**Introduction**

Rainbow Tree Apartments is a house management system. It has the information about the availability status of houses, rental and lease information, amenities and other facilities in that community. By visiting applications like this, one can choose their homes according to their needs. This application is very helpful for the people who are new to the city or for people who are out of the city and want to reserve a house.

**Outline of the project**

Two category people access the application. One will be the administrator and other will be the user. Both Administrators and users should register if they are first time users except the first Admin. The Administrator can login, add or delete user account, update the information of available house rent and leasing information.

A user can login check the facilities, availability of apartment and other essential information. If the user likes it, he can choose from the list of options available and can send a request to admin. The admin will login, check the notifications and can accept or reject the request of the user. Once the request is accepted, the record is inserted to database and will not be displayed for the other users. The rent amount is calculated based on the lease duration and choice of options a user chooses. If the user moved out, then his account would be deleted by Admin.

**Technologies used:**

**Front End:** HTML, CSS, JSP, JAVA SCRIPT

**Logic:** JAVA, Servlets

**Database platform:** Oracle

**Server:** Apache Tomcat

**Frame work:** Springs.

**UML Diagrams:**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML describes the real time systems it is very important to make a conceptual model and then proceed gradually. Conceptual model of UML can be mastered by learning the following three major elements:

1. UML building blocks

2. Rules to connect the building blocks

3. Common mechanisms of UML

The building blocks of UML can be defined as:

* a) Things
* b) Relationships
* c) Diagrams

# (a) Things:

Things are the most important building blocks of UML. Things can be:

* 1.Structural
* 2.Behavioral
* 3.Grouping
* 4.Annotational

**1. Structural things**: The Structural things define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

## A) Class:

Class represents set of objects having similar responsibilities.

class

## B) Interface:

Interface defines a set of operations which specify the responsibility of a class.

Interface

## C) Collaboration:

Collaboration defines interaction between elements.

Collaboration

## D) Use case:

Use case represents a set of actions performed by a system for a specific goal.

Use case

## E) Component:

Component describes physical part of a system.

Component

## F) Node:

A node can be defined as a physical element that exists at run time.



# 2. Behavioral things: A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:

## A) Interaction:

Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

Interaction

## B) State machine:

State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.



# 3. Grouping things: Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

## Package:

Package is the only one grouping thing available for gathering structural and behavioral things.



4. Annotational things: Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note is the only one Annotational thing available.

## Note:

A note is used to render comments, constraints etc of an UML element.

Note

# (2) Relationship:

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

1. Dependency:

Dependency is a relationship between two things in which change in one element also affects the other one.

Dependency

1. Association:

Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

Association

1. Generalization:

Generalization can be defined as a relationship which connects a specialized element with a generalized element. It describes inheritance relationship in the world of objects.Generalization

1. Realization:

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.

Realization

# UML Diagrams:

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.

UML includes the following nine diagrams and the details are described in the following chapters.

1. **Class diagram:** It describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.
2. **Object diagram:** It shows a complete or partial view of the structure of a modelled system at a specific time.
3. **Use case diagram:** It shows the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.
4. **Sequence diagram:** It shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespan of objects relative to those messages
5. **Collaboration diagram:** It is a specific type of interaction diagram, where the focus is on timing constraints.
6. **Activity diagram:** It represents the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
7. **State chart diagram:** It is standardized notation to describe many systems, from computer programs to business processes.
8. **Deployment diagram**: It serves to model the hardware used in system implementations, and the execution environments and artifacts deployed on the hardware.
9. **Component diagram:** It depicts how a software system is split up into components and shows the dependencies among these components.

**Use case Diagram:**

It shows the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases. The complete use case diagram for the project is given as:

**Use case for User**

****

**Use Case For admin**



**Class Diagram**

It describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.

****

**Sequence diagram:**

It shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespan of objects relative to those messages.

Sequence diagram for user:

****

Sequence diagram for Admin:



**Collaboration Diagram**

It is a specific type of interaction diagram, where the focus is on timing constraints.

Collaboration Diagram for User:



Collaboration Diagram for Admin:



**Activity Diagram**

It represents the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Activity Diagram for User:

****

Activity Diagram for Admin:



**State Chart Diagram**

It is standardized notation to describe many systems, from computer programs to business processes.

State Chart Diagram for User:



State Chart Diagram for Admin:

****

**Deployment Diagram**

It serves to model the hardware used in system implementations, and the execution environments and artifacts deployed on the hardware.



**6. TECHNOLOGY DESCRIPTION**

**6.1. Software Description:**

* **HTML:**

Hyper Text Markup Language (HTML) is a language used to create hypertext documents that have hyper links embedded in them. It consists of tags embedded in the text of a document with HTML. We can build web pages or web documents. It is basically a formatting language and not a programming language. The browser reading the document interprets mark up tags to help format the document for subsequent display to a reader. HTML is a language for describing structured documents. HTML is a platform independent. WWW (World Wide Web) pages are written using HTML. HTML tags control in part the representation of the WWW page when view with web browser. The browser interprets HTML tags in the web document and displays it. Different browsers show data differently. Examples of browsers used to be web pages include:

1. Netscape
2. Internet Explorer

**JAVA:**

Java was developed by James Gosling, Patrick Naught on, Chris Wrath, Ed Frank and Mike Sheridan at SUN Micro Systems Incorporation in 199I.It took 18 months to develop the first working version .This language initially called "OAK", but was renamed " JAVA " In 1995.Before the initial implementation of OAK in 1992 and the public announcement of Java in 1995, many more contributed to the design and evolution of the language.

### The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - * Simple
      * Architecture neutral
      * Object oriented
      * Portable
      * Distributed
      * High performance
      * Interpreted
      * Multithreaded
      * Robust
      * Dynamic
      * Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. The following figure illustrates how this works.



*Fig: Java Flow*

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



*Fig: Some OS Supporting Java*

### The Java Platform

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

* The Java Virtual Machine (JVM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms. The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware



*Fig 4.1.3. Java Platform*

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

Java is a powerful but lean object oriented programming language. Lt has generated a lot of excitement because it makes it possible to program for internet by creating applets programs that can be embedded in a web page. The context of an apple is limited only by one’s imagination. Java is more than programming language for writing applets. It is being used more and more for writing standalone application as well.

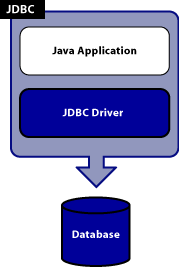
* 1. **Database Description:**
* **JAVA Database Connectivity**: (JDBC)

JDBC is an API developed by Sun Microsystems that provides a standard way to access data using the Java programming language. Using JDBC, an application can access a variety of databases and run on any platform with a Java Virtual Machine. It isn’t necessary to write separate applications to access different database systems, JDBC allows you to write one application that can send SQL statements to different database systems.

* **How Does JDBC Work?**

JDBC makes it possible to do the following things within a Java application:

* Establish a connection with a data source
* Send queries and update statements to the data source
* Process the results



*Fig 4.2.1. JDBC Architecture*

## *Features of JDBC*

* Retrieval of auto-generated keys
* Multiple open result sets

**JDBC Performance**

* Retrieving only required data
* Selecting functions that optimize performance
* Managing connections and updates

### Establishing a Connection

The first thing to do, of course, is to install Java, JDBC and the DBMS on your working machines. Since we want to interface with MySql database, we would need a driver for this specific database as well. To access a database, a connection must be opened between our program (client) and the database (server). This involves a step:

Class.forName("com.mysql.jdbc.Driver");

**Creating JDBC Statements**

A JDBC Statement object is used to send your SQL statements to the DBMS, and should not to be confused with an SQL statement. A JDBC Statement object is associated with an open connection, and not any single SQL Statement. You can think of a JDBC Statement object as a channel sitting on a connection, and passing one or more of your SQL statements to the DBMS. The following code snippet, using our Connection object con, does it for you:

Statement stmt = con.createStatement ();

## *What is JSP?*

Java Server Pages (JSP) is a technology that lets you mix regular, static HTML with dynamically-generated HTML. Many Web pages that are built by CGI programs are mostly static, with the dynamic part limited to a few small locations. But most CGI variations, including servlets, make you generate the entire page via your program, even though most of it is always the same. JSP lets you create the two parts separately.

### Handling Errors with Exceptions

The truth is errors always occur in software programs. Often, database programs are critical applications, and it is imperative that errors be caught and handled gracefully. Programs should recover and leave the database in a consistent state. Rollback-s used in conjunction with Java exception handlers is a clean way of achieving such a requirement. The client (program) accessing a server (database) needs to be aware of any errors returned from the server. JDBC give access to such information by providing two levels of error conditions: SQLException and SQLWarning. SQLExceptions are Java exceptions, which, if not handled, will terminate the application. SQLWarnings are subclasses of SQLException, but they represent nonfatal errors or unexpected conditions, and as such, can be it.

**JAVA SERVER PAGES (JSP)**

A server side technology, Java Server Pages are an extension to the Java servlet technology that was developed by Sun. JSPs have dynamic scripting capability that works in tandem with HTML code, separating the page logic from the static elements – the actual design and display of the page – to help make the HTML more functional (i.e. dynamic database queries).

A JSP is translated into Java servlet before being run and it processes HTTP requests and generates responses like any servlet. However, JSP technology provides a more convenient way to code a servlet. Translation occurs the first time the application is run. A JSP translator is triggered by the. JSP file name extension in a URL. JSPs are fully interoperable with servlets. You can include output from a servlet or forward the output to a servlet and a servlet can include output from a JSP or forward output to a JSP.

## *Advantages of JSP*

* Vs. Active Server Pages (ASP). ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
* Vs. Pure Servlets. JSP doesn’t give you anything that you couldn’t in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.
* Vs. JavaScript. JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client’s environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can’t access server-side resources like databases, catalogs, pricing information, and the like.
* Vs. Static HTML. Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

**JSP Directives**

JSP pages use JSP directives to pass instructions to the JSP engine. These may include the following:

* JSP page directives communicate page-specific information, such as buffer and thread information or error handling.
* Language directives specify the scripting language, along with any extensions.
* The include directive (shown in the example above) can be used to include an external document in the page. A good example is a copyright file or company information, file – it is easier to maintain this file in one central location and include it in several pages than to update it in each JSP page. However, the included file can also be another JSP file.
* A taglib directive indicates a library of custom tags that the page can invoke.

**JSP Tags:**

Most JSP processing will be implemented through JSP-specific XML-based tags. JSP 1.0 includes a number of standard tags, referred to as the core tags. These include:

**jsp:useBean** This tag declares the usage of an instance of a JavaBeans component. If the Bean does not already exist, then the JavaBean component instantiates and registers the tag.

**Jsp: setProperty** This sets the value of a property in a Bean.

**Jsp: getProperty** This tag gets the value of a Bean instance property, converts it to a string, and puts It in the implicit object “out”.

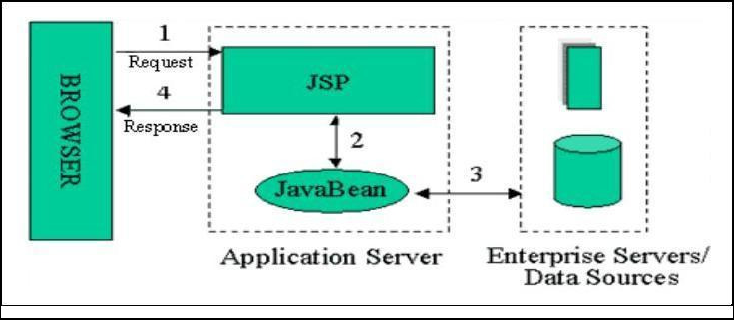
The advantage of tags is that they are easy to use and share between applications. The real power of a tag-based syntax comes with the development of custom tag libraries, in which tool vendors or others can create and distribute tags for specific purposes.

**MVC-1 ARCHITECTURE:**

The main aim of the MVC architecture  is to separate the business logic and application data from the presentation data to the user.

1). **Model:** The model object knows about all the data that need to be displayed. It is model who is aware about all the operations that can be applied to transform that object. It only represents the data of an application. The model represents enterprise data and the business rules that govern access to and updates of this data. Model is not aware about the presentation data and how that data will be displayed to the browser.

2)**View:** The view represents the presentation of the application. The view object refers to the model. It uses the query methods of the model to obtain the contents and renders it. The view is not dependent on the application logic. It remains same if there is any modification in the business logic. In other words, we can say that it is the responsibility of the of the view’s to maintain the consistency in its presentation when the model changes.

3)**Controller:**  Whenever the user sends a request for something then it always go through the controller. The controller is responsible for intercepting the requests from view and passes it to the model for the appropriate action. After the action has been taken on the data, the controller is responsible for directing the appropriate view to the user. In  GUIs, the views and the controllers often work very closely together****

*Fig 4.2.3: MVC1 Architecture*

Apache Tomcat Server:

Introduction

For administrators and web developers alike, there are some important bits of information you should familiarize yourself with before starting out. This document serves as a brief introduction to some of the concepts and terminology behind the Tomcat container. As well, where to go when you need help.

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 powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations.Apache Tomcat includes tools for configuration and management, but can also be configured by editing [XML](http://en.wikipedia.org/wiki/XML) configuration files.

**Directories and Files**

Throughout the docs, you’ll notice there are numerous references to **$CATALINA\_HOME**. This represents the root of your Tomcat installation. When we say, “This information can be found in your $CATALINA\_HOME/README.txt file” we mean to look at the README.txt file at the root of your Tomcat install

* These are some of the key tomcat directories, all relative to $CATALINA\_HOME:
* /bin – Startup, shutdown, and other scripts. The \*.sh files (for Unix systems) are functional duplicates of the \*.bat files (for Windows systems). Since the Win32 (10) command-line lacks certain functionality, there are some additional files in here.
* /conf – Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container.
* /logs – Log files are here by default.
* /webapps – This is where your webapps go with their root folder.

**Components of Tomcat:**

### Catalina

Catalina is Tomcat’s servlet container. Catalina implements [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems)’ specifications for [servlet](http://en.wikipedia.org/wiki/Java_servlet) and Java Server (JSP). In Tomcat, a Realm element represents a “database” of usernames, passwords, and roles (similar to [Unix groups](http://en.wikipedia.org/w/index.php?title=Unix_group&action=edit&redlink=1)) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.

* **Coyote**

Coyote is Tomcat’s HTTP Connector component that supports the HTTP1.1 protocol for the web server or application container. Coyote listens for incoming connections on a specific TCP port on the server and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client.

* **Jasper**

Jasper is Tomcat’s JSP Engine. Tomcat 5.x uses Jasper 2, which is an implementation of the Sun Microsystems’s JavaServer Pages 2.0 specification. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.

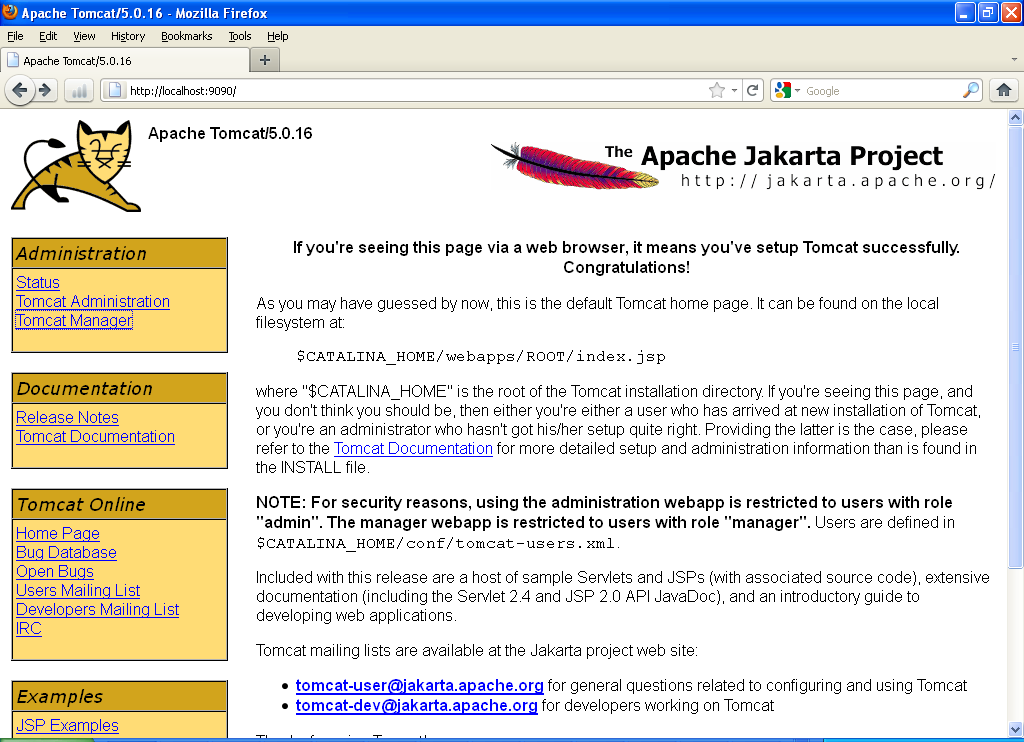
* **Jasper 2**

From Jasper to Jasper 2, important features were added:

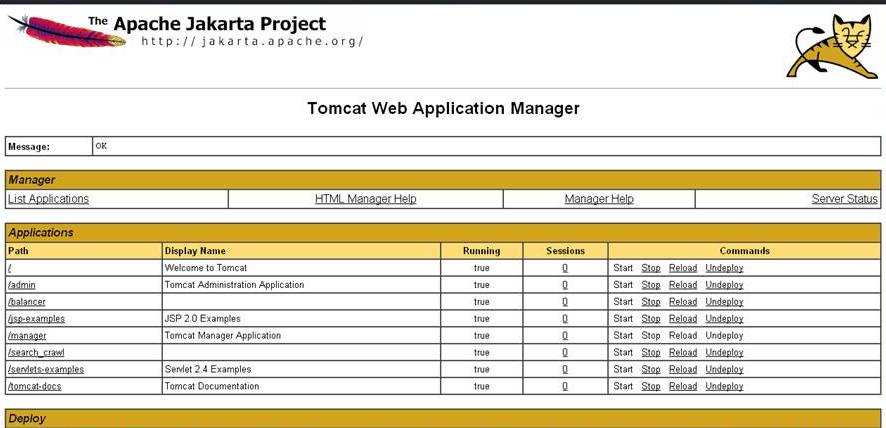
JSP Tag library pooling – Each tag markup in JSP file is handled by a tag handler class. Tag handler class objects can be pooled and reused in the whole JSP servlet.

Background JSP compilation – While recompiling modified JSP Java code, the older version is still available for server requests. The older JSP servlet is deleted once the new JSP servlet has finished being recompiled. Recompile JSP when included page changes – Pages can be inserted and included into a JSP at runtime. The JSP will not only be recompiled with JSP file changes but also with included page changes.

JDT Java compiler – Jasper 2 can use the Eclipse JDT (Java Development Tools) Java compiler instead of Ant and javac.

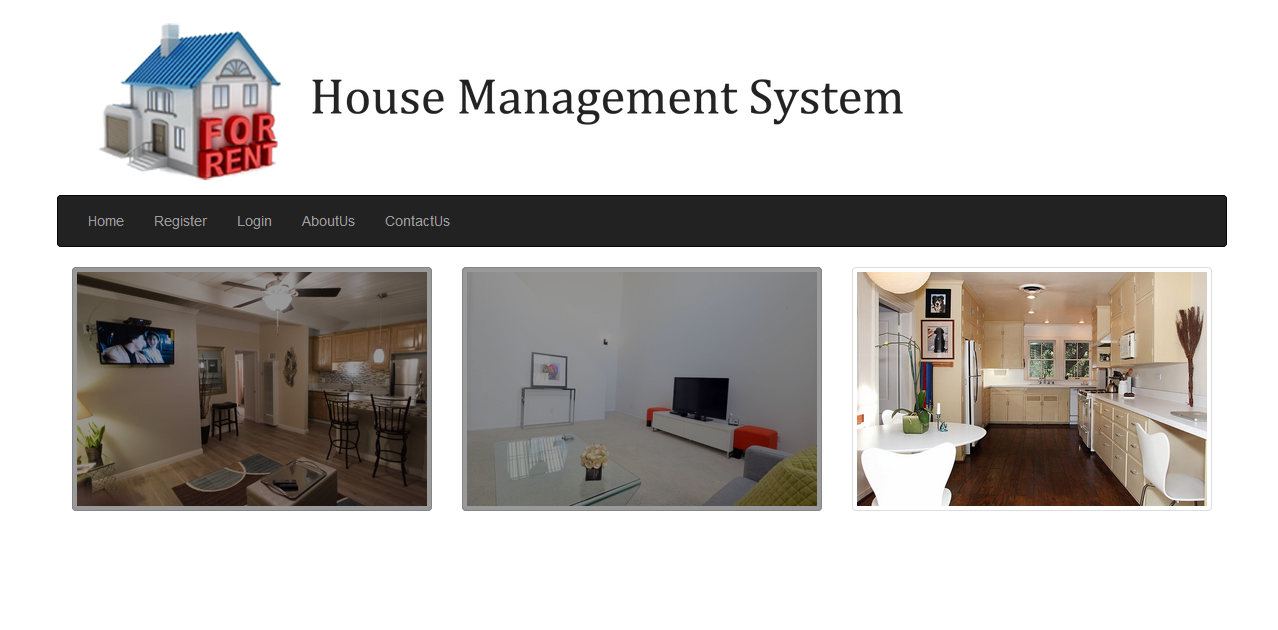


*Fig 2.5.5.1: Apache Tomcat Server*

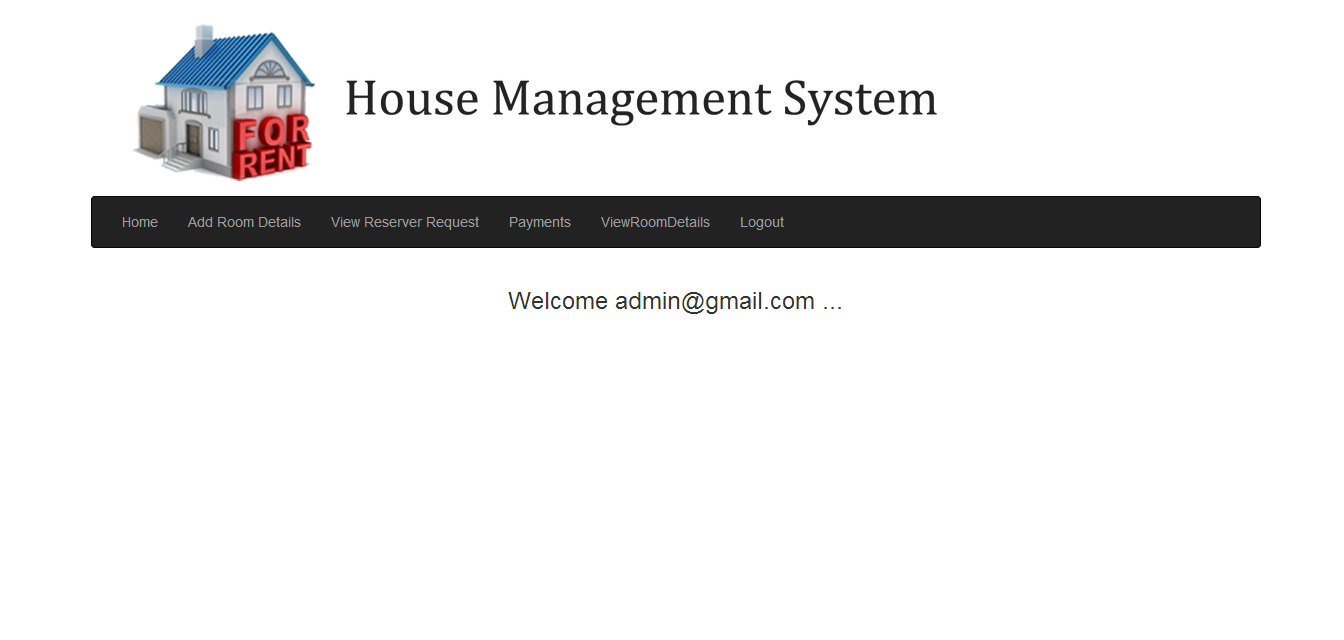
**

*Fig 2.5.5.2: Tomcat Application Manager*

**Screenshots:**



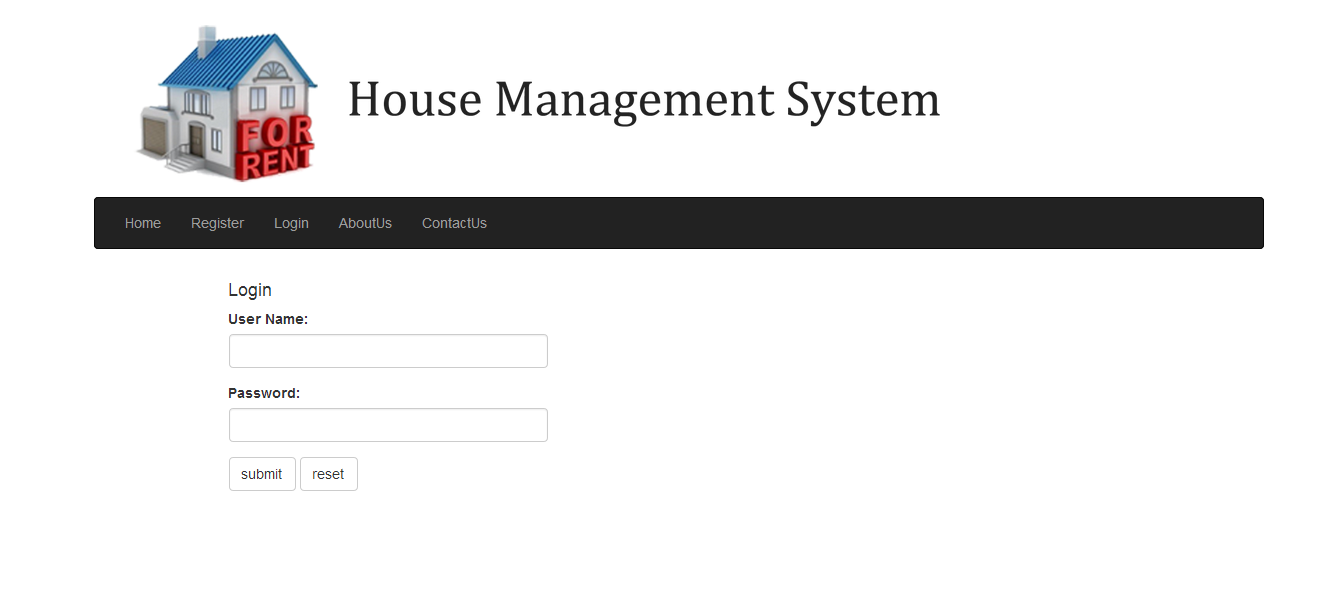
Home Page



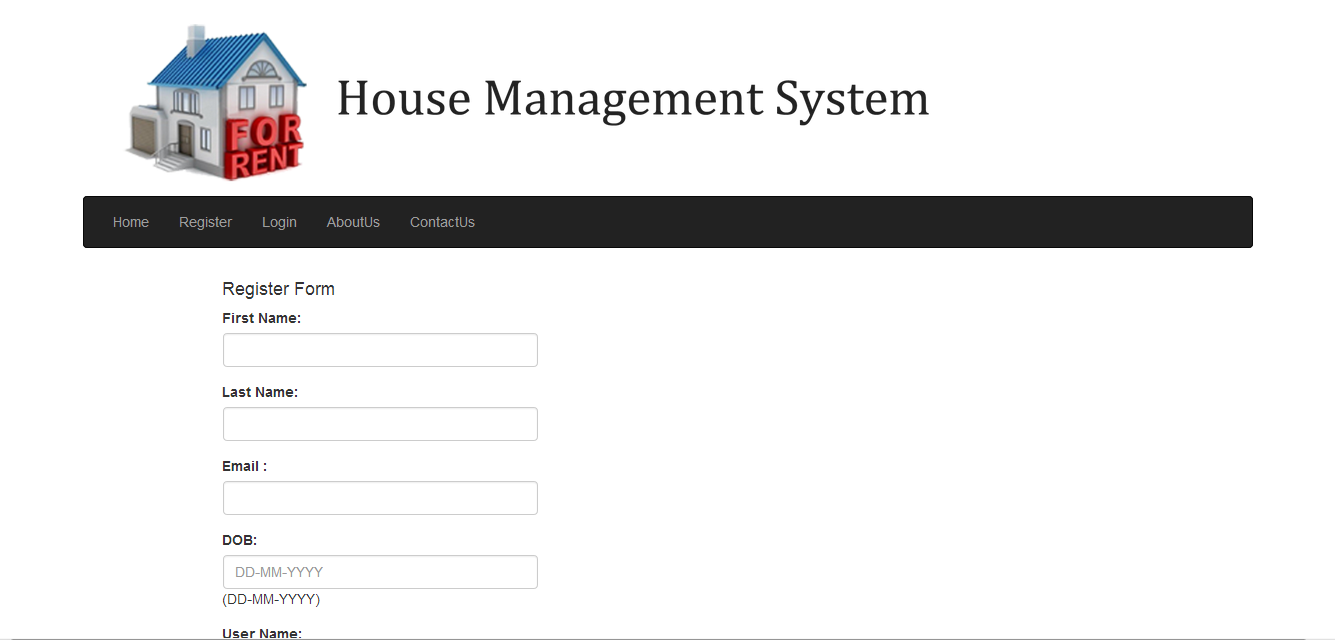
Admin Home



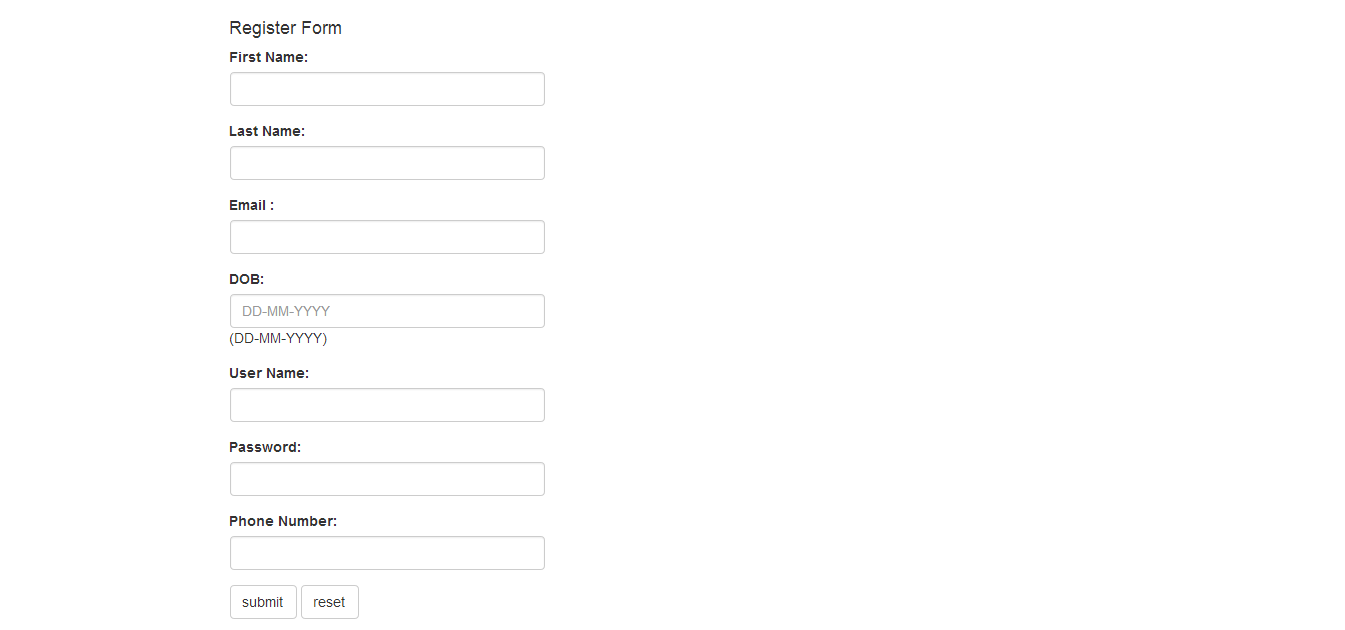
User Home



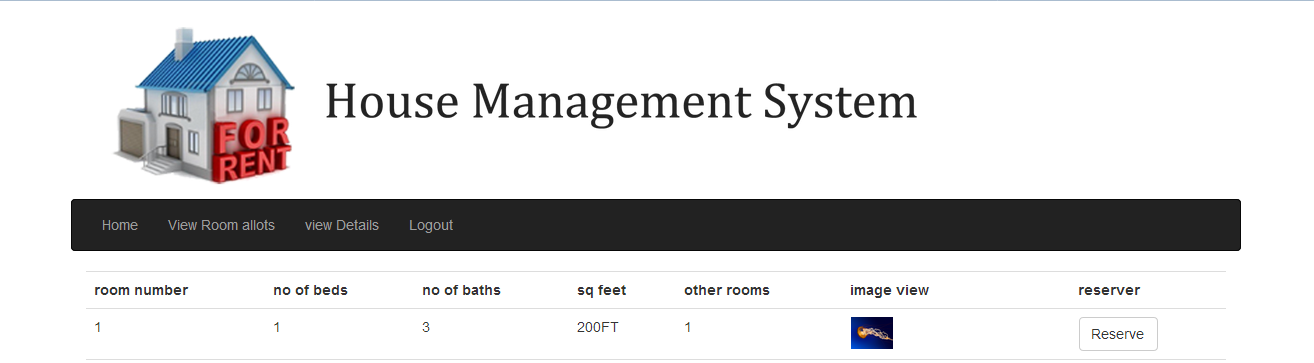
Login Page



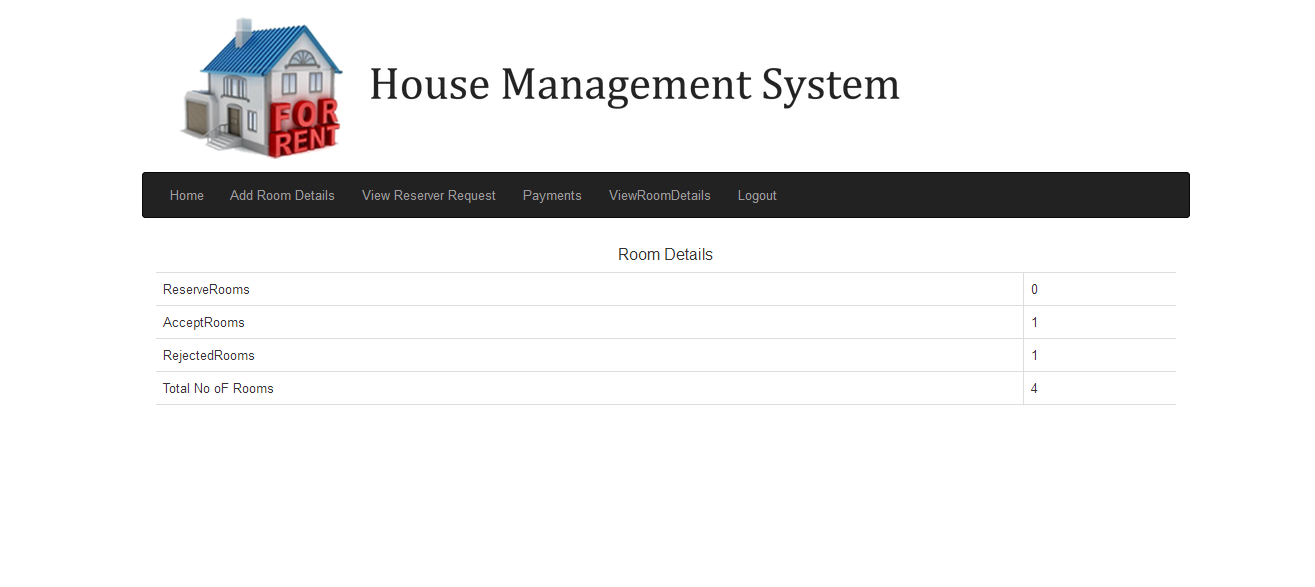
Register Page Head



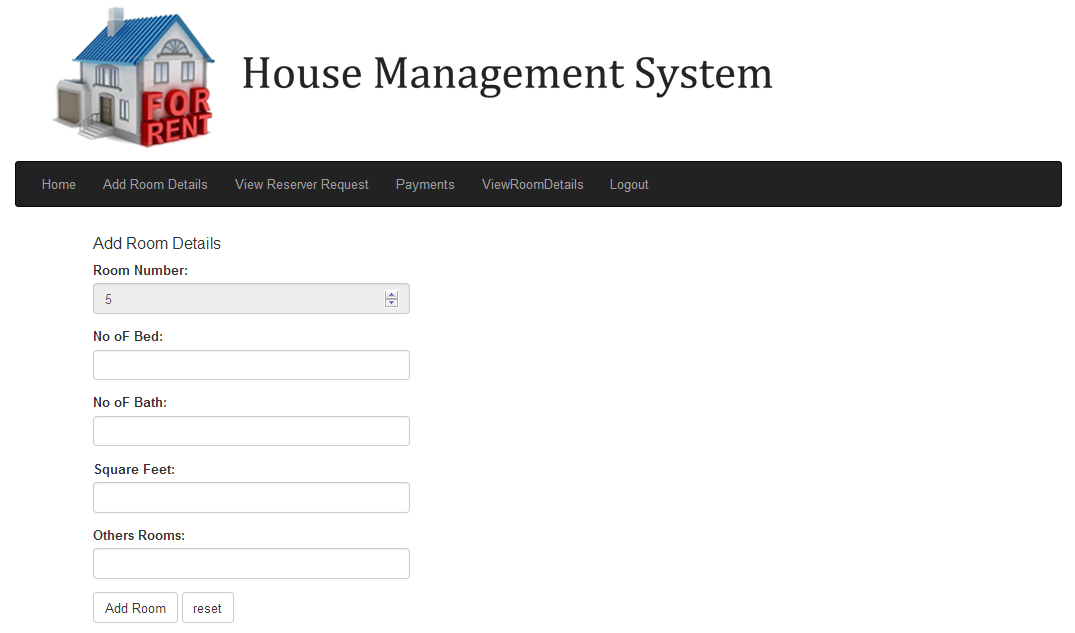
Register Page Cont..



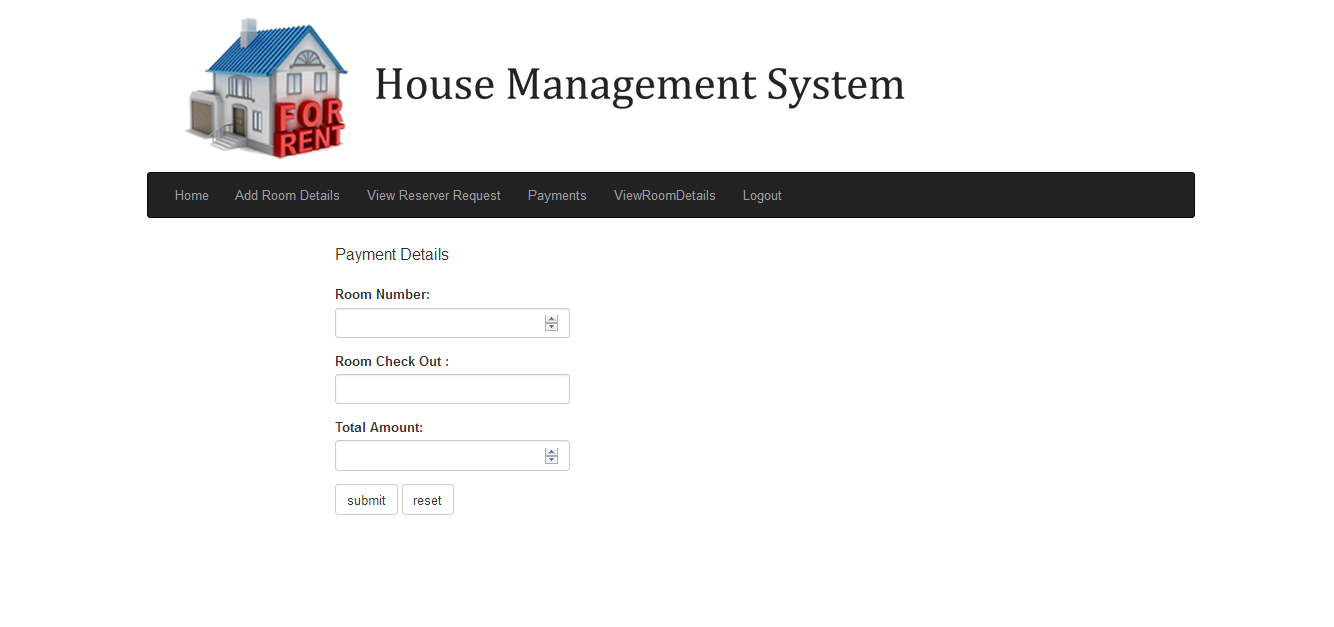
View Room Image



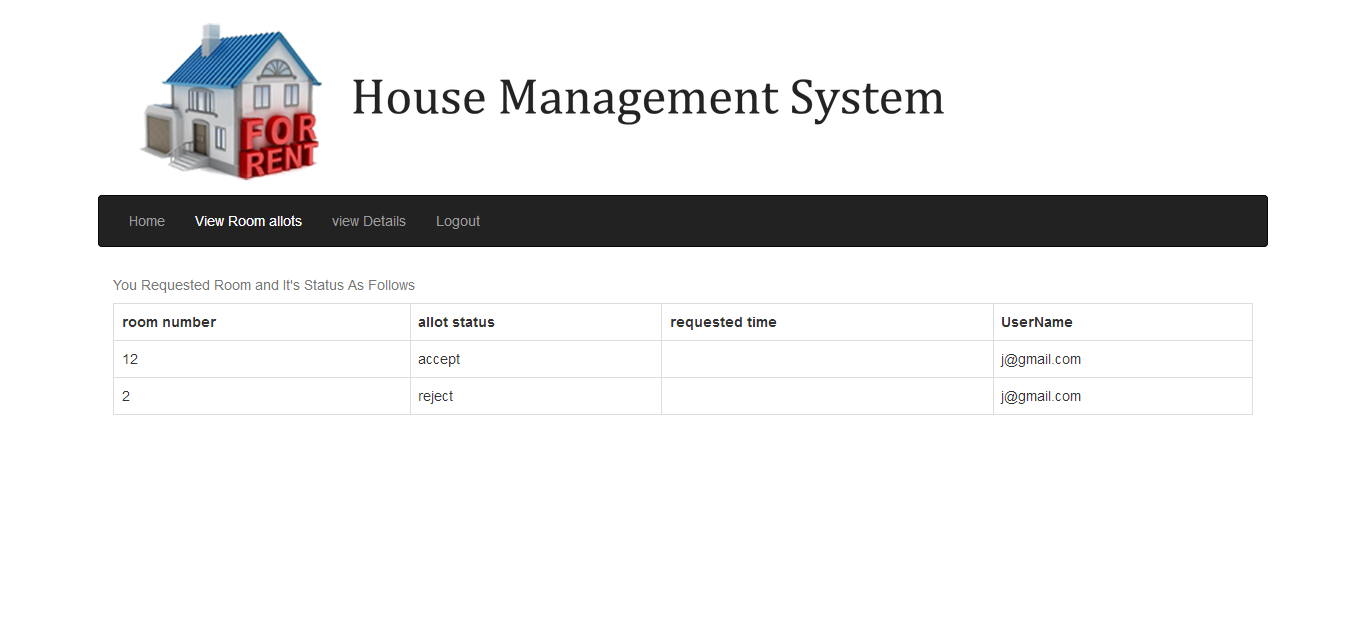
View Room Details



Add Room Details



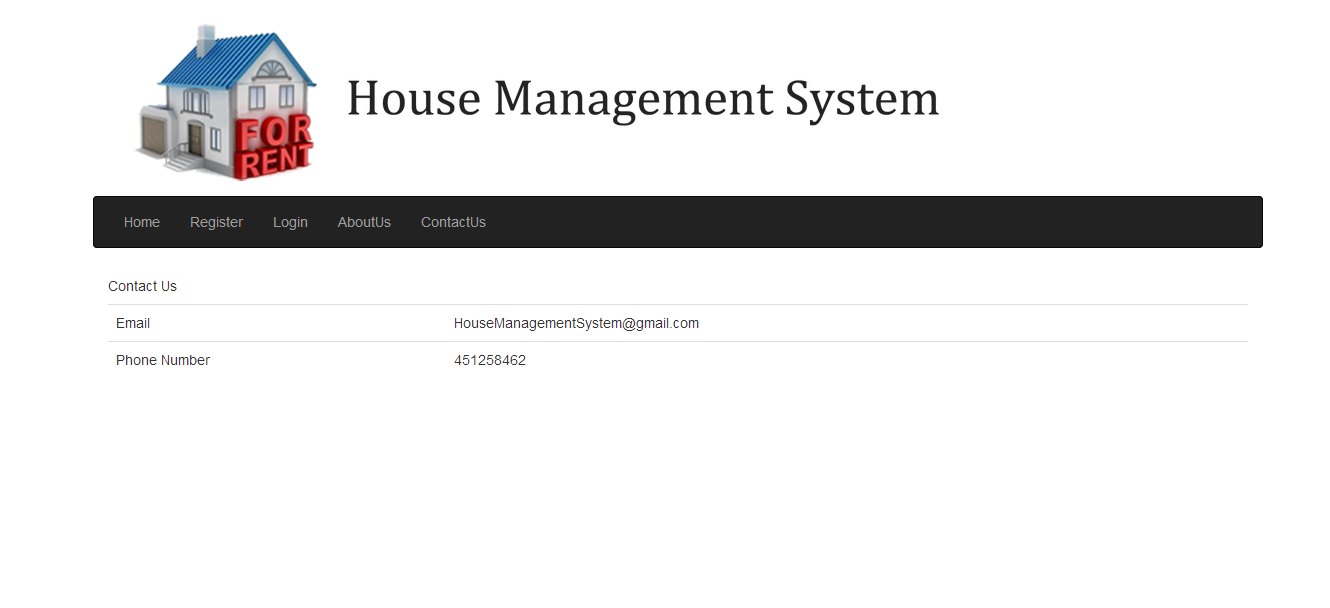
Payment Page



Room allot Details



Home Page



Contact us Page