## CHAPTER 1

**INTRODUCTION**

**1.1 OVERVIEW OF THEPROJECT**

This Web Application helps user to register individual home or apartment to assist you in finding the perfect rental home or property. Also, we can find your next rental from search view in your targeted area. This website is designed to attend to all our needs from buying property, selling property or renting/leasing of property in India. Here we found the better opportunity to invest our value of entire life. Property helps us to maintain the database of various property &agents’ information. It not only helps us to maintain the agent information but here we also allow agents to access the portal updated information across the global environment. We know it is a tiring to call individual property agents, arrange appointment, finding better time for appointment and they will assist you. For such complex process we provide a one simple online form which requires your basic information and we will assist in sort time period.

## CHAPTER 2

**SYSTEM ANALYSIS**

* 1. **EXISTINGSYSTEM**

The housing sector remains vigilant to face the challenges of the change of the existing system. People migrating to other cities or states either for different purposes. Finding a shelter, which fits all the requirements of the customer is hard, most of them don’t match their needs.

**DISADVANTAGES**

* Very hard to find the way for houses
* We need to provide more commission to search house.
* Everyone is not comfortable to follow this.

## PROPOSEDSYSTEM

This system is take care for sales the house via application, the admin can be access to upload the house details, once the uploaded with photo user can getting attractive and more happy to visit the house, the purpose of this software is an user friendly.

**ADVANTAGES**

* Every user can visit the page after the registration
* Once the process has been done admin directly reach you, you don’t need to collect the details.
* All the process has been handling into the single application.

## FEASIBILITYSTUDY

The feasibility of the system is analyzed in this phase and business proposal is put forth with general plan for the project and cost estimates. During the system analysis of the project, the feasibility study of proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in feasibility analysis are

* + - Technical Feasibility
    - Economic Feasibility
    - Operational Feasibility

## Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, whether the certain current resources and technology can be upgraded or added in the software to accomplish specified user requirements.

The technical requirements of the application are simple and basic. Python is used for the developers of the application and the framework is largely used by many, thus there will be enough support for future enhancements. The framework is stable and the support from the developers is constantly updated. The devices which have internet connectivity are enough for the application.

## Economic Feasibility

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on.

The cost of application development is very less and the cost of implementation is also less. It can be developed with the system with minimum requirements and can also be operated with the system with some basic requirements that are available the existing systems. For this, it is essential to consider expenses made on purchases and activities required to carry out software development.

## Operational Feasibility

Operational feasibility assesses the extent to which the required software performs a series of steps to solve user requirements. This feasibility is dependent on developer and involves visualizing whether the software will operate after it has been developed and be operative.

The application is developed based on the user requirements and is developed on the priority of the user requirements such as an integrated service and reviewing platform.

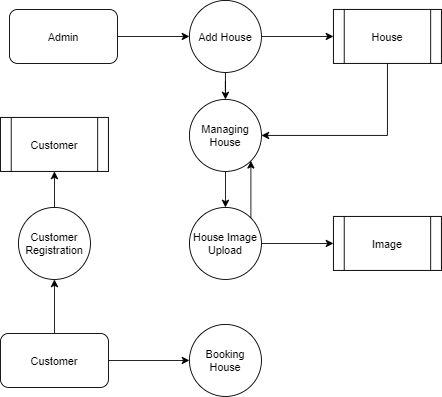
## PROBLEMANALYSIS

The orders from public are submitted through email, letter or form to the corresponding authority. The registered orders are forwarded to corresponding authority and report about the orders is submitted. The status of orders addressed is not recorded and maintained properly. The maps are not implemented. So, the customer does not know live location of the package. There is need for the software which receives the orders through online, forwarded to respective peoples for rectification and post the status of orders.

## CONTEXT AND DATA FLOWDIAGRAM

A data-flow diagram (DFD)is a way of representing a flow of a data of a process or system. The DFD also provides information about the outputs and inputs of each entity and process itself. A data-flow diagram is a part of structured-analysis modelling tools..





**SYSTEMCONFIGURATION**

### Hardware Requirements

Processor : P 4 700 GHz

RAM Capacity : 4GB

Hard Disk : 180GB

### Software Requirements

Operating System : Windows 8,10

Front End : JAVA

Back End : SQL

### Software Descriptions

### Java

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.

The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being stricter than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Unlike Windows executables (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE),

### MySQL

MySQL is the popular Open-Source Relational SQL Database Management System. MySQL is being used for developing various web-based software applications. The MySQL development project has made its [source code](https://en.wikipedia.org/wiki/Source_code) available under the terms of the [General Public](https://en.wikipedia.org/wiki/GNU_General_Public_License) [License.](https://en.wikipedia.org/wiki/GNU_General_Public_License) It is used to store the information.

MySQL was owned and sponsored by the single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

## CHAPTER 3

**SYSTEM DESIGN**

**3.1 INPUT DESIGN**

Input Design is the process of converting a user-oriented description of the input into a computer-based system. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. Input Design is the first phase in the system design. Input designing is to converting the user-oriented information to the computer- oriented form. The input data items are grouped and analyzed to find out whether the proposed system can be developed from the user input. The system is developed using various processes screens formats.

The main objective of input design is to

* + - It should serve specific purpose effectively such as storing, recording, and retrieving the information.
    - It ensures proper completion with accuracy.
    - It should be easy to fill and straightforward.
    - It should focus on user’s attention, consistency and simplicity.

### DATABASE DESIGN

The most important consideration in designing the database is how the information will be used. The main objective of designing a database is Data Integration, Data Integrity and Data Independence.

### Data Integration

In a database, information from several files is coordinated, accessed and operated upon as through it is in a single file. Logically, the information is centralized, physically, the data may be located on different devices, connected through data communication facilities.

### Data Integrity

Data integrity means storing all data in one place only and how each application access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy, that is data items need not be duplicated. A reduction in the direct access storage requirement.

### Data Independence

Dataindependenceistheinsulationofapplicationprogramsfromchangingaspects of physical data organization. This objective seeks to allow changes in the content and organizationofphysicaldatawithoutreprogrammingofapplicationandallowmodifications to application programs without reorganizing the physical data.

### 3.2.1 TABLE DESIGN

The table needed for each module were designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

**TABLE NAME: ADMIN**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Admin id | Int | 10 | Primary key |
| Username | Varchar | 30 | Not null |
| password | Varchar | 30 | Not null |

**TABLE NAME: HOUSE**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| house id | Int | 10 | Primary key |
| House name | Varchar | 20 | Not null |
| Owner name | Varchar | 20 | Not null |
| Contact number | Int | 10 | Not null |
| Address 1 | Varchar | 30 | Not null |
| Address 2 | Varchar | 30 | Not null |
| Area name | Varchar | 30 | Not null |
| City name | Varchar | 20 | Not null |
| District name | Varchar | 30 | Not null |
| State name | Varchar | 10 | Not null |
| Land mark | Varchar | 20 | Not null |
| Pin code | Varchar | 6 | Not null |
| Total square fit | Varchar | 5 | Not null |
| Bed room | Int | 5 | Not null |
| Hall | Int | 5 | Not null |
| Kitchen | Int | 5 | Not null |
| Others | Varchar | 20 | Not null |
| Price details | Int | 5 | Not null |

**TABLE NAME: IMAGES**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Image id | Int | 10 | Primary key |
| House id | Int | 10 | Foreign key |
| Image path | Varchar | 30 | Not null |
| Image name | Varchar | 20 | Not null |

**TABLE NAME: CUSTOMER**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Customer id | Int | 10 | Primary key |
| Customer name | Varchar | 20 | Not null |
| Mobile | Int | 10 | Not null |
| Email | Varchar | 20 | Not null |
| Username | Varchar | 20 | Not null |
| Password | Varchar | 20 | Not null |

**TABLE NAME: BOOKING**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Booking id | Int | 10 | Primary key |
| Customer id | Int | 10 | Foreign key |
| House id | Int | 10 | Foreign key |
| Booking status | Varchar | 20 | Not null |

* 1. **MODULEDESCRIPTION**

The main module in this project are listed below

* User Registration
* Add Houses
* Houses Details
* Booking House
* House Images Upload & download

1. **User Registration**

When the user needs to search home or anything, should need to register. Once the user has been registered with username and password, they can able to login.

1. **Add Houses**

The main role of admin for create the house and house details, also upload house images. Which details can be viewed in the user page, once the user liked can book it

1. **Houses Details**

The registered houses are showing in the home page, after login user can view the page, admin also can be modified the house details as well. Once the house has been booked, we don’t show to the other users.

1. **Booking House**

If user liked house, the user can book the house via this application, after the booking admin will get notify and reach them customer.

1. **House Images Upload & download**

We have good option to provide the house images and uploading in to the server, if the user needs the pics just, we gave download option also there in the user page screen

## CHAPTER 4

## SYSTEM TESTING

Testingisanintegralpartofanysystemdevelopmentlifecycle.Insufficient and untested applications may tend to crash and the result is loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation. Software testing can be looked upon as one among many processes, an organization performs, and that provides the lost opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors.

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software aretobetestedforthebestqualityassurancethatthesystemmeetsthespecificationand requirements for its intended use and performance. System testing is the most useful practical process of executing a program with the implicit intention of finding errors that make the program fails. System testing is done in three phases.

* + - * Unit Testing
      * Integration Testing
      * Validation Testing

### UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software the module. Using the detailed design and the process specification testing is done to registration by the user with in the boundary of the Login module. The login form receives the username and password details and validates the value with the database. If valid, the home page is displayed.

### INTEGRATION TESTING

Integration Testing is the process of this activity can be considered as testing the design and hence module interaction. The primary objective of integration testing is to discover errors in the interfaces between the components. Login form and registration form are integrated and tested together. If the user is newly registered, the received details will be stored in the registration table. While logging in, the application will check for valid user name and password in the registration table and if valid the user is prompted for submitting complaints.

### VALIDATION TESTING

Validation are independent procedures that are used together for checking that a product, service, or system meets [requirements](https://en.wikipedia.org/wiki/Requirement) and [specifications](https://en.wikipedia.org/wiki/Specification_(technical_standard)) and that it fulfills its in purpose the actual result from the expected result for the complaint process. Select the complaint category of the complaint by user. The input given to various forms fields are validated effectively. Each module is tested independently. It is tested that the complaint module fields receive the correct input for the necessary details such as complaint category, complaint id, reference name, complaint description, email for further process.

## CHAPTER 5

* 1. **CONCLUSION**

Effectively resolving the house issues is important to the buyer's long-term future, the Online Home Rental Project will be an important tool for creating rental housing stability by helping tenants speak with greater credibility through initiating and documenting communications and building productive relationships with sellers

## APPENDICES APPENDIX I SAMPLE CODE

package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dao.ApiDao;

import com.example.demo.response.GetCitizenResponse;

import com.example.demo.response.GetComplaintResponse;

import com.example.demo.service.ApiService;

@RestController

@RequestMapping(value = { "/api" })

public class ApiController {

@Autowired

ApiService service;

@Autowired

ApiDao dao;

@GetMapping("/login/{username}/{password}")

public String login(@PathVariable String username,@PathVariable String password) {

return dao.login(username,password);

}

@GetMapping("/add\_customer/{customer\_name}/{mobile}/{email}/{username}/{password}")

public String add\_customer(@PathVariable String customer\_name,

@PathVariable String mobile,

@PathVariable String email,

@PathVariable String username,

@PathVariable String password

) {

dao.add\_customer(customer\_name,mobile,email,username,password);

return "User Register Sucessfully";

}

@GetMapping("/add\_house/{name}/{owner}/{contact}/{address1}/{address2}/{city}/{district}/{state}/{landmark}/{pincode}/{squre}/{bedroom}/{hall}/{kitchen}/{others}/{price}")

public String add\_car(

@PathVariable String name,

@PathVariable String owner,

@PathVariable String contact,

@PathVariable String address1,

@PathVariable String address2,

@PathVariable String city,

@PathVariable String district,

@PathVariable String state,

@PathVariable String landmark,

@PathVariable String pincode,

@PathVariable String squre,

@PathVariable String bedroom,

@PathVariable String hall,

@PathVariable String kitchen,

@PathVariable String others,

@PathVariable String price) {

dao.add\_house(name,owner,contact,address1,address2,city,district,state,landmark,pincode,squre,bedroom,hall,kitchen,others,price);

return "House Register Sucessfully";

}

@GetMapping("/add\_booking/{customer\_id}/{houseid}")

public String add\_car(@PathVariable Integer customer\_id,

@PathVariable Integer houseid

) {

dao.add\_book(customer\_id,houseid);

return "House Register Sucessfully";

}

@GetMapping("/get\_houses")

public List<Object[]> get\_houses() {

return dao.get\_houses();

}

@GetMapping("/get\_bookings")

public List<Object[]> get\_bookings() {

return dao.get\_bookings();

}

}package com.example.demo.configuration;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableTransactionManagement

public class HibernateConfiguration {

@Value("${db.driver}")

private String DB\_DRIVER;

@Value("${db.password}")

private String DB\_PASSWORD;

@Value("${db.url}")

private String DB\_URL;

@Value("${db.username}")

private String DB\_USERNAME;

@Value("${hibernate.dialect}")

private String HIBERNATE\_DIALECT;

@Value("${hibernate.show\_sql}")

private String HIBERNATE\_SHOW\_SQL;

// @Value("${hibernate.hbm2ddl.auto}")

private String HIBERNATE\_HBM2DDL\_AUTO;

@Value("${entitymanager.packagesToScan}")

private String ENTITYMANAGER\_PACKAGES\_TO\_SCAN;

@Bean

public LocalSessionFactoryBean sessionFactory() {

LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(ENTITYMANAGER\_PACKAGES\_TO\_SCAN);

Properties hibernateProperties = new Properties();

hibernateProperties.put("hibernate.dialect", HIBERNATE\_DIALECT);

hibernateProperties.put("hibernate.show\_sql", HIBERNATE\_SHOW\_SQL);

// hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE\_HBM2DDL\_AUTO);

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public DataSource dataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(DB\_DRIVER);

dataSource.setUrl(DB\_URL);

dataSource.setUsername(DB\_USERNAME);

dataSource.setPassword(DB\_PASSWORD);

return dataSource;

}

@Bean

public HibernateTransactionManager transactionManager() {

HibernateTransactionManager txManager = new HibernateTransactionManager();

txManager.setSessionFactory(sessionFactory().getObject());

return txManager;

}

}package com.example.demo.configuration;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableTransactionManagement

public class HibernateConfiguration {

@Value("${db.driver}")

private String DB\_DRIVER;

@Value("${db.password}")

private String DB\_PASSWORD;

@Value("${db.url}")

private String DB\_URL;

@Value("${db.username}")

private String DB\_USERNAME;

@Value("${hibernate.dialect}")

private String HIBERNATE\_DIALECT;

@Value("${hibernate.show\_sql}")

private String HIBERNATE\_SHOW\_SQL;

// @Value("${hibernate.hbm2ddl.auto}")

private String HIBERNATE\_HBM2DDL\_AUTO;

@Value("${entitymanager.packagesToScan}")

private String ENTITYMANAGER\_PACKAGES\_TO\_SCAN;

@Bean

public LocalSessionFactoryBean sessionFactory() {

LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(ENTITYMANAGER\_PACKAGES\_TO\_SCAN);

Properties hibernateProperties = new Properties();

hibernateProperties.put("hibernate.dialect", HIBERNATE\_DIALECT);

hibernateProperties.put("hibernate.show\_sql", HIBERNATE\_SHOW\_SQL);

// hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE\_HBM2DDL\_AUTO);

sessionFactory.setHibernateProperties(hibernateProperties);

return sessionFactory;

}

@Bean

public DataSource dataSource() {

DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(DB\_DRIVER);

dataSource.setUrl(DB\_URL);

dataSource.setUsername(DB\_USERNAME);

dataSource.setPassword(DB\_PASSWORD);

return dataSource;

}

@Bean

public HibernateTransactionManager transactionManager() {

HibernateTransactionManager txManager = new HibernateTransactionManager();

txManager.setSessionFactory(sessionFactory().getObject());

return txManager;

}

}package com.example.demo.dao;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import javax.transaction.Transactional;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.query.NativeQuery;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Repository;

@Repository

@Transactional

public class ApiDao {

@Autowired

SessionFactory sf;

public String login(String username, String password) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select \* from admin where username='"+username+"' and password='"+password+"'";;

NativeQuery nq = session.createNativeQuery(sql);

if (nq.list().size() != 0) {

return "admin";

} else {

String sql1 = "select \* from student where username='"+username+"' and password='"+password+"'";;

NativeQuery nq1 = session.createNativeQuery(sql1);

if (nq1.list().size() != 0) {

List<Object[]> a = nq1.list();

return "id="+a.get(0)[0];

}else {

return "Invalid";

}

}

}

public void add\_customer(String name, String mobile, String email, String username, String password) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `user` (`id`, `name`, `mobile`, `email`, `username`, `password`) VALUES"

+ " (NULL, '"+name+"', '"+mobile+"', '"+email+"', '"+username+"', '"+password+"');";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_house(String name, String owner, String contact, String address1, String address2, String city,

String district, String state, String landmark, String pincode, String squre, String bedroom, String hall,

String kitchen, String others, String price) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `house` (`id`, `name`, `owner`, `contact`, `address1`, `address2`, `city`, `district`, `state`, `landmark`, `pincode`, `squre`, `bedroom`, `hall`, `kitchen`, `others`, `price`) VALUES "

+ "(NULL, '"+name+"', '"+owner+"', '"+contact+"', '"+address1+"', '"+address2+"', '"+city+"', "

+ "'"+district+"', '"+state+"', '"+landmark+"', '"+pincode+"', '"+squre+"', '"+bedroom+"', '"+hall+"', "

+ "'"+kitchen+"', '"+others+"', '"+price+"');";

session.createSQLQuery(sql).executeUpdate();

}

public void add\_book(Integer customer\_id, Integer houseid) {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "INSERT INTO `booking` (`id`, `userid`, `houseid`) VALUES "

+ "(NULL, "+customer\_id+", "+houseid+");";

session.createSQLQuery(sql).executeUpdate();

}

public List<Object[]> get\_houses() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "select \* from house";

NativeQuery nq = session.createNativeQuery(sql);

return nq.list();

}

public List<Object[]> get\_bookings() {

// TODO Auto-generated method stub

Session session = sf.getCurrentSession();

String sql = "SELECT user.name,user.mobile,house.name as housename,house.owner,house.contact FROM `booking` LEFT JOIN house on(house.id=booking.houseid) LEFT JOIN user on(user.id=booking.userid)";

NativeQuery nq = session.createNativeQuery(sql);

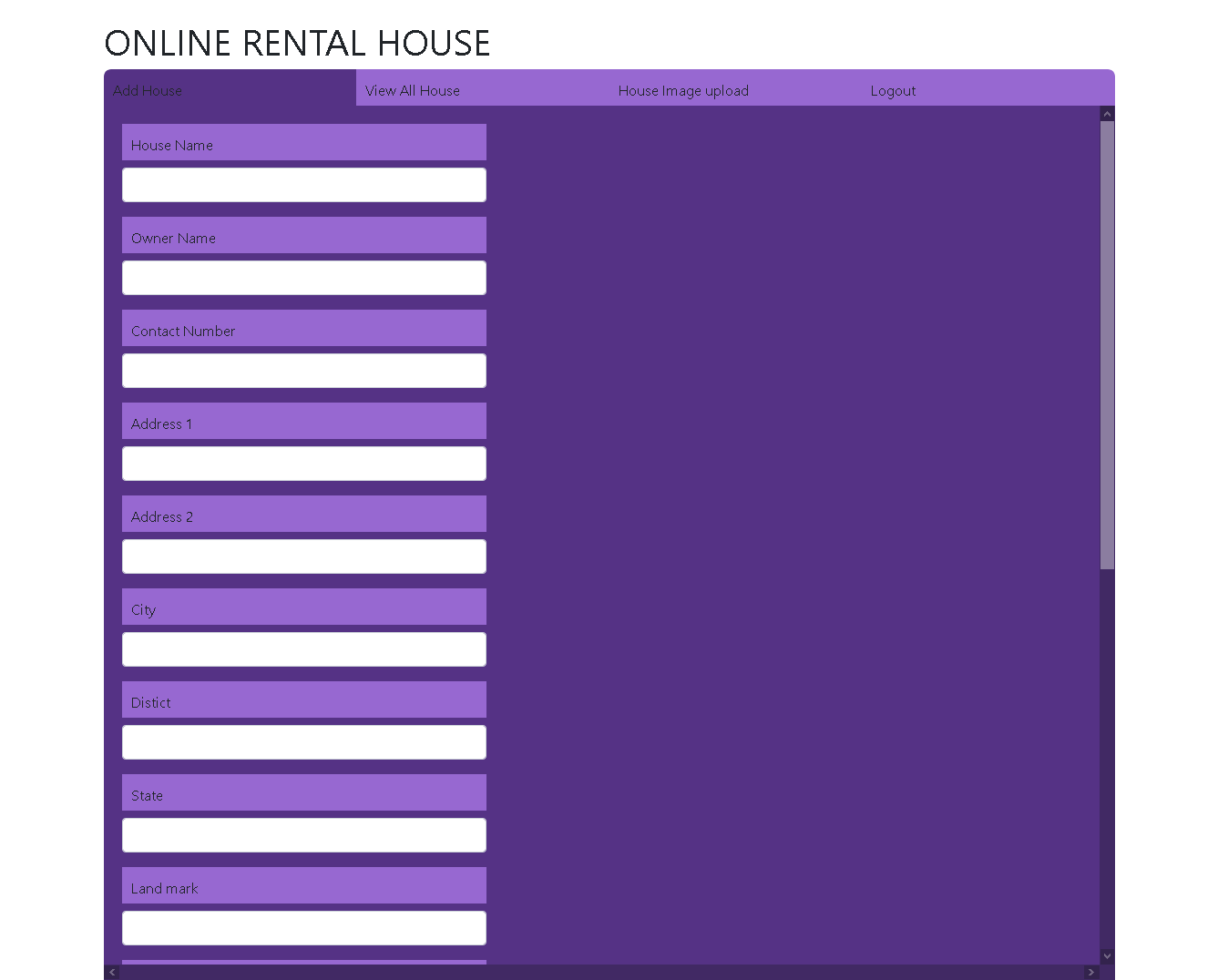
return nq.list();

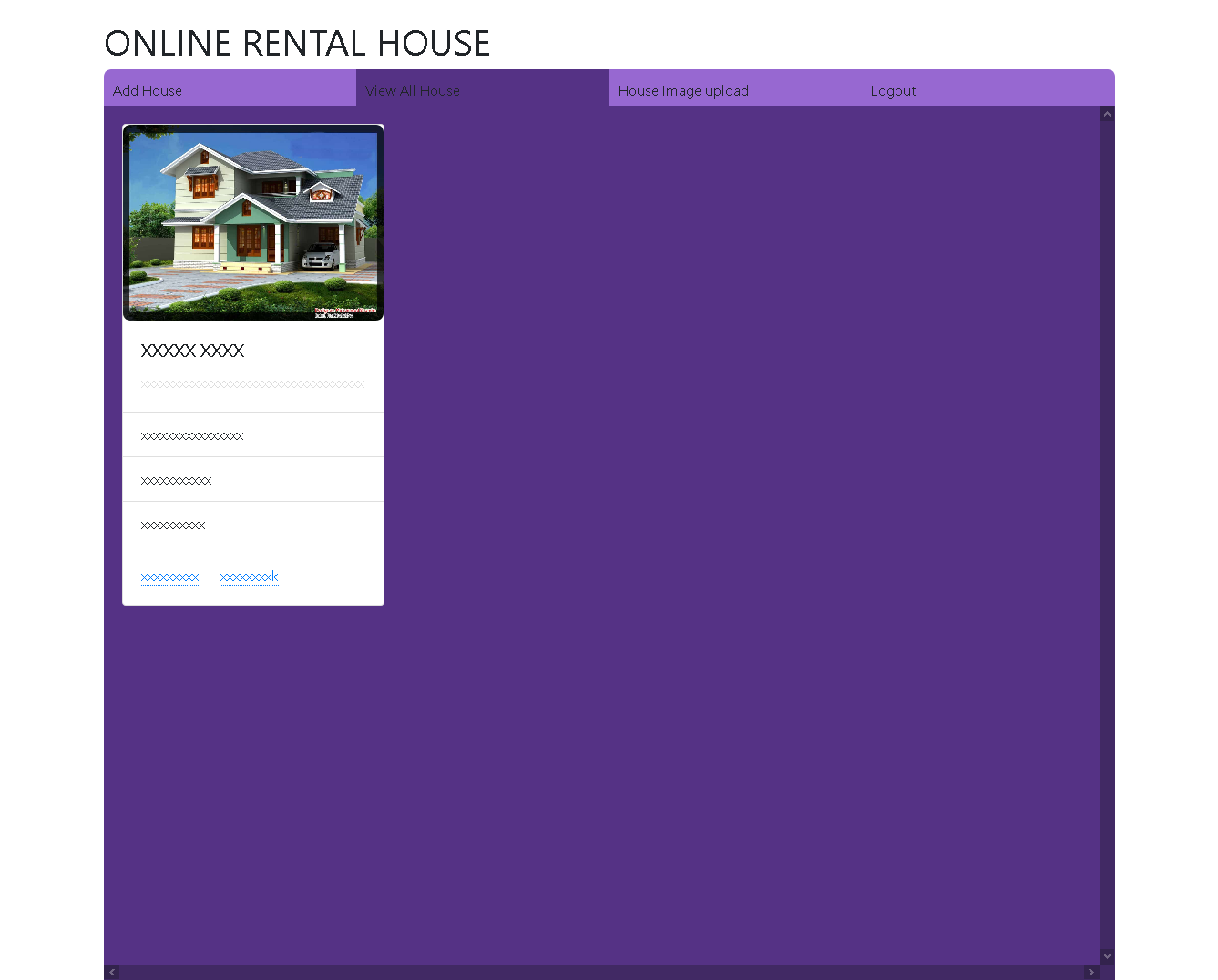
}

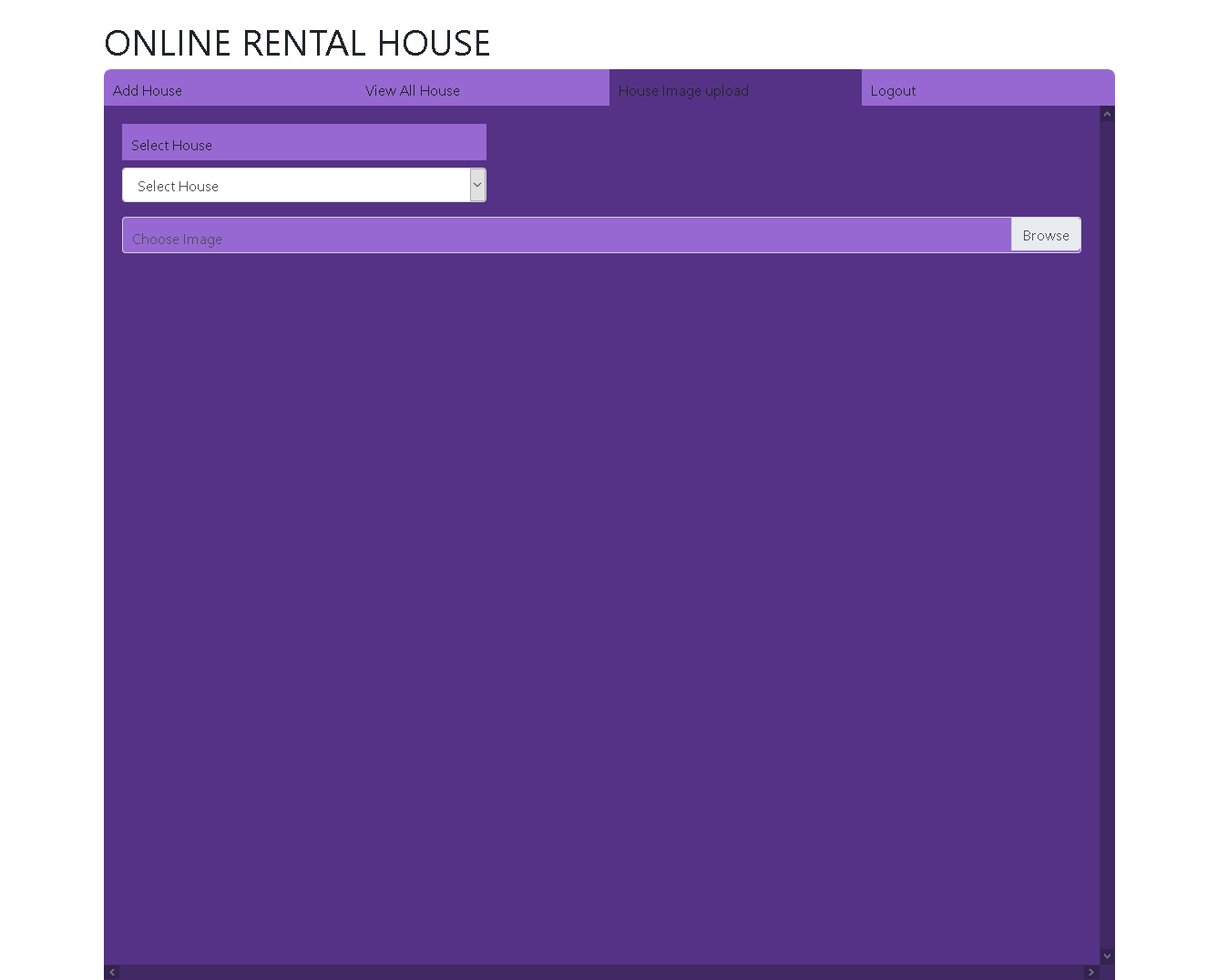
}

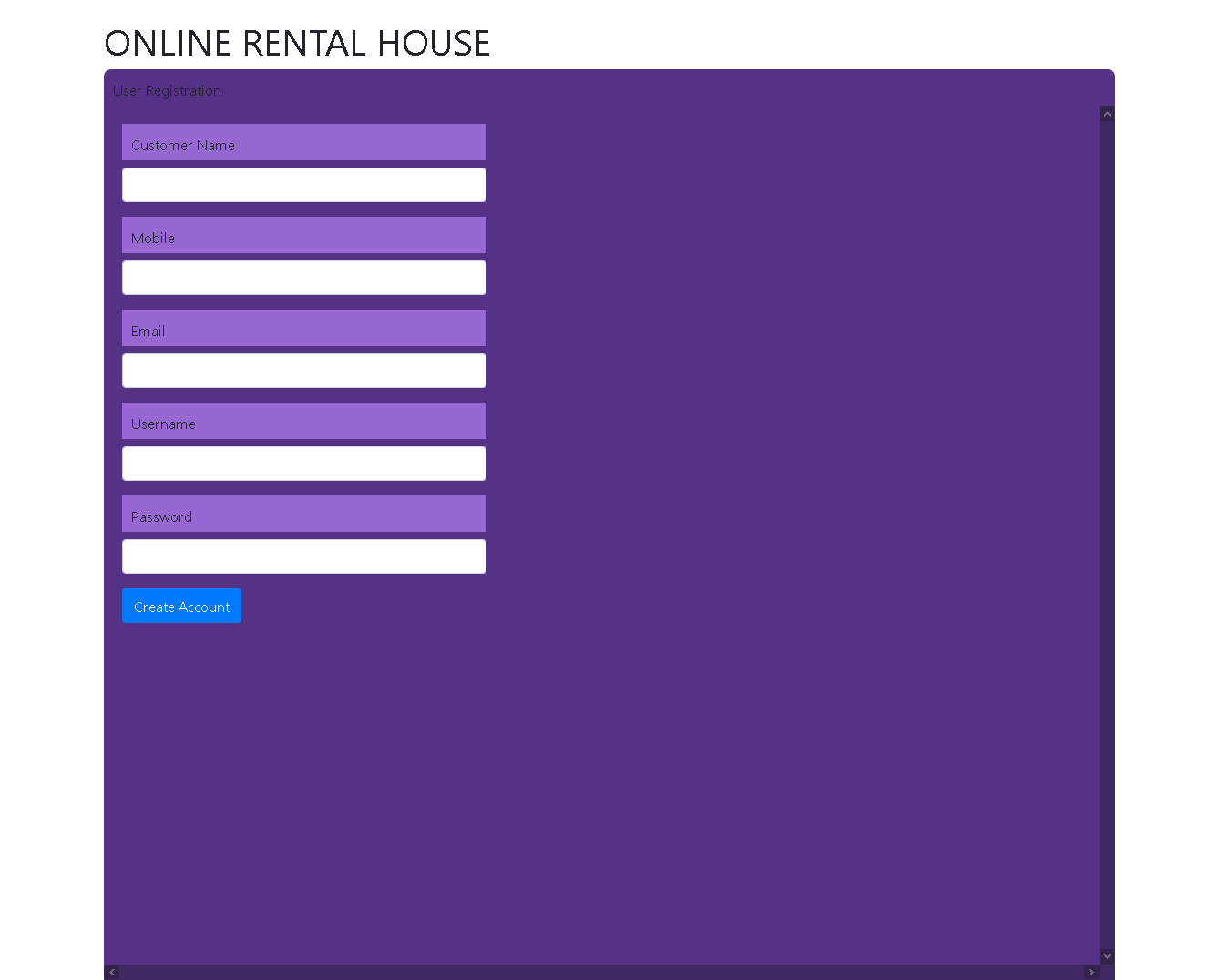
## APPENDIX II

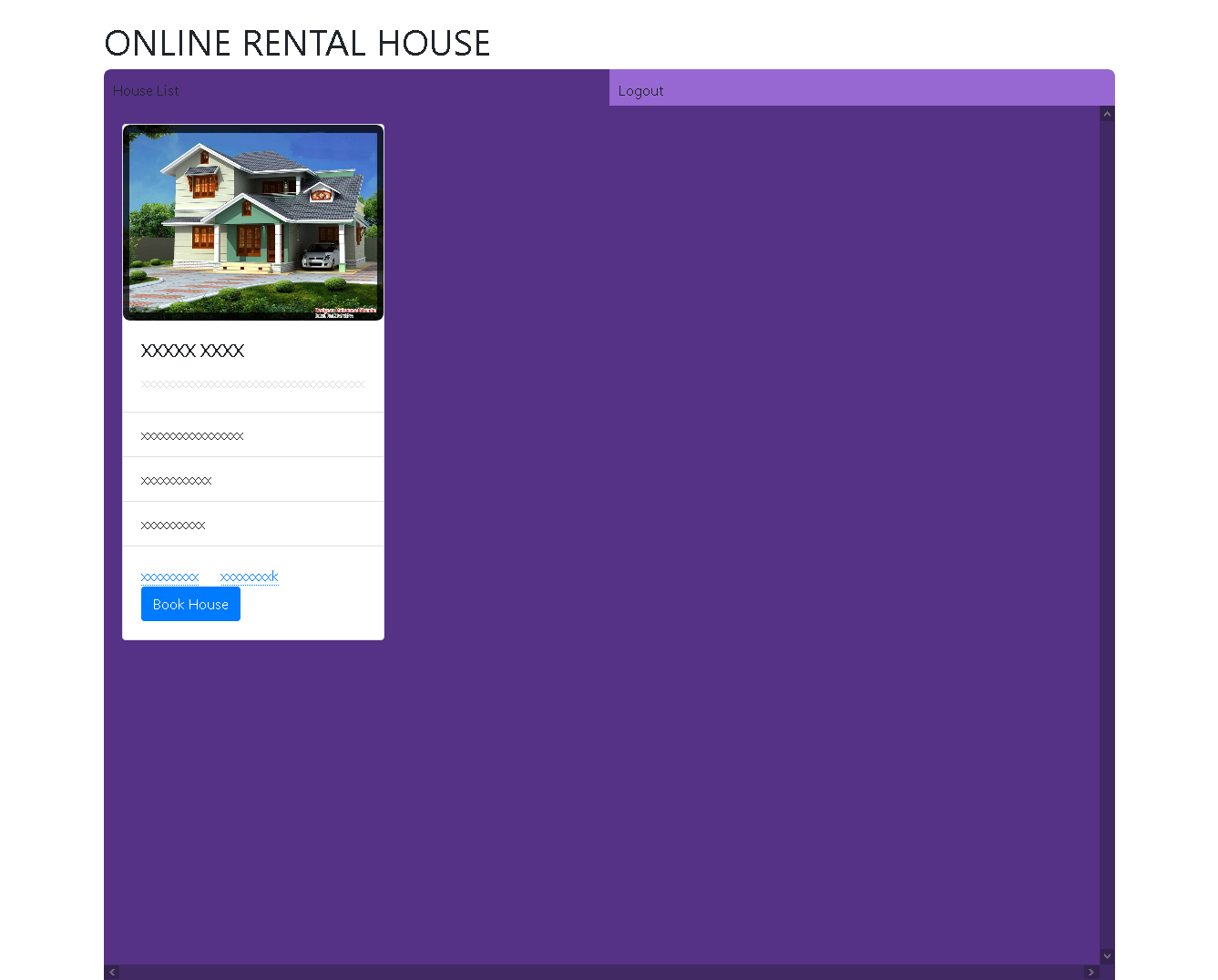
**SAMPLE SCREENSHOTS**











**REFERENCES**

**BOOK REFERENCES**

1. Jeff Forcier, Paul Bissex, Wesley J. Chun, (2018) “Python Web development with Django (Developer’s Library)”, WesleyProfessional.
2. Roger S. Pressman, (2017) “Software Engineering”, Tata McGraw-Hill –Europe.

## WEB REFERENCES

1. https://[www.javaprogramming.net](http://www.javaprogramming.net)
2. https://[www.w3schools.com](http://www.w3schools.com/)
3. https://[www.tutorialspoint.com](http://www.tutorialspoint.com/)