:** 750 MB of free disk space

Recommended Hardware Configurations:

- **Microsoft Windows XP Professional SP3:**
  - **Processor:** 2.6 GHz Intel Pentium IV or equivalent
  - **Memory:** 1 GB
  - **Disk space:** 1 GB of free disk space

- **Microsoft Windows Vista:**
  - **Processor:** 2.6 GHz Intel Pentium IV or equivalent
  - **Memory:** 1 GB
  - **Disk space:** 1 GB of free disk space

RESULT AND DISCUSSIONS

5.1.Result and Discussions

We have successfully developed a professional property dealing website which can cater all the needs of its clients related to their business in property dealing. We have used the latest and most efficient java technologies in developing this website. These technologies range from STRUTS 2 Framework, DAO layer Architecture, STRUTS Validator Framework, STRUTS Tiles Framework and recent versions of MY SQL 5.0, Macromedia Dreamweaver etc.

We have highly focussed ourselves during the complete development phase to follow the industry guidelines for the coding and nomenclature of our modules and components. All of our web-pages are in struts and we have taken every effort to minimise the redundant code. We have also minimised the number of HTML pages so as to optimize the size of our website on server. The heavier (in terms of size) the website is the more time it takes to load on to the server.

When we compare our work with that of existing professional websites or from the work done by other workers(teams) in the same company we find dramatic difference(positive) in the functionality we have delivered.

Following are the shortcomings in the work of other workers (teams) in general; working under the supervision of same company which we have overcome:

1. No one other than our team implemented the STRUTS Tiles Framework. This led us to reduce the redundant code and increased the reusability of our code.
2. Most of the teams developing this website did not implement the STRUTS Validator Framework or implemented it in a partial manner. Whereas we implemented it on various fields necessary and with the correct combination of logic.
3. None of the teams did not provided user with a option of Quick Search which a client can access without logging, which is implemented by our team.
4. Most of the teams did not implemented the ADMIN Account which is used to activate or delete new users which is implemented efficiently in our website.
5. Very few of the have teams used the JSTL components which we have used in appropriate places.

Following are the results which we have obtained when we compared our website with other sites available in the market:

1. Our website is benchmarked against one of the most leading websites developed by market leader IT companies.
2. We have delivered all and even more (in some cases) functionalities then professionally deployed websites of same category on internet.
3. We have used quality standards and programming techniques which are equivalent to that used in industry.

CONCLUSION AND RECOMMENDATION

6.1.Conclusion and Recommendation

We have successfully developed a professional property dealing website which can cater all the needs of its clients related to their business in property dealing.

The **functionalities** which were required by our website and which are indeed delivered by us are as follows:

1. We have developed our website in **STRUTS framework.**
2. Our website implements user's **security via session tracking.**
3. Clients are provided with the option of **uploading their requirements** in advanced and quick search modules of our website.
4. Non-registered users can sign up and **create new accounts** and registered users can **modify their existing account**.
5. An **admin account** is **used to validate** any new accounts created by user.
6. Clients are able to **upload their company's image** which is **displayed** to them **in** their **account information** section.
7. **Appearance** of website is **professional** and **industrial nomenclature** is strictly followed in coding.
8. **Database Access Object Layer Architecture** is used in database connectivity.
9. There are three types of users of our website: Normal user, Agent, Builder. All of these users have different functionalities attached with them which will be discussed later in this report.
10. Agent and builder can upload additional details about their company and their work area.
11. Html coding and number of html pages are kept to minimum.

Apart from fulfilling the minimum the criteria and requirements we have added additional features too in our website which enhance its usability, security and overall functionality. These **additional features** are as follows:

1. We have used **Struts Tiles framework** for developing user interface of our website.
2. We have used **Struts Validator framework** to filter and check user input before entering it into our database.
3. We have provided space to put **ads** on our website.
4. We have **disabled page traversing via url rewriting** which enhances our website's security further.
5. We have used **JSP standard tag library components** to display messages at certain locations.

6. Even non-registered users can access **quick search** option of our website which provides generic search results in comparison to **advanced search** which provide specific result.
7. All of our web-pages are in Struts.

Although we have accomplished all the specification related requirements but we would like to recommend that our website can further be enhanced and improved in terms of improved user interface, technology used and search algorithm used.

6.2.Implication for Future Work

Our website offers a wide range of possible updates to be carried out on it to further enhance its functionality. We have created our website keeping in mind reusability and continued optimization of its code. Evolution of The areas where we would like to improve in future are as follows:

- We would like to implement our website in Spring Framework.
- We would like to improve on the search algorithm.
- We would like to improve the user interface.

SNAPSHOTS

7.1.Snapshots of Working Website

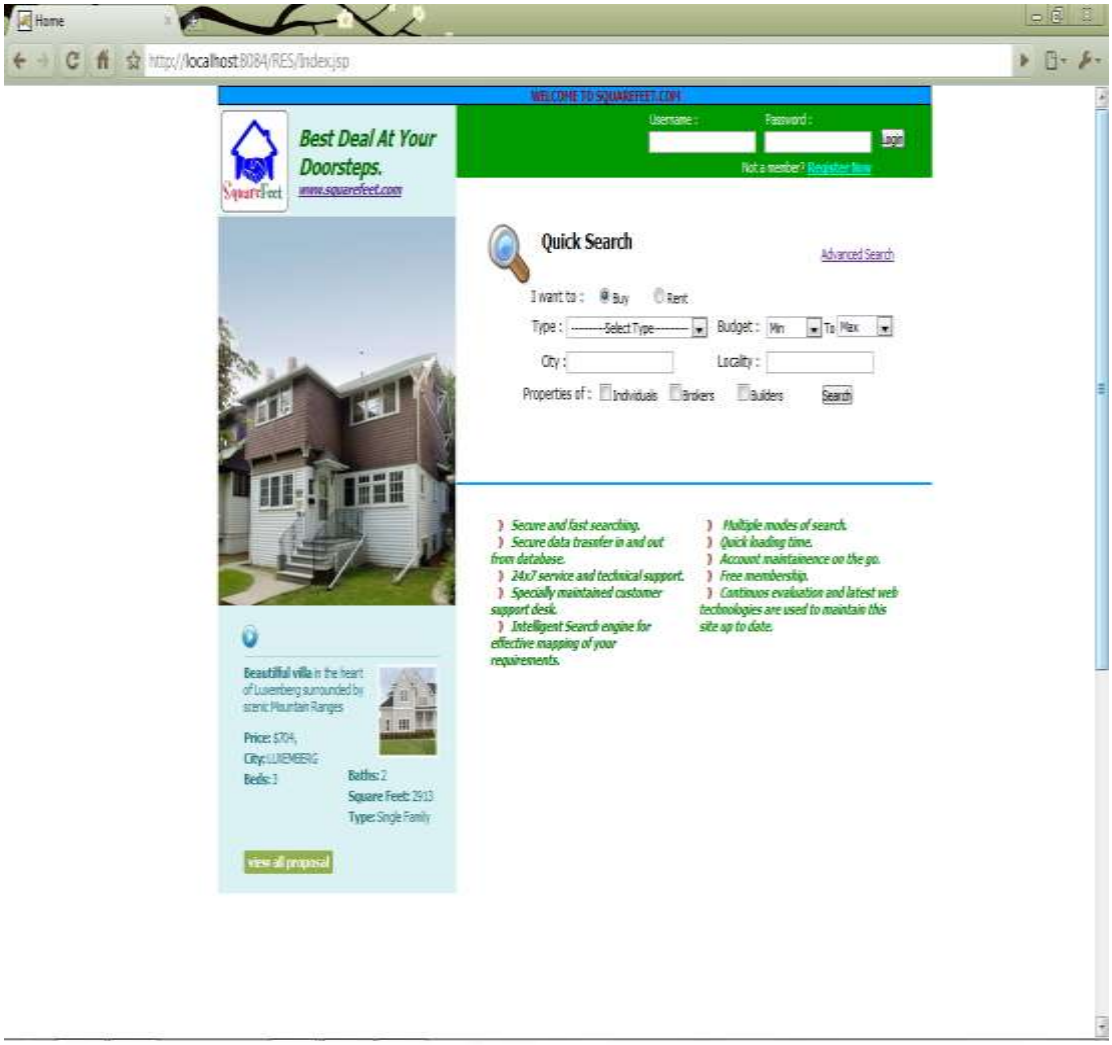


Fig.1: Home Page

This snapshot shows the view of our homepage. It demonstrates the features like:

- ❖ User login section top right side.
- ❖ Quick search section in middle.
- ❖ Ads Space in bottom left.
- ❖ Features offered by our site.

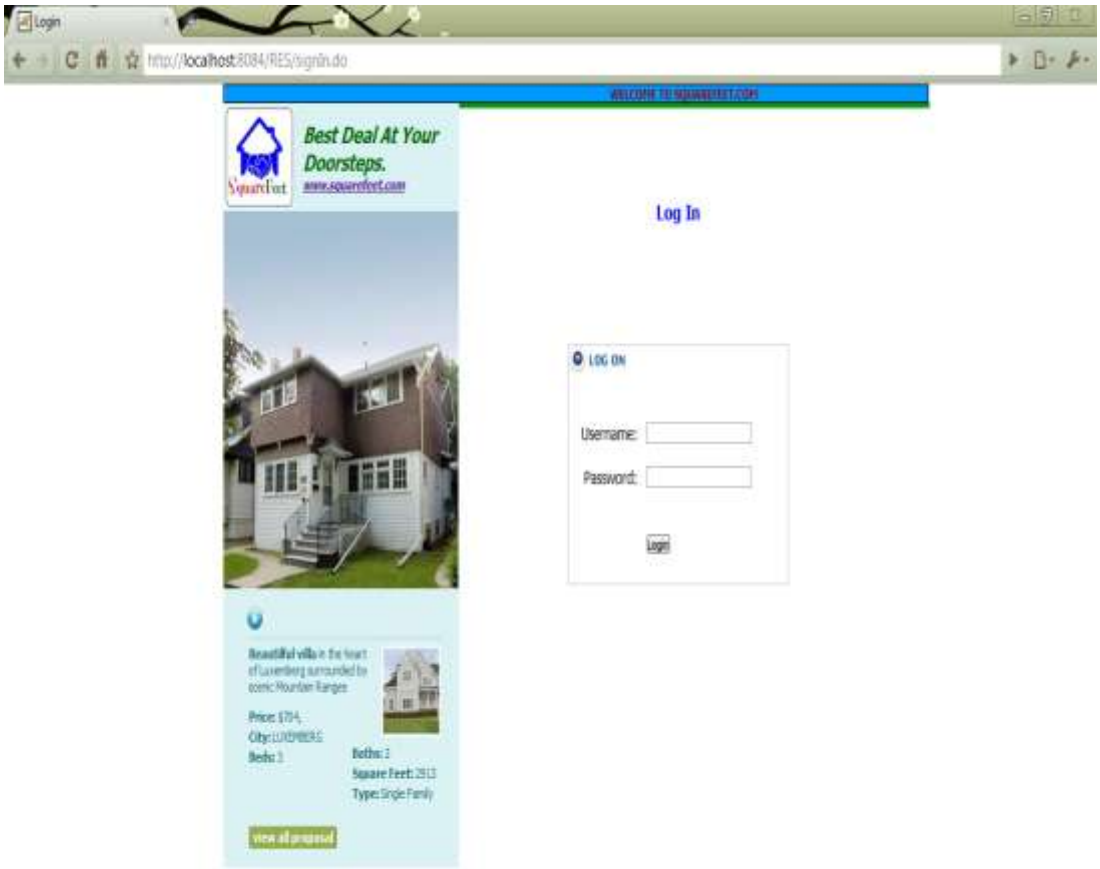


Fig. 2: Alternative Login page

This Snapshot demonstrates the alternative login page provided to user. This page appears if we make any false attempt of login on home page.

This page contains following components:

- User login space.
- Space for ads.

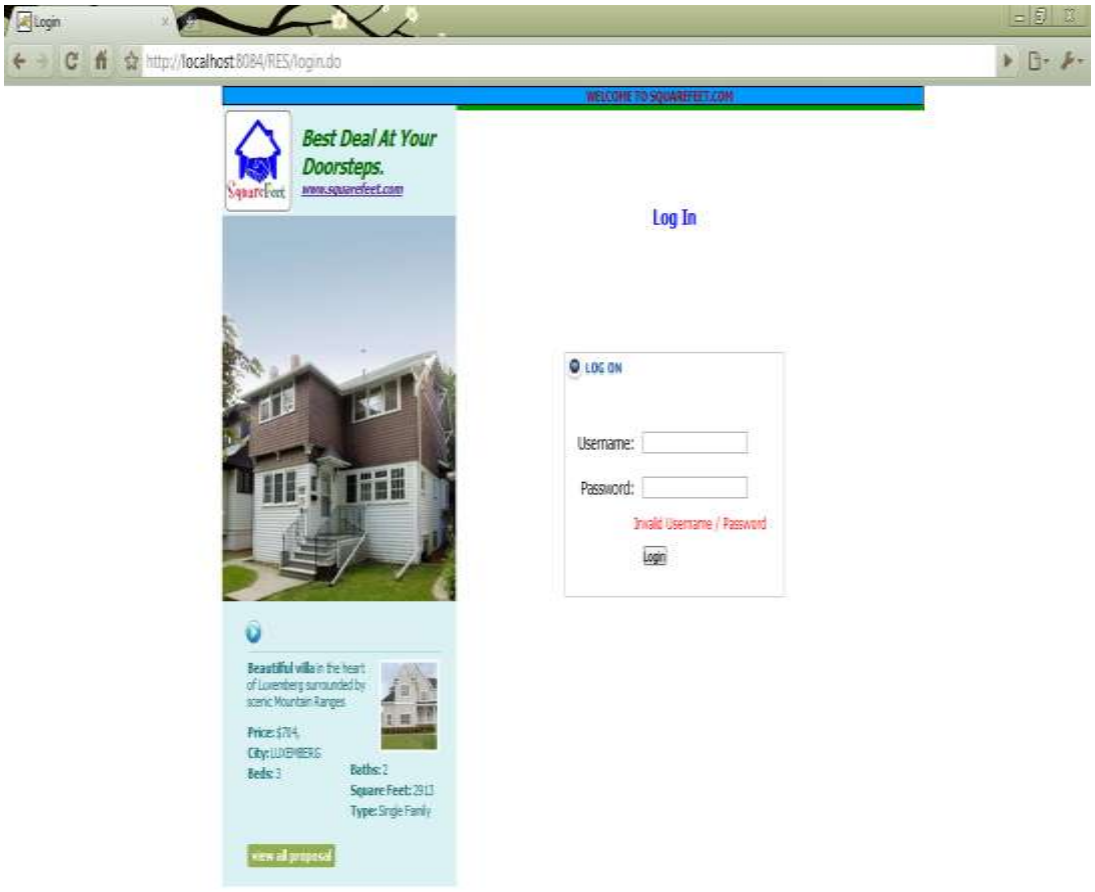


Fig. 3: Invalid Login Demonstration

This snapshot demonstrates the message which is displayed when we fill an invalid username or password. We have used JSTL components to display this message.

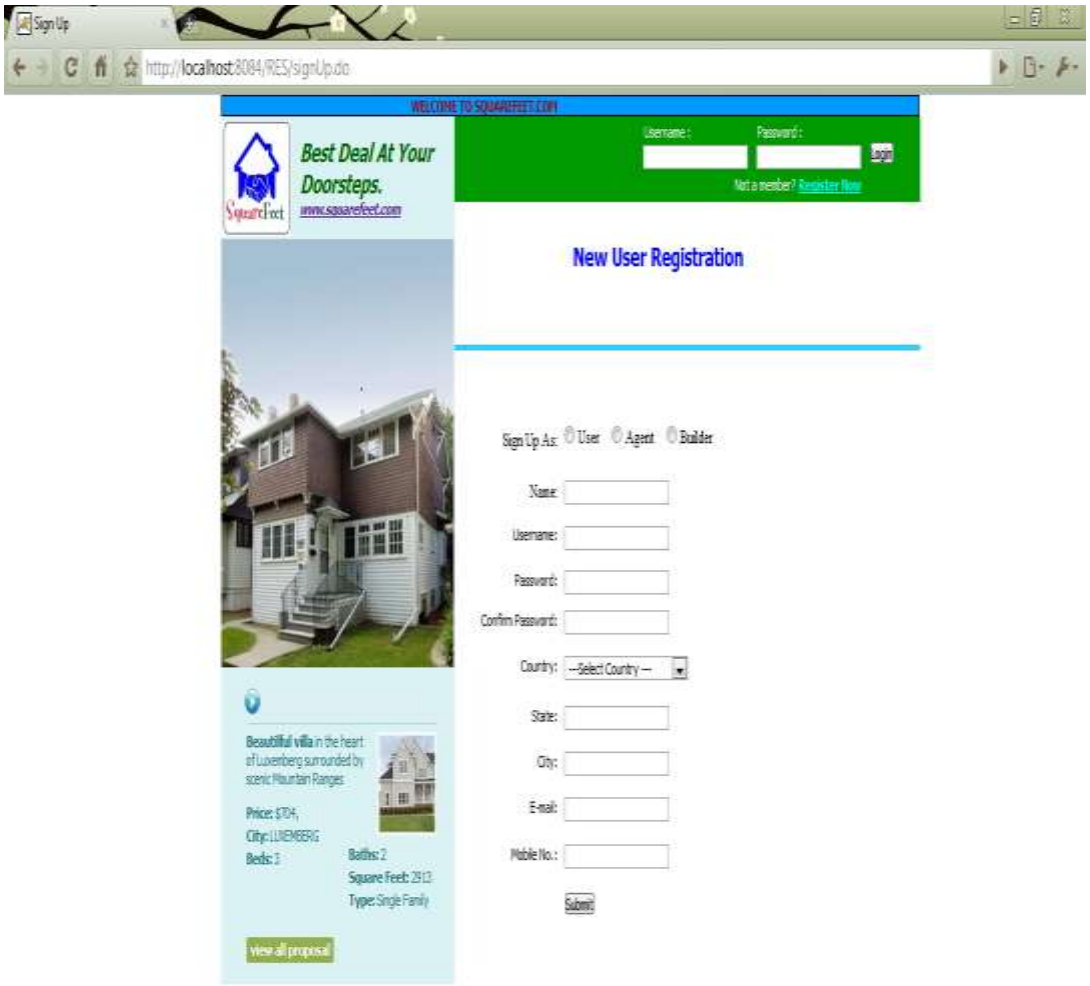


Fig.4: New User Sign Up form

This snapshot demonstrates the new user registration page. All these details corresponding to any new user are fed into the database and are displayed later on when the user logs into his account.

This snapshot demonstrates following features:

- User login section.
- Details to be filled in by new user.
- Role to be selected by new user : user, agent, builder
- Space for ads.

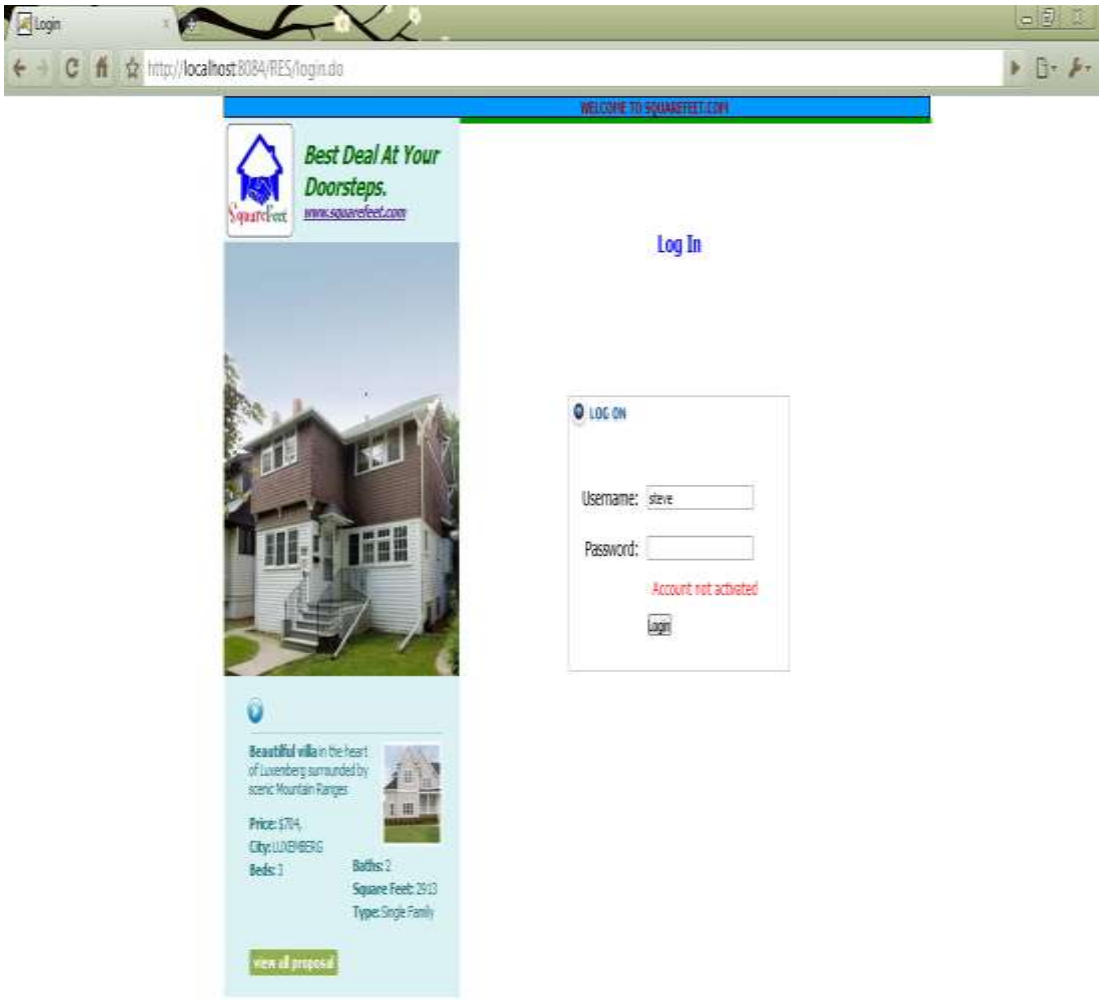


Fig. 5: Account not activated.

We have provided a feature in our website which governs the activation of any newly registered user via a secure admin account. Only registered and activated users can access features of website other than quick search.
This activation of account is essential for any newly registered user to sign in.

Fig. 6: Admin Home Page

This snapshot demonstrates the admin home page. From where the administrator can manage the various newly registered user's account.
He can either activate or can delete any records from this page. Only activated users can access further components of website.

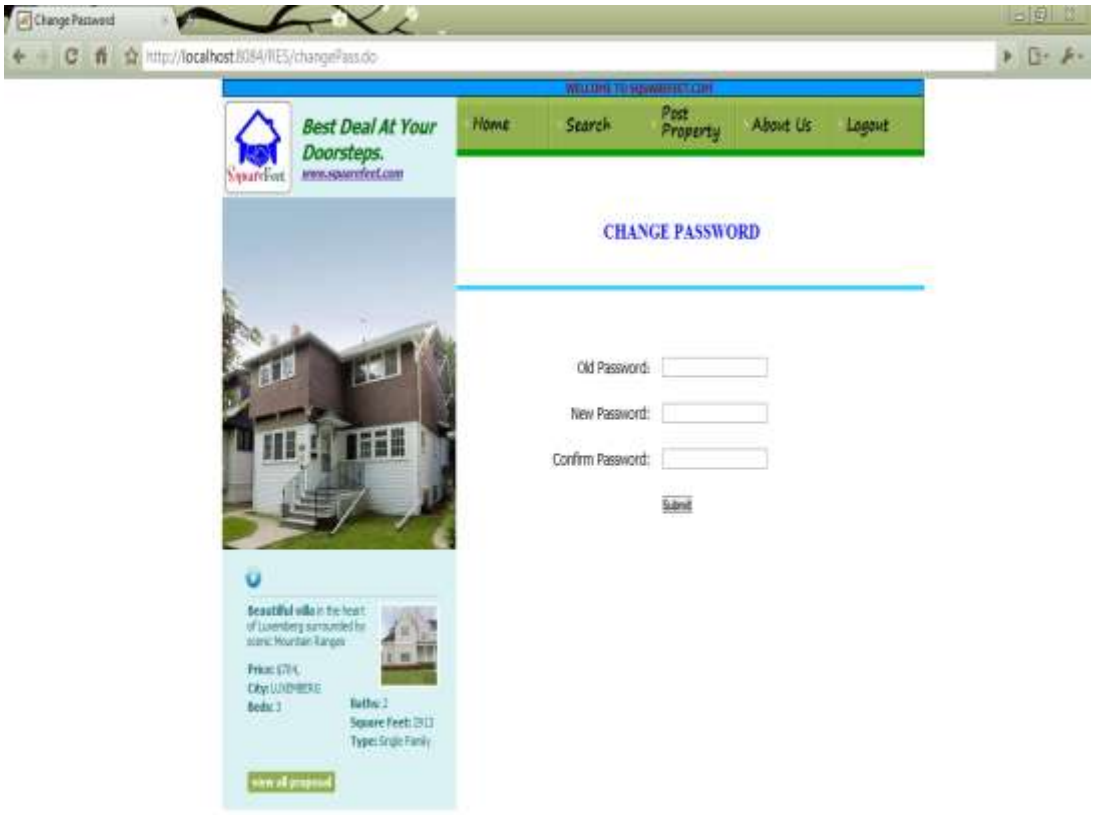


Fig.7: Change password page

This snapshot demonstrates the page in which registered and logged in users can change their passwords.
This page also shows the following buttons:

- Home
- Search
- Post Property
- About Us
- Logout

These buttons transfers our view to the page requested.

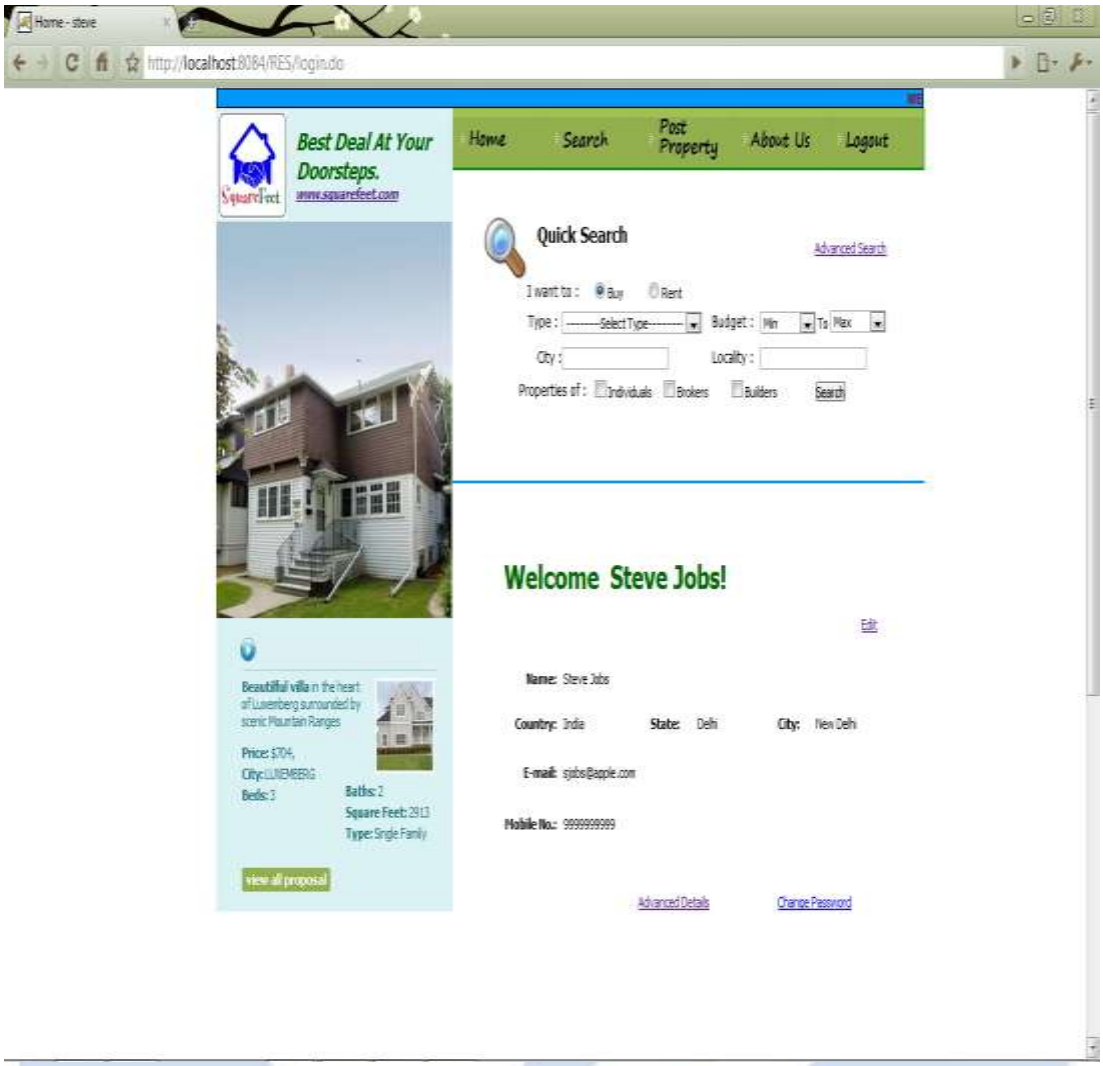


Fig.8: Builder/Agent Home Page

This snapshot demonstrates the home page of agent or builder. It demonstrates following components:

- Details of user
- Quick search module
- General navigation components
- Ads space.
- Link to change password
- Link to access to additional details

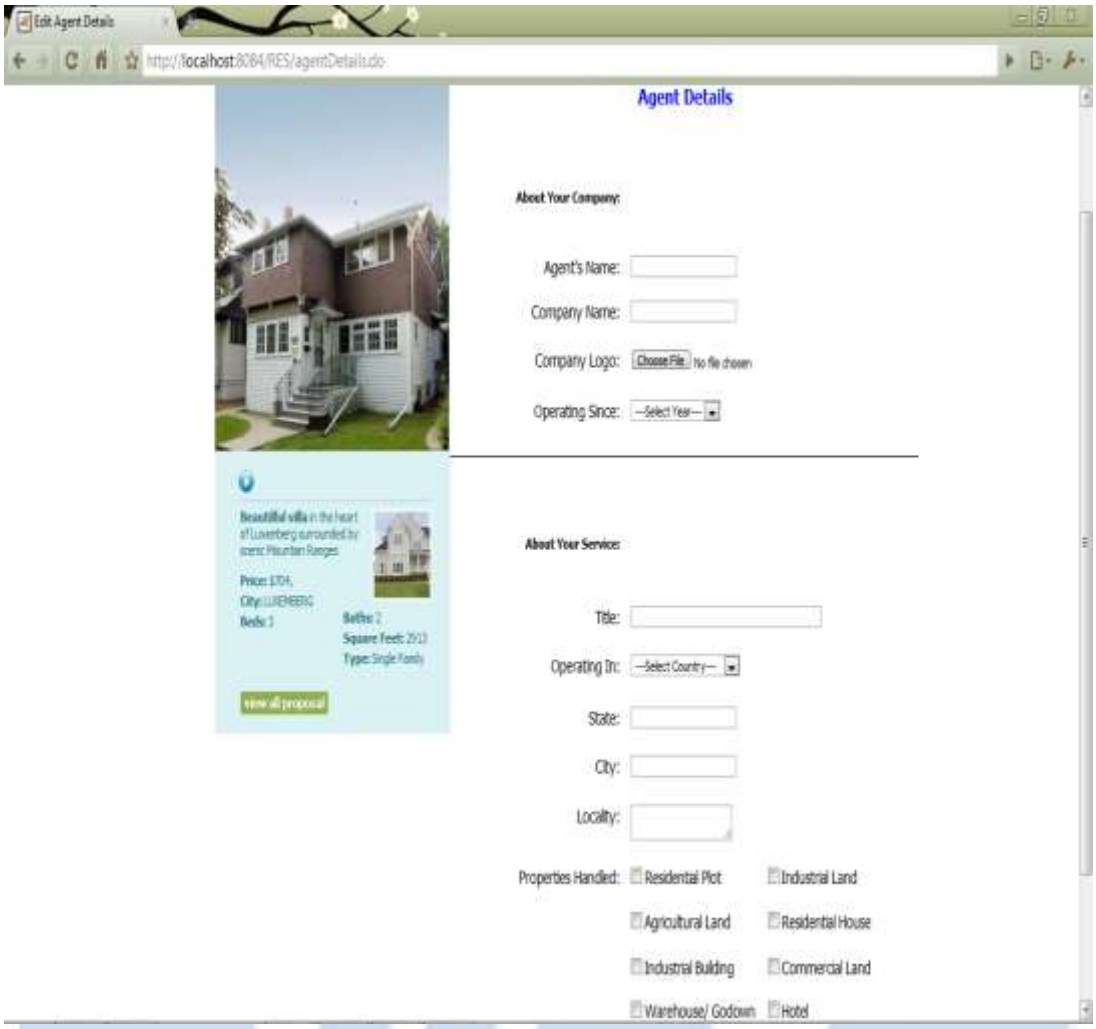


Fig.9: Agent/Builder Details Form

This form gets the generic details from agent and builder and loads them into the database. This form also asks about the general details of the company for which the agent or builder deals.

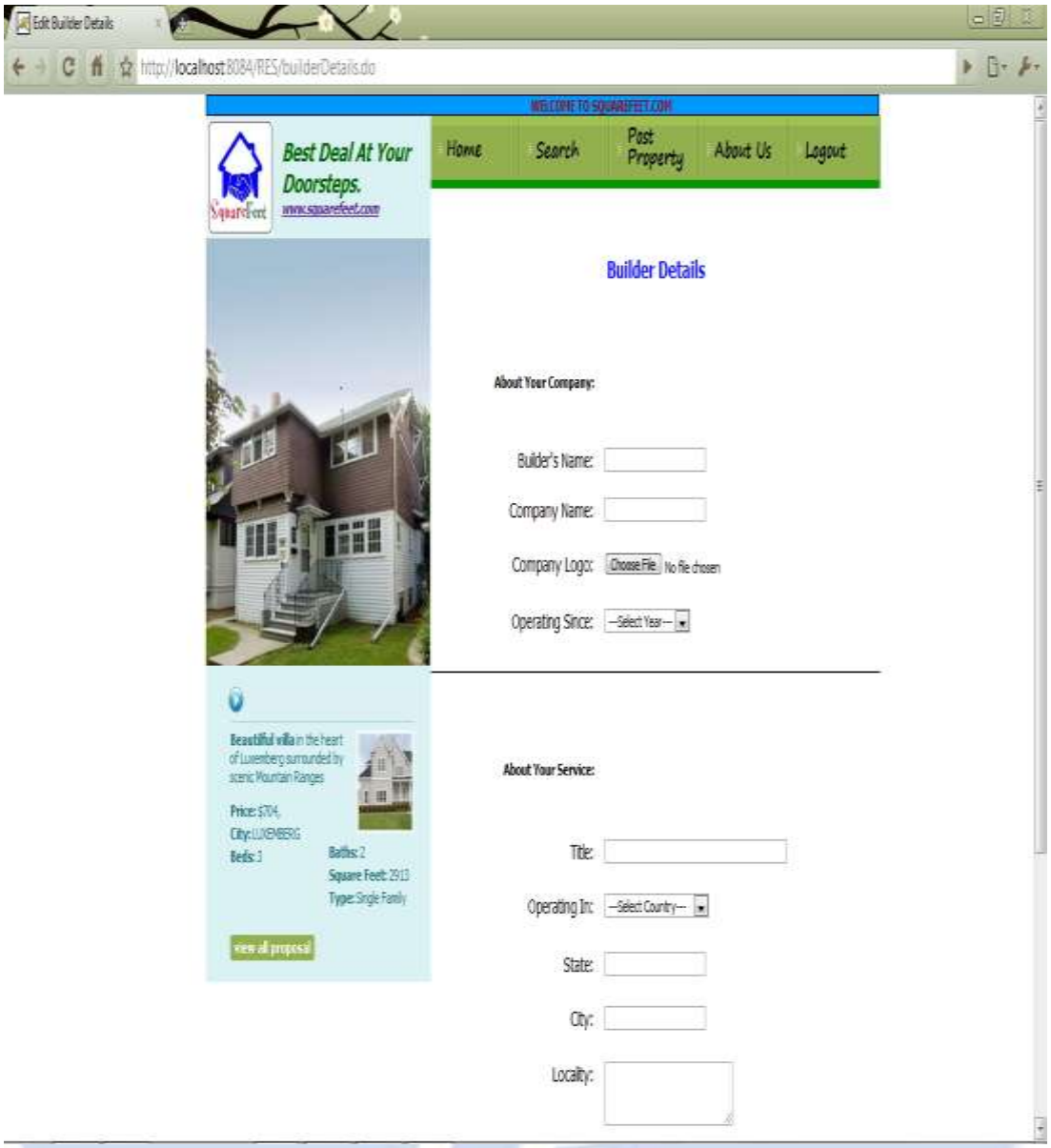


Fig. 10: Additional Builder/Agent Details

This snapshot demonstrates the various components like:

- Additional details of company
- Additional details of builder/agent
- Uploading image(logo) of company
- Navigation controls
- Ads

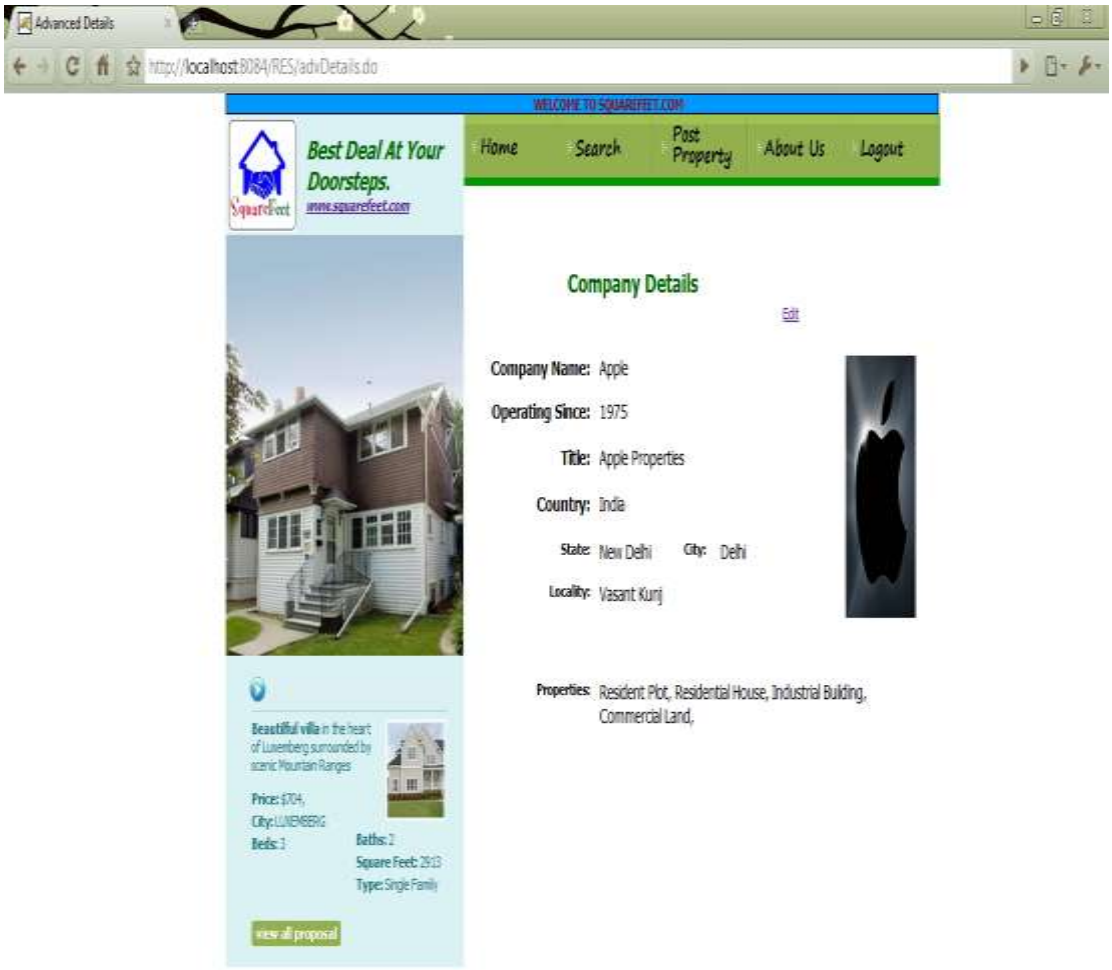


Fig.11: Company details Page

This page demonstrates following features:

- Company logo which was loaded in Fig.10
- Company details
- Link to edit details
- Navigation tools
- Ads

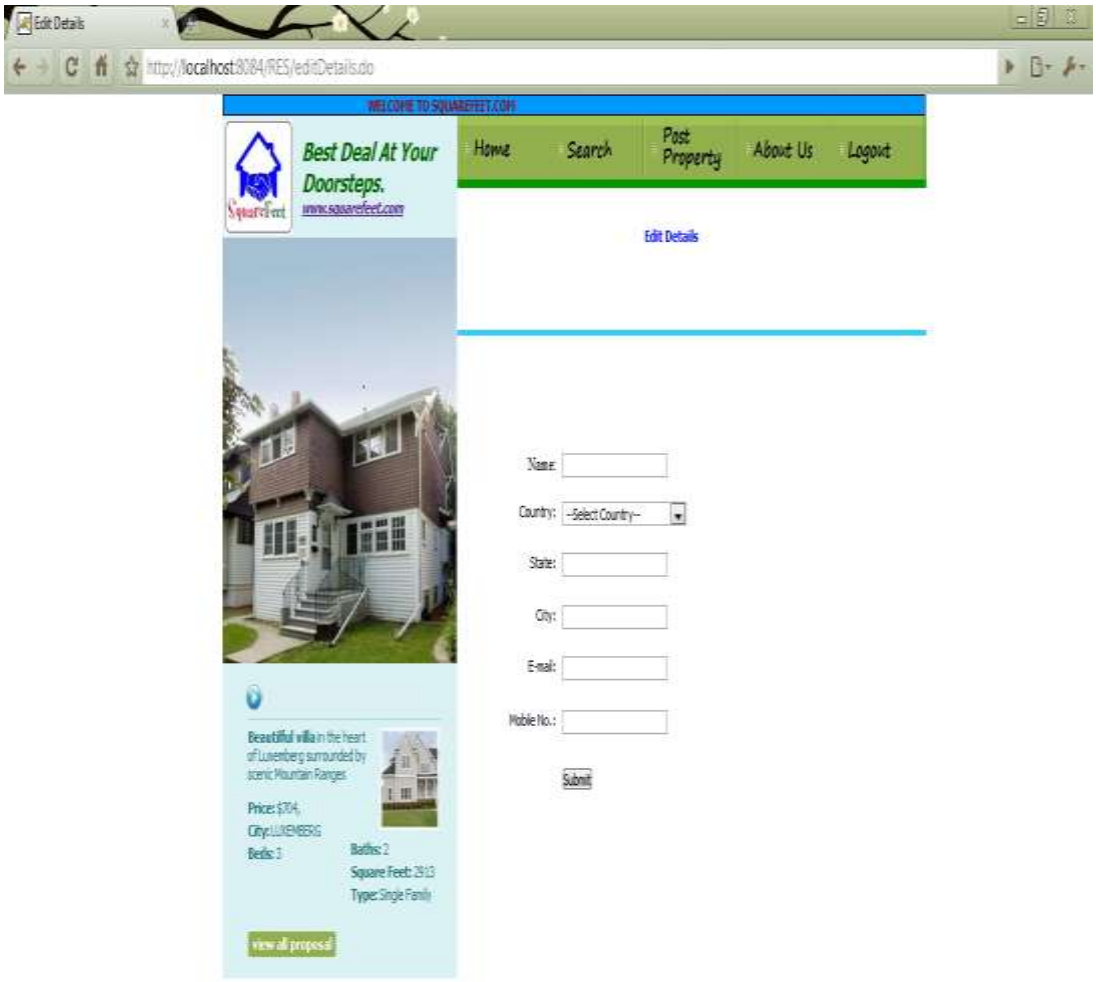


Fig.12: Edit Details Page

This page demonstrates the various fields which can be edited once you are on the company details web page. These new details are overwritten on the previous ones in the database.

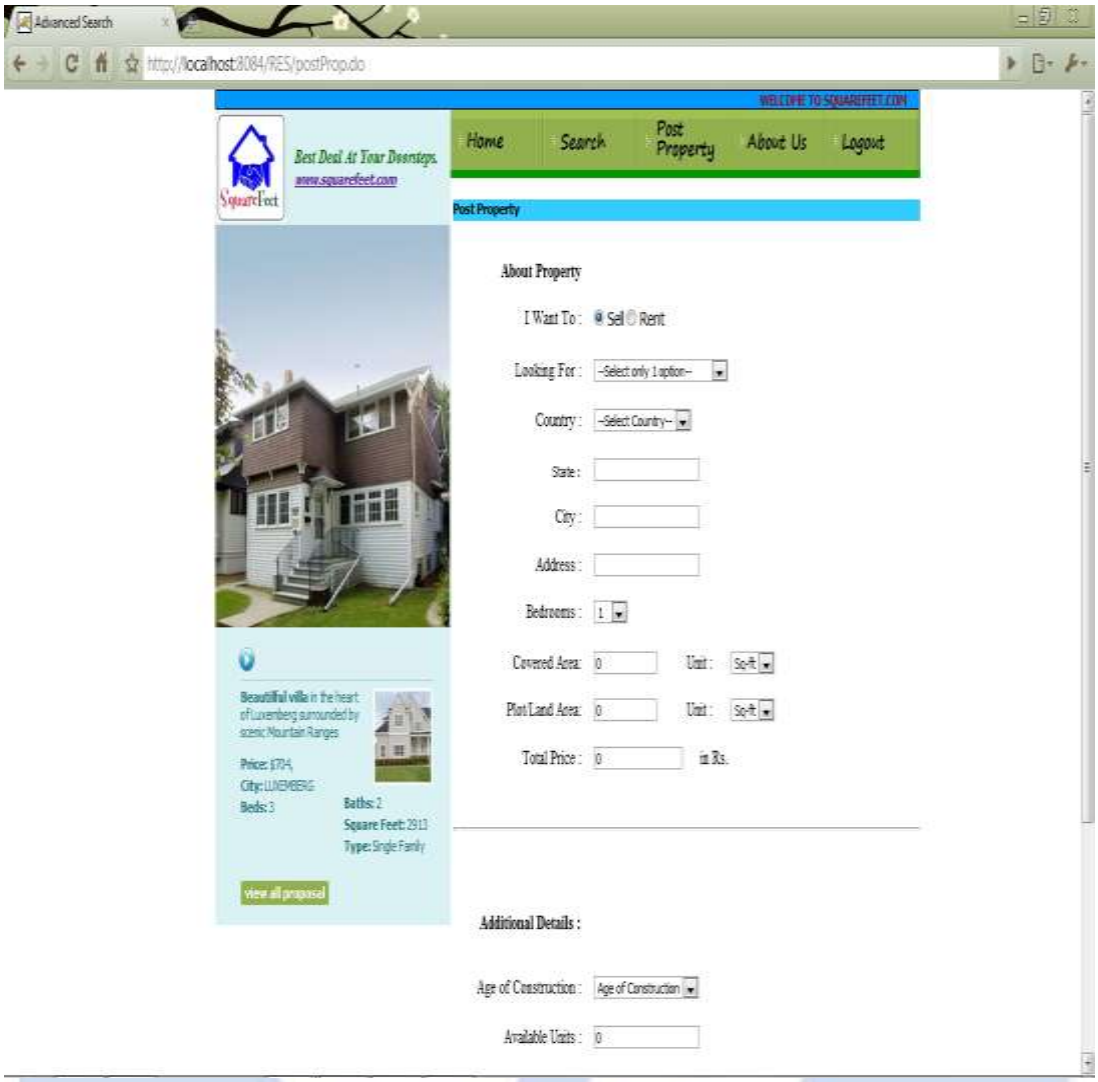


Fig.13: Post Property Page

On this page we can post the details of our property which we want to sell or rent. We can also post additional of our property.

This page contains following components:

- Navigation controls
- Normal Details about property
- Additional details about property
- Ads

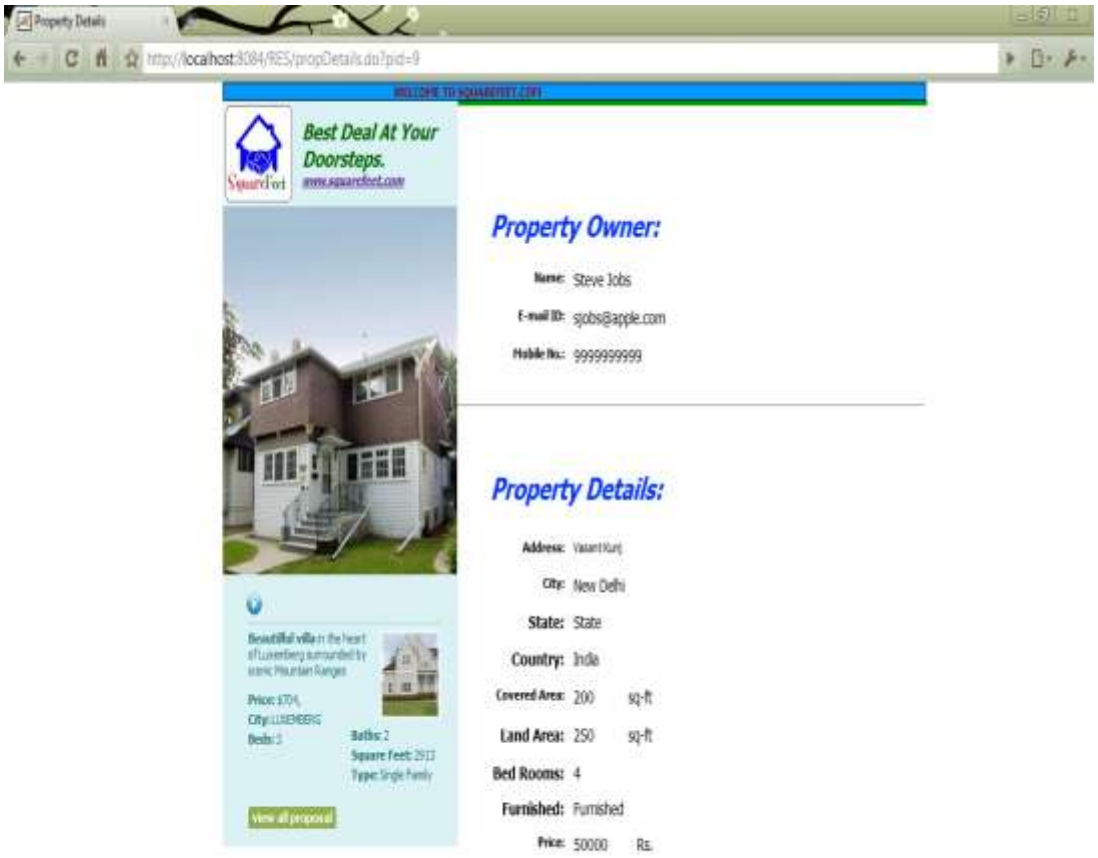


Fig. 14: Property Details Page

This page displays the details of the property we searched for and also the details of its owner.

This page is displayed after the user clicks on the search results. Whenever user searches a property and gets a result he can choose to view detailed information about that property and hence he is guided to this page.

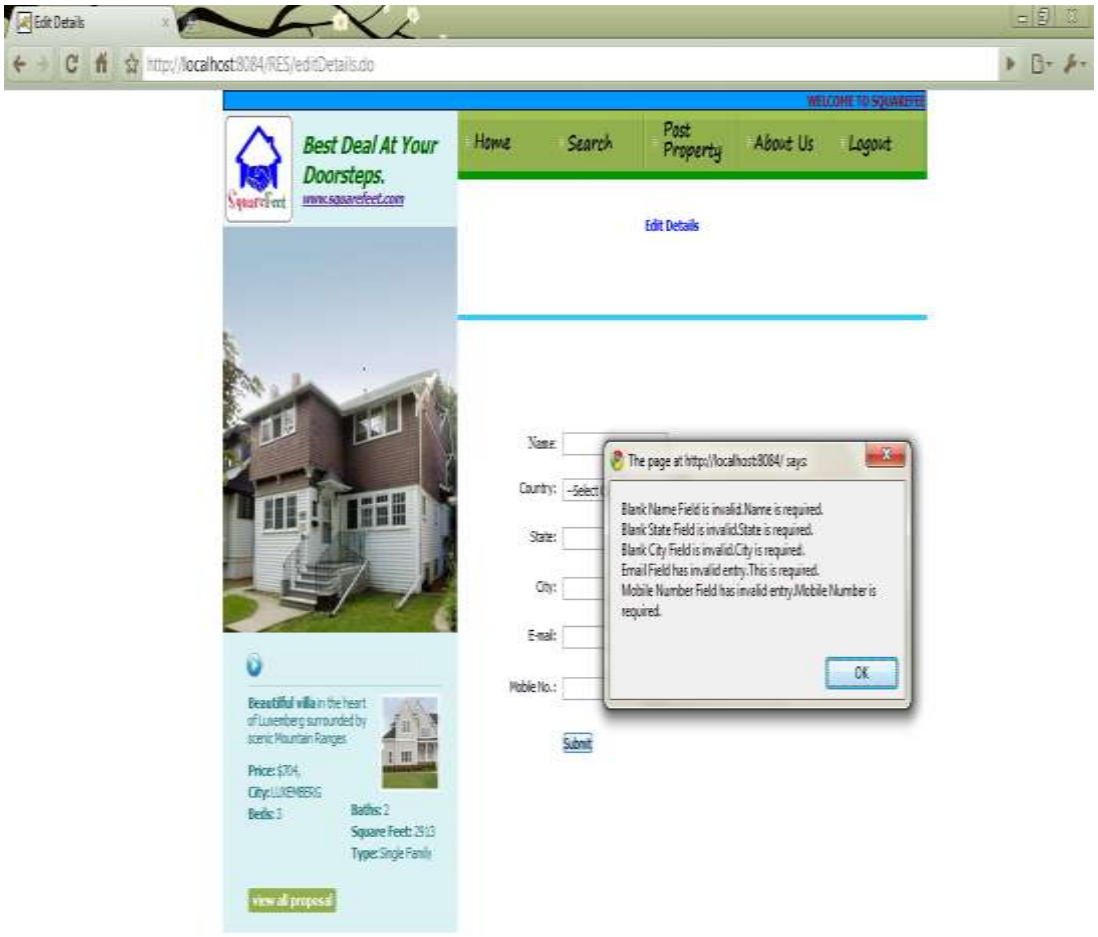


Fig.15: Validator Framework demonstration

This page demonstrates the application of Validator framework to our project. We need to fill the correct and appropriate data. For e.g. Email should not be blank and should be in the format abc@xyz.com

Similarly the mobile number should be exactly equal to 10 digits.

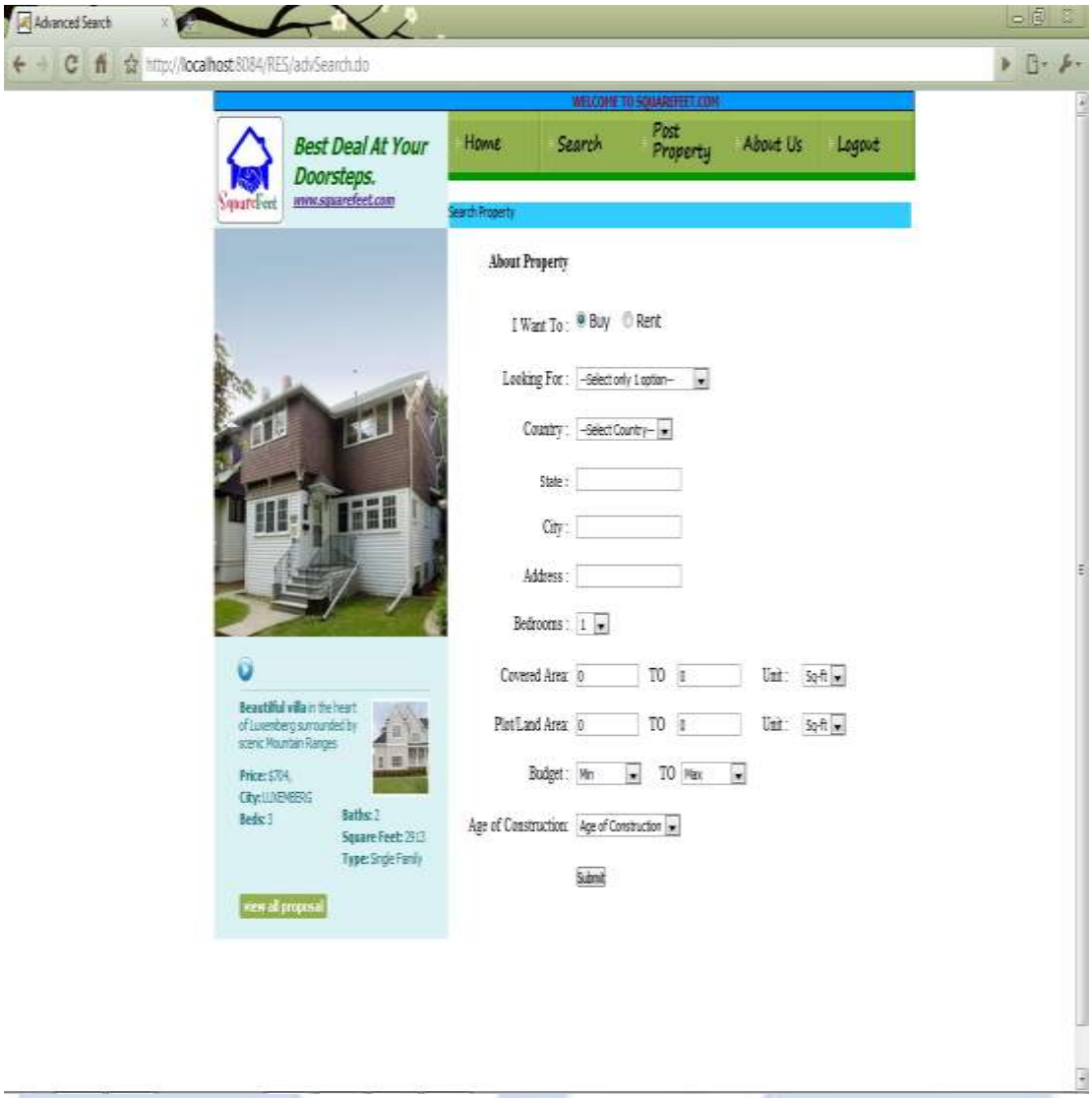


Fig.16: About Property

This page demonstrates the details of property we want to search. User needs to fill the details of property in the fields provided, the requirements of the user is then compared with the data existing in the database to find the best possible results. These results are then displayed to user performing the search.

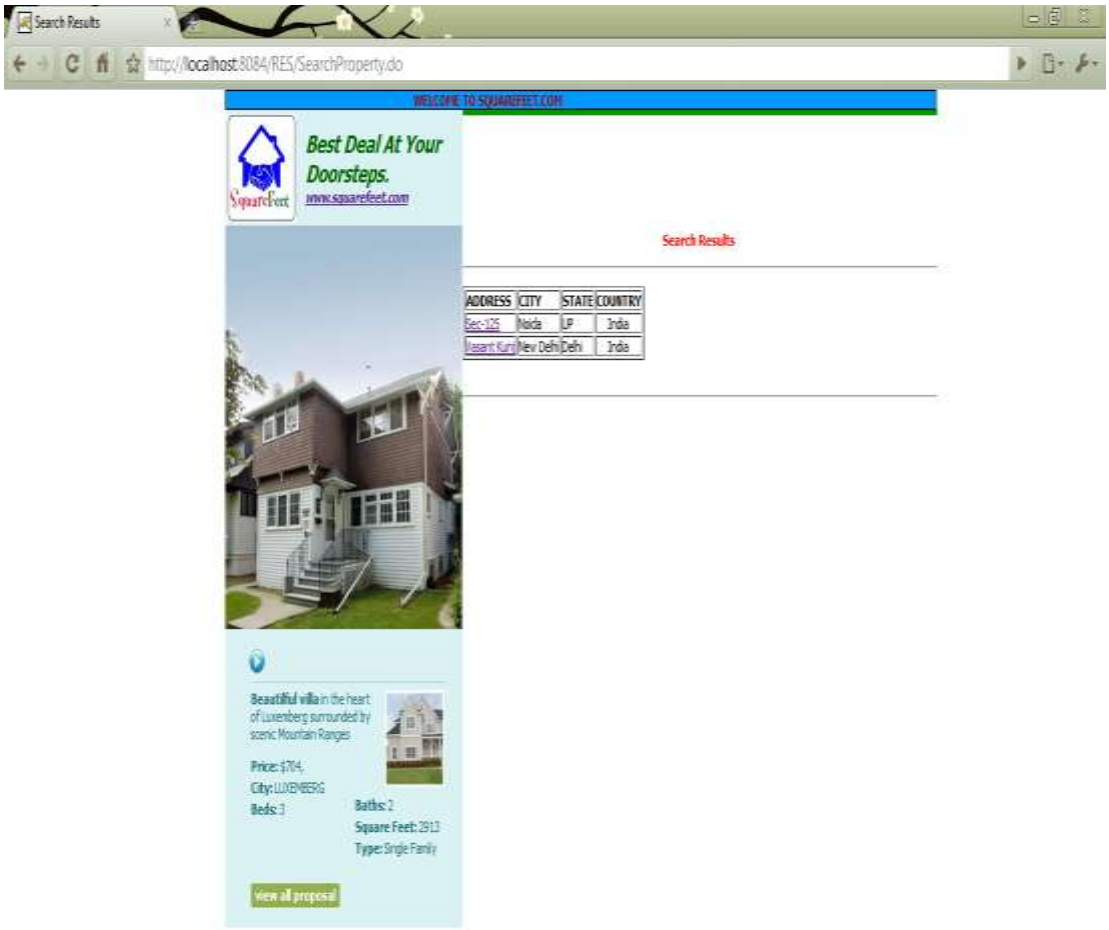


Fig.17: Search Results Page

This page demonstrates the result of search performed by the user. If any property matching the user requirements is present in the database then it is displayed here. If we need to obtain the further details of the property then we can click on any of the property displayed here and a new window opens with further property details.

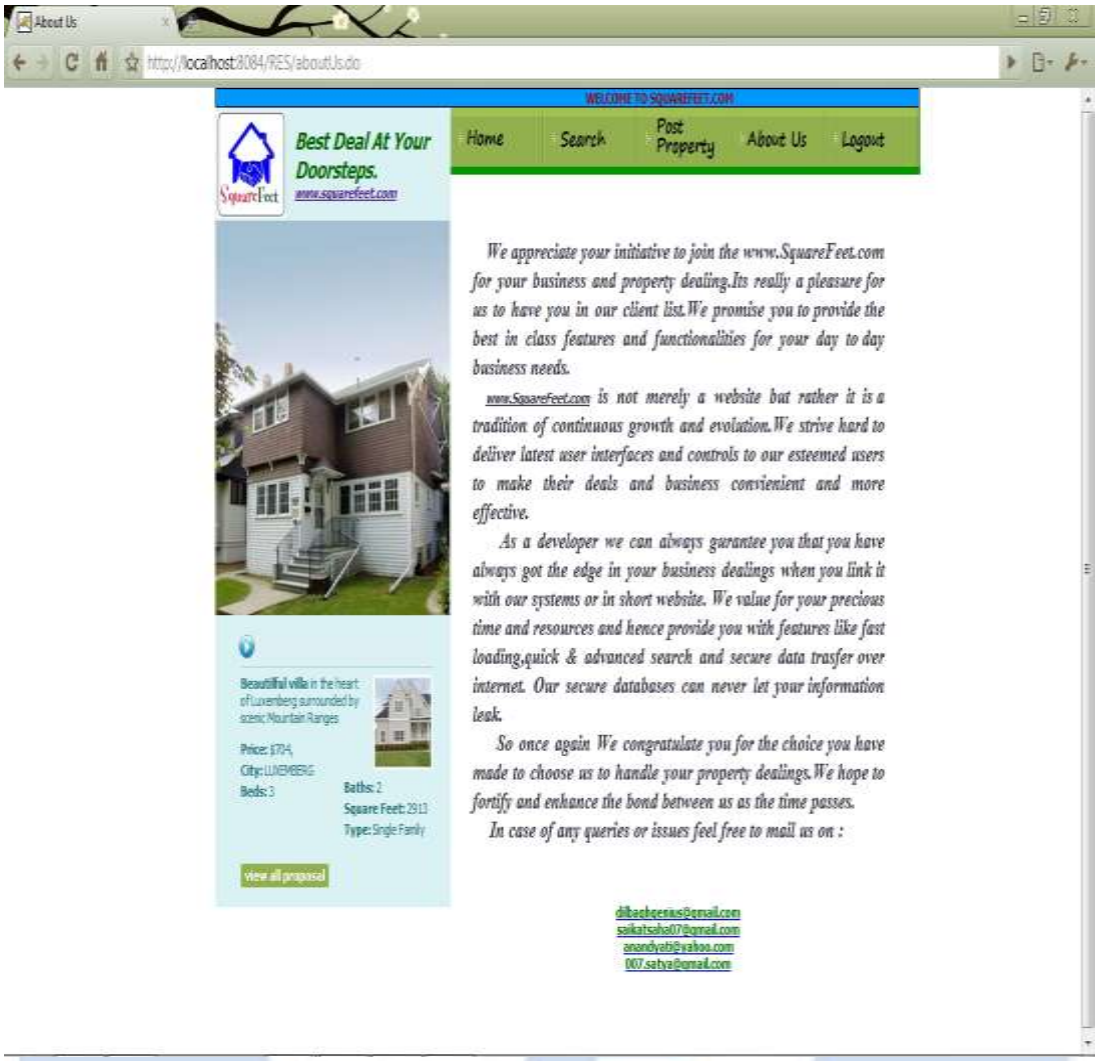


Fig.18: About Us Page.

This page demonstrates the information about the site, its functionality and also serves as an notion of thanks to the customers using our website. This page is a static page which conveys the developer's message to their clients. It contains the email addresses of the developers which can be used in case of any technical problem or to forward any feedback.

REFERENCES

Bibliography:

1.) A Programmer's Guide to Java Certification by Khalid A. Mughal, Addison Wesley Publication (ISBN : 0201596148).
2.) Head First Servlets & JSP by Kathy Sierra, 2nd Edition, O'Reilly Publication.
3.) Professional Java Server Programming by Allamaraju ,J2EE 1.3 Edition, Apress Publication.
4.) Practical Apache Struts 2 Web 2.0 Projects by Ian Roughley, Apress Publication(ISBN10: 978-1-59059-903-7).
5.) Java Servlet Programming - Jason Hunter, William Crawford – 2001
6.) The Definitive Guide to MySQL 5 - Michael Kofler, David Kramer – 2005
7.) Java Servlet and JSP Cookbook: Practical ... - Bruce W Perry – 2004
8.) Java for the web with Servlets, JSP and EJB- a developer's guide to scalable j2ee solutions by Budi Kurniawan, by New Riders Publication.
9.) Thinking in java by Bruce Eckel(2002,ISBN: 0131002872)

Internet Resources:

1.) msdn.microsoft.com/en-us/library/aa581778.aspx
2.) struts.apache.org/
3.) www.roseindia.net/**struts**/**struts2**/**struts**-**2**-architecture.shtml
4.) www.**struts2**.net/
5.) dev.mysql.com/doc/.../5.1/.../pluggable-storage-overview.html
6.) www.javabeat.net/.../51-introduction-to-**struts**-**validator**-**framework**.html
7.) onjava.com/pub/a/onjava/2002/12/11/jakartastruts.html
8.) struts.apache.org/1.2.9/userGuide/dev_tiles.html
9.) www.webopedia.com/**j2ee**.html