SSN COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION)

KALAVAKKAM

DEPARTMENT OF INFORMATION TECHNOLOGY

UIT1511 – SOFTWARE DESIGN LAB

(REGULATION 2018)

Estate Management System

for Lalitha Apartments

By

V.Meganathan -195002072

Nandini R -195002075

Navaneetha Krishnan S -195002075

Alagappan S -195002301

JULY'21 – December'21

UIT1511 - Software Design Lab
Final Project Report

-Team Pigeons

Problem Description:

An interactive interface, through which tenant, transactional, maintenance data about apartments, can be stored in an organized and efficient manner. The stored data can also be used to generate statistical reports, to analyse financial data regarding the apartment.

Statement of Work:

Software Requirements Specification
for

Estate Management System(REMS)

Version 1.5 approved

Prepared by V.Meganathan

Nandini R

Navaneetha Krishnan S

Alagappan S

IT Department, SSN College of Engineering

29/10/2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Versions
Nandini R & V.Meganathan	22-08-2021	● Designed and added the layout for UI ● Updated References tab ● Added user class characteristics ● Updated Software tools and design constraints and dependencies	1.2

		• Updated Business rules	
Nandini R & V.Meganathan	5-10-2021	Updated System Features	1.3
V.Meganathan	29-10-2021	Updated and enforced document convention, Added references, Updated Software used in development	1.4
V.Meganathan	10-12-2021		1.5

Contributor List

Section Number	Contributor Name
1	Meganathan
2	Nandini & Alagappan
3	Nandini & Alagappan & Naveen
4	Meganathan & Naveen
5	Client & Meganathan
6	Meganathan

Introduction

Purpose

An interactive interface, through which housing data ,finances, and maintenance records of apartments, can be stored in an organized and efficient manner. The stored data can also be used to generate statistical reports, to analyse financial data regarding the apartment.

Document Conventions

Heading Font : Times New Roman, Size 14, Bold applied, Color Black.
Sub-Heading Font : Times New Roman, Size 13, Bold applied, Color Black.
Smaller Sub-Heading Font: Times New Roman, Size 12, Bold applied, Color Black
Content Font : Times New Roman, Size 12, Color Black

(For Section 4) Each feature will have the priority specified in its description.
Hyperlinks font: Times New Roman, Size 12, Bold applied, Italics applied, Color Blue.

Intended Audience and Reading Suggestions

It is assumed that the intended audience has sufficient technical and domain knowledge of the software tools and the project at hand. It is not mandatory, but will help in understanding the tasks in their full gravity.

- It is directed towards
  - Client and Employees at Real Estate Enterprise
  - Development Team
  - Testers

- It is also recommended that certain readers focus on particular sections of the document based on their relevance to them.

- Client and enterprise employees are recommended to focus on Section 1 and Section 5, which handles the product aim and the non-functional requirements stated by them.

- Product Owner is encouraged to place more attention on the requirements section.

- For the Testers and development team it is suggested that they focus on sections 2,3 and 4 which handles the technical aspects of the project implementation.

- Scrum Master ideally is to have a good grasp on the entire document.

Product Scope

Employees of Real Estate enterprises directly benefit from this product, as it would improve their data recording, storage and analysis processes by making it easier and simpler to use. .

It aims to use the Relational Database model to make the most of resources while ensuring efficient data storage and simple access.

This product is completely scalable regardless of the amount of property involved.

References

UML Class Diagram Tutorial
UML Use Case Diagram Tutorial
How to Make a UML Sequence Diagram
The Complete 2021 Web Development Bootcamp
Difference Between HLD and LLD - javatpoint

UML - Class Diagram
UML Component Diagram
The Flask Mega-Tutorial Part I: Hello, World! (till chapter 5)
Git Tutorial for Beginners: Learn Git in 1 Hour
Git and GitHub for Beginners - Crash Course
Git Branches Tutorial
Creating a Dynamic Select Field With Flask-WTF and JavaScript
Welcome to Flask — Flask Documentation (2.0.x) (ad-hoc basis)

Overall Description

Product Perspective

The product is a self-contained venture, developed with the objective of making the processes of data recording, storage and analysis in the Real Estate business easier. It aims to make the most use of resources while giving out an unchallenging, palpable interactive experience.

Product Functions

The major functions of the product:

It is aimed at making the data recording process hassle free, for Real Estate related data. It is meant to substitute and improve upon the current practice of manually recording data in separate Spreadsheets.

It also ensures Data consistency, by enforcing constraints, before the entered data is submitted to the repository. This makes sure that there is no redundant or conflicting data present.

The product would mainly make strides, in analysis and review of the stored data. Past and current methods involve manually picking out the required data from spreadsheets and consolidating , cleaning, and processing the data before being able to submit a report. The product would do all of this in mere clicks of the cursor.

The product would also double down as an advertisement to the client's apartment, attracting potential tenants.

User Classes and Characteristics

The main user classes of this product would be

Customers: This product would provide a great platform for the enterprise concerned to advertise their apartments. This would be beneficial to customers, as they would be able to get ample information about the apartments like vacant homes, safety features, facilities and rules.
Employees: The employees of the enterprise would be the primary beneficiaries of this product, as it directly addresses their daily working difficulties with data recording and retrieval. With this product, the entire process would be simple and easy to use, and any complexity involved would be abstracted.

Operating Environment

Software Components to be used are (the right part of "==" denotes version)

HTML==5
CSS
JavaScript
Node.js & Express.js
alembic==1.6.5
Flask==2.0.1
Flask-Login==0.5.0
Flask-Migrate==3.1.0
Flask-SQLAlchemy==2.5.1
Flask-WTF==0.15.1
Jinja2==3.0.1
Mako==1.1.5
MarkupSafe==2.0.1
python-dateutil==2.8.2
python-dotenv==0.19.0
python-editor==1.0.4
SQLAlchemy==1.4.23
Werkzeug==2.0.1
WTForms==2.3.3

Design and Implementation Constraints

Data prior to the realisation of the product would need to be uploaded into the database to make complete use of this project. The cleansing, normalization and consisting checking of that data would need to be manually done.
Since data is manually entered, there is a potential for human error in the data recording process.
The business rules of the apartments are subject to regular change.
Travel and health restrictions due to pandemic.

User Documentation

The documentation planned are
User manual
ER diagram of the Relational Model used

Assumptions and Dependencies

Assumptions

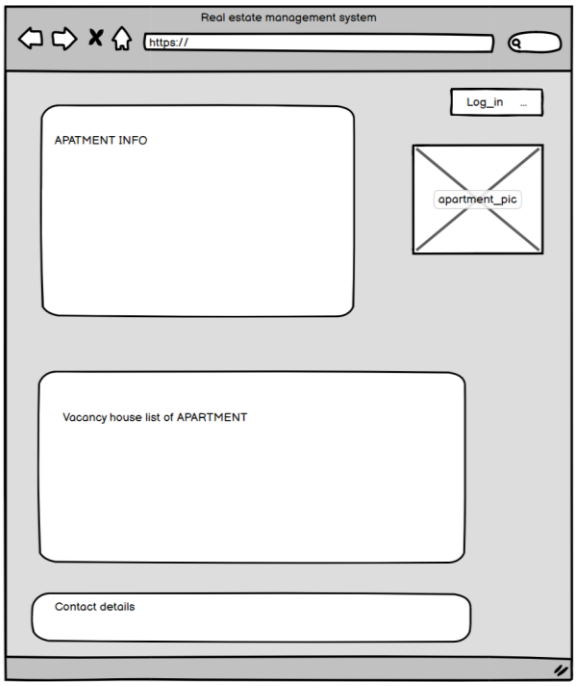
The users are familiar with the attributes they would be handling.
The apartment homes involved are all rented.

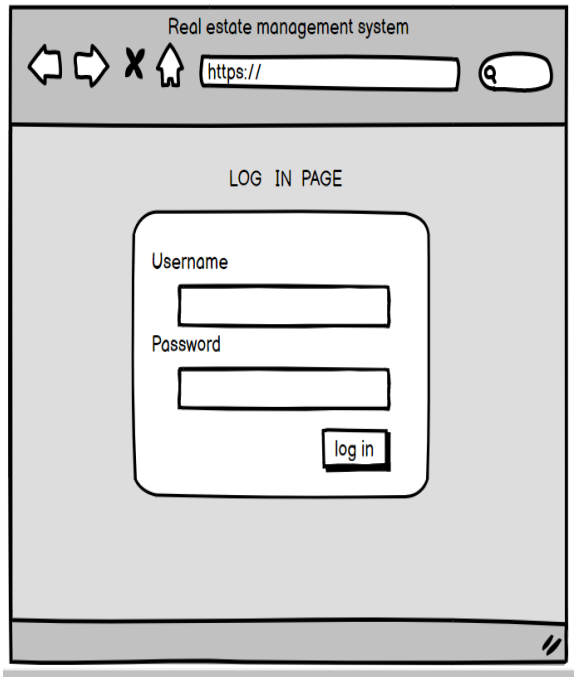
Dependencies

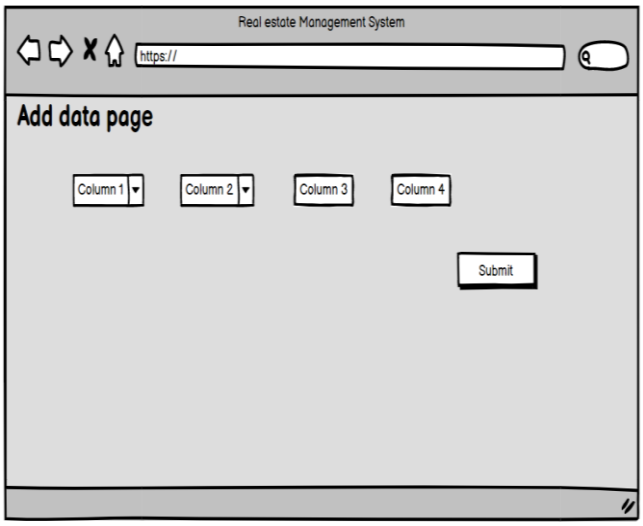
Internet connection is mandatory to use the product.
Entering wrong values of data will give incorrect analysis. Hence data being entered into the database is subjective to human error.

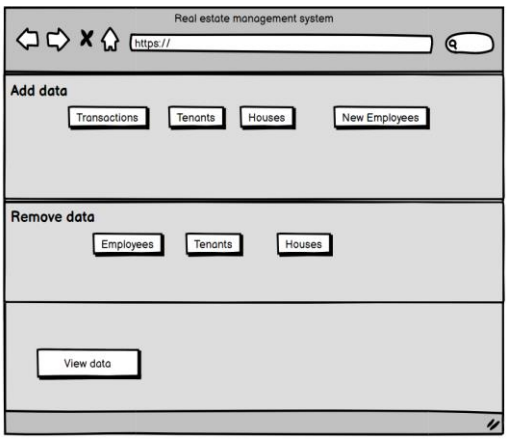
External Interface Requirements

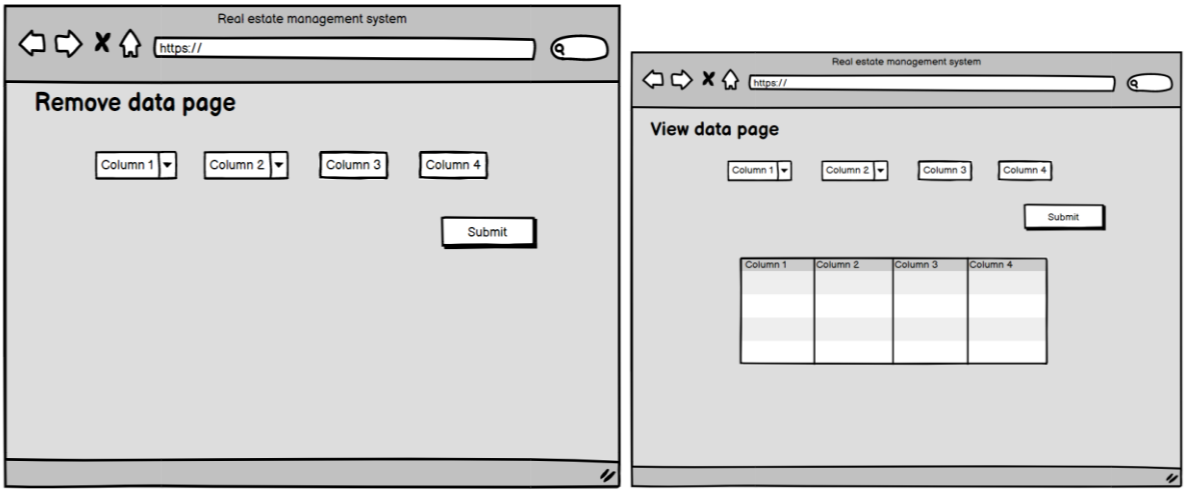
User Interfaces











Hardware Interfaces

-

Software Interfaces

HTML - used to develop the structure of the website
CSS & Bootstrap- for styling the website
JavaScript - allows the website to have dynamic behaviour
Node.js & Express.js - both allow the JS newfound backend functionality.
Pycharm Professional 2021.2.2- IDE for Flask Server Scripting and Jinja Template Scripting
Terminal Program- to interact with the SQLAlchemy Database.

Communications Interfaces

-

System Features

Vacancy List

4.1.1 Description and Priority

High Priority: Records of currently vacant homes, along with their details are to be displayed. Further details regarding the home must be shown, when clicked.

4.1.2 Stimulus/Response Sequences

When the user clicks the vacancy list option in the home screen, they will be redirected to the vacancy list page via hyperlink. Clicking on a particular home, will expand further details about the home.

4.1.3 Functional Requirements

A list displaying all the vacant homes, their addresses,etc.
Must display more details on a home when clicked on.

Employee Home page

Description and Priority

High Priority: An interface through which users can add data into our repository.
This feature aims to capture only the necessary data from the user, and will abstract the process of storing the data into the appropriate tables .

4.2.2 Stimulus/Response Sequences

The user will be redirected to this page, if they choose the employee home page option in the home page. They can choose the type of data they want to record, enter it , and click submit. The software can identify the where and how the data is to be stored and does accordingly.

4.2.3 Function Requirements

To contain a menu to choose what data to enter.
Also contains options to logout and redirect to homepage

4.3 Apartment features Page

4.3.1 Description and Priority

Medium Priority: An interface through which users can view the features of the apartment.

4.3.2 Stimulus/Response Sequences

The user will be redirected to this page, if they choose the apartment features option in the home page. They can see the description about each and every house of the apartment.

4.3.3 Functional Requirements

Must display all the details about a property/house.

4.4 Apartment Advertisement Page

4.4.1 Description and Priority

Medium Priority: It is an advertisement page for the apartment to highlight the facilities of the apartment, and make clear the details and policies of it.

4.4.2 Stimulus/Response Sequences

This will function as the home page.

4.5 Multiple user functionality

4.5.1 Description and Priority

High Priority: To enable employees to edit data about the apartment, using their own credentials. This would help to keep track which change has been updated by which employee.

4.5.2 Stimulus/Response Sequences

Upon clicking the login button in the home page, it will redirect to this page.

4.5.3 Functional Requirements

All users are expected to have their own unique login credentials, which would provide for their unique identification.

4.6 Employee Entry Page

4.6.1 Description and Priority

High Priority: A page to add details about new employees, with easy and straightforward UI, and store them in a serialised and retrievable way.

4.6.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

4.7 House Entry Page

4.7.1 Description and Priority

High Priority: A page to add new houses into apartments, to accomodate any future expansion projects for the apartment.

4.7.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

4.8 Tenants Entry Page

4.8.1 Description and Priority

High Priority: A page to add details about a new Tenant, and to identify their house, based on their apartment. To store the data in an easy and retrievable way.

4.8.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

4.9 Transactions Entry Page

4.9.1 Description and Priority

High Priority: A page to store all data related to apartment transactions, and to store them in a retrievable way.

4.9.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

4.10 Tenant Removal Page

4.10.1 Description and Priority

High Priority: A page which enables us to remove data about a particular tenant, without affecting any other related data.

4.10.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

4.11 House Removal Page

4.11.1 Description and Priority

High Priority: A page which enables us to remove data about a particular House, while deleting all tenant data related to it.

4.11.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

4.12 Employee Removal Page

4.12.1 Description and Priority

High Priority: A page which enables us to remove data about a particular employee, without affecting any other related data.

4.12.2 Stimulus/Response Sequences

Upon clicking the button in the Data entry page, it will redirect here.

Other Nonfunctional Requirements

Performance Requirements

The UI must be easy and intuitive .
The waiting and processing times must be reasonable.
Data entering and looking up must be fairly simple.

Safety Requirements

Data security is of utmost priority.
Data consistency is expected to be enforced on the data being stored.

Security Requirements

The Client is to be given their own login credentials to access the product.

Software Quality Attributes

The product must function consistently regardless of the scale of property involved.

Business Rules

Advanced deposit amount for new tenants is 10 or 15 months rent.

1 bhk homes will cost Rs.4000 per month in rent, with maximum accommodation of 2 adults and 1 kid whose age is less than 10 .

2 bhk homes will cost Rs.6000 per month in rent, with maximum accommodation of 4 adults or 2 adults and 2 kids , both aged less than 18.

3 bhk homes will cost Rs 10000 per month in rent, with maximum accommodation of 4 adults or 2 adults and 3 kids.

Other Requirements

-

Appendix A: Glossary

-

Appendix B: Analysis Models

-

Appendix C: To Be Determined List

6A
6B

Use Case Text

By Pigeons

Version 2.0

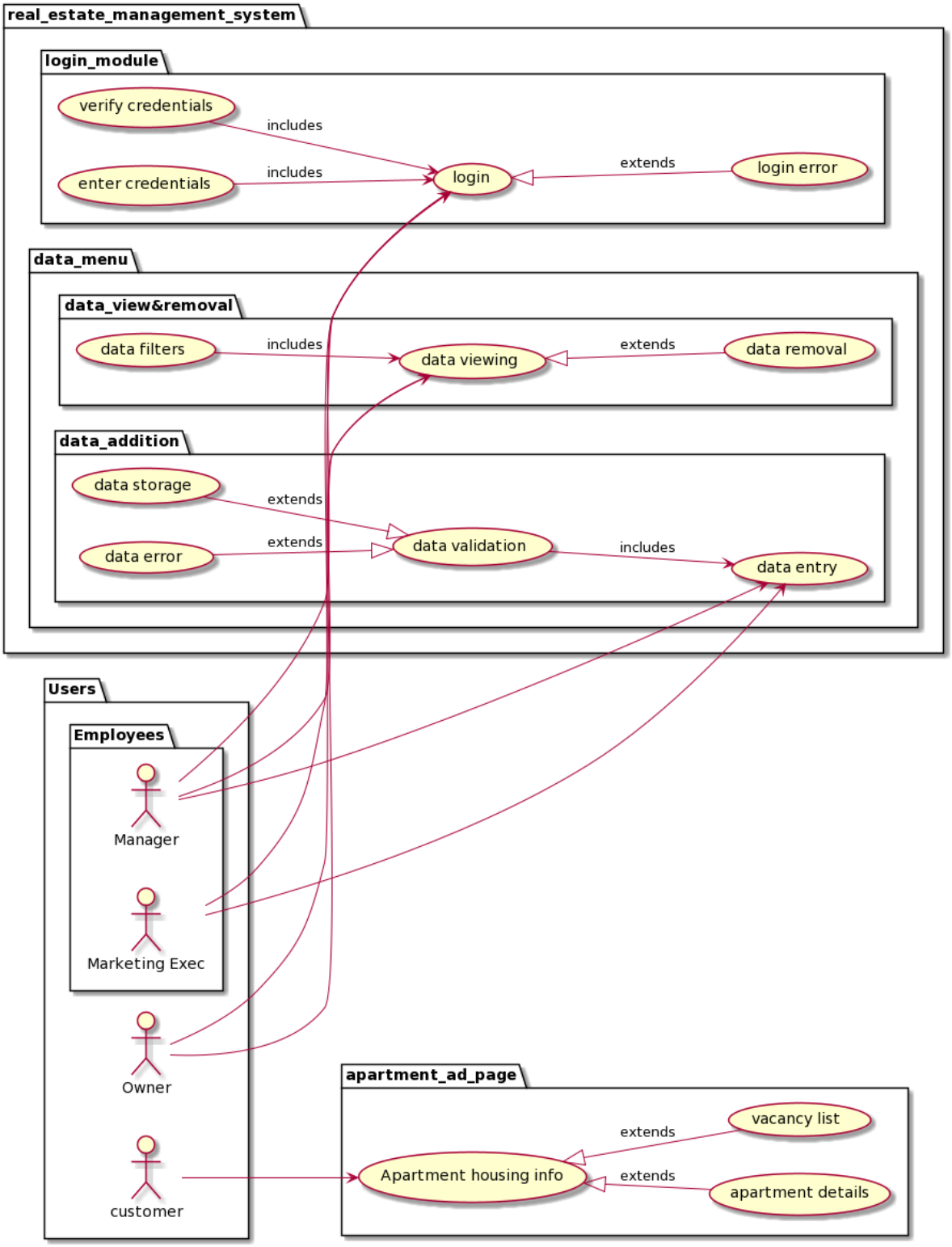
Revision History

Date	Author	Description of change
1/11/	V.Meganathan	Updates, to use case issues and formatting

Contribution Table:

Use Case Numbers	Contributor
UC-01,UC-04	Meganathan V
UC-02	Nandini R
UC-03	Navaneetha Krishnan S
UC-05,UC-06	Alagappan S

Use Case Diagram



Use Case Diagram Code:

```
@startuml
left to right direction
package Users{
actor customer as c1
package Employees{
actor "Marketing Exec" as e1
actor "Manager" as e2
}
actor "Owner" as e3
}

package apartment_ad_page{

usecase " Apartment housing info "  as shd
usecase "vacancy list" as vl
usecase "apartment details" as ad

shd<|--vl : extends
shd<|--ad : extends

c1-->shd
}

package real_estate_management_system{
```

```
package login_module{
usecase login
usecase "verify credentials" as vc
usecase "enter credentials" as ec
usecase "login error" as le
login<|--le : extends
ec-->login : includes
vc-->login : includes
e1->login
e2->login
e3->login
}
package data_menu{
package data_addition{
usecase "data storage" as ds
usecase "data entry" as de
usecase "data validation" as dval
usecase "data error" as der
dval-->de : includes
der--|>dval : extends
ds--|>dval : extends
e1-->de
e2-->de
}
package data_view&removal{
usecase "data viewing" as dview
usecase "data filters" as df
```

usecase "data removal" as dr

dview<|--dr : extends

df-->dview : includes

e2-->dview

e3-->dview

}

}

}

@enduml

Use Cases

Use Case: Apartment Advertisement Page

Id: UC- 01

Description

An overview of the apartment and its facilities, intended to serve as an advertisement to the apartment for potential tenants.

Level: High

Primary Actor

Customer

Supporting Actors

-

Stakeholders and Interests

Customers- They can view and explore the facilities provided by the apartment and even check for vacant homes.

Pre-Conditions

Customers should have Internet connection to access the webpage
The Vacancy list of available homes must be updated regularly.

Post Conditions

Success end condition

Customers are able to obtain the necessary information about apartments from the home page.
Vacancy lists data are consistent with real time home availability.
Valid contact information is provided.

Failure end condition:

Vacancy lists data are not updated
Apartment facilities are inconsistent with actual apartments, or are outdated.
Contacts provided are a dead-end.

Minimal Guarantee

Customers can view the home-page without any constraints.

Trigger

When a user clicks on a home in the vacancy list, details about that home will be displayed.

Main Success Scenario

The user is shown a detailed overview of the apartment and its facilities.
The user can view details of vacant homes available for rent.
The contact info provided is valid .

Extensions

If the contact info provided is incorrect:
If the contact information is displayed wrong, the admin can be informed to update the correct information.
If vacancy list information is obsolete:
It can be changed by employees
By logging in and updating the database.

Frequency: Once a while in two to three days .

Assumptions:
Internet Connection is available without any interrupt

Special Requirements

Performance

The website has to verify and direct the employee to the main page in reasonable time.

User Interface

The website shall display all options and messages in English language.

Security

The contact information provided will be of a secure line.

Issues

None

Use Case: Login page

Id: UC- 02

Description

The employee is asked to enter the provided login details to log in to the website.

Level: High

Primary Actors

Owner
Marketing Executive
Manager

Supporting Actors

Nil

Stakeholders and Interests

Employees and Owner –Enters credentials

Pre-Conditions

Users should have Internet connection to access the webpage
Users should have precise login credentials.

Post Conditions

Success end condition

User is successfully logged into the web page
User can easily log in again

Failure end condition:

If the provided login credentials is wrong then user will not be able to log in

Minimal Guarantee

User can view the homepage unconfined.

Trigger

When an unlogged User opens the webpage and enters the login credentials.

Main Success Scenario

The user enters the Username and Password
The system verifies if the provided credentials are correct.
If the entered credentials are correct then the employee will be directed to the main page.

Extensions

If the entered log in credentials are incorrect:
If the user forgets the password , they have to contact the admin.
By contacting admin they'll be directed to a set a new password
After creating new password , they can come back to login page
Then by entering the altered login credentials they can log in to the main page.

Frequency:

Once in two to three days .

Assumptions:

Internet Connection is available without any interrupt

Special Requirements

Performance

The website have to verify and direct the employee to main page within a reasonable time of user affirmation

User Interface

The website shall display all options and messages in English language.

Security

The system shall display the letters of password in a masked format when they are entered by the employee.
i.e. Mask the password with characters such as ****. Rationale – This is to ensure that a bystander will not be able to read the password being entered by the employee.

The password of the employees will be stored in a secure database

Issues

Error message flashing has to be handled.

To do

Sort out the flashing of error messages to the user.

Use Case: Data Menu

Id: UC- 03

Description:

The user can add, remove and view data using a choose function.

Level: High

Primary Actor:

Owner
Employees

Supporting Actors

Nil

Stakeholders and Interests

Users can choose what data to update

Pre-Conditions

Users should have Internet connection to access the webpage
Users should have appropriate login details to enter into the employee home page.

Post Conditions

Success end condition

Users is successfully logged into the web page
User can add, remove and view data easily again and again by choose function

Failure end condition:

If the provided login credentials is wrong then user will not be able to log in
Any error in login id the employee can not alter the data in employee home page

Minimal Guarantee

User login id is correct , they can alter and view the data by choosing a function in the home page.

Trigger

Users are successful in login credentials. By using the choose function they can add, remove and view the data.

Main Success Scenario

The User enters the Username and Password ,if the provided credentials are correct.
The system verifies it and move onto the home page from login page
They can see the choose function option in home page and they can alter and view the data

Extensions

If the entered log in credentials are incorrect:

If the user forgets the password , they can change it.
Then by entering the altered login credentials they can log in to main page
By using the choose function, they can alter and view data entries.
In case of any error in choosing a function, create a new login id and then use it properly.

Frequency

Once in two to three days .

Assumptions

Internet Connection is available without any interrupt

Special Requirements

Performance

The website have to verify and direct the employee to main page within 15 seconds of user affirmation

User Interface

The website shall display all options and messages in English language.

Security

The system shall display the letters of password in a masked format when they are entered by the employee. i.e. Mask the password with characters such as ****. Rationale – This is to ensure that a bystander will not be able to read the password being entered by the employee.

The data details which was added and or removed of the employees will be stored in a secure database

Issues

The entered data details may be wrong or forgot to remove that data entry and save as it is so be careful

To do

Identify login id and assuming the choose function (whether add, remove or view the data) and show your details
And make employees to alter and view the data

Use Case: Data Entry Page

Id: UC- 04

Description

The page where an employee updates a new record into the database.

Level: High

Primary Actor

Owner
Employees

Supporting Actors

Nil

Stakeholders and Interests

Employee- Is incharge of uploading accurate and consistent information.
Owner- requires the data entered here, for later use.

Pre-Conditions

User should have Internet connection to access the webpage
User must have access to accurate information.

Post Conditions

Success end condition

The concerned record is successfully updated to its corresponding table.
The data is accurate and consistent.

Failure end condition:

Data is not added to the table.

Minimal Guarantee

Users can see whether the table was successfully updated or not.

Trigger

When the employee clicks on what data to update in the employee homepage. After typing the fields in, click on the submit button.

Main Success Scenario

The database shows that it has been successfully updated.
The system verifies if the provided credentials are correct.
If the update is successful, the website will ask if the user wants to repeat the process or be redirected to the employee home page.

Extensions

If the data entered is incorrect:
There is a domain error while filling the data.
There is a system failure.
A mandatory column has been left empty.

Frequency

Once in two to three days .

Assumptions

Internet Connection is available without any interrupt.
Data entered is accurate and consistent.

Special Requirements

Performance

The wait time to update the changes into the database, must be reasonable.

User Interface

The website shall display all options and messages in English language.

Security

The data entered shall be secure, and can be viewed only if provided proper clearance and/or credentials..

Issues

None

To do

Identify the required database to store details entered.

Identify the constraints for different variable types.

Use Case: View data page

Id: UC- 05

Description

The employee can use is log in page use by filter constraints to view the data

Level: Medium

Primary Actor
Employee

Supporting Actors

Nil

Stakeholders and Interests

Employees can view transaction details

Pre-Conditions

Employee should have Internet connection to access the webpage
Employee have login id to view the data

Post Conditions

Success end condition

Employee is successfully logged into the web page
Employee can view the data successful

Failure end condition:

If the provided login credentials is wrong then user will not be able to log in
Any error in log in id the employee can not see this view data details

Minimal Guarantee

Employee login id is correct he can see the view data

Trigger

Employee login is successful By use of filter constraints they can see the view data

Main Success Scenario

The Employee enters the Username and Password enter successful
They can see they filter constraints works and see the view data from the view data page

Extensions:

If the entered log in credentials are incorrect:
If the user forgets the password , they can change it.
By filter constraints use properly they can see the view data details
In case any error in filter constraints create new login id use properly

Frequency

Once in two to three days .

Assumptions:

Internet Connection is available without any interrupt

Special Requirements

Performance

- The website have to verify and direct the employee to main page within 15 seconds of user affirmation

User Interface

- The website shall display all options and messages in English language.

Security

- The system shall display the letters of password in a masked format when they are entered by the employee.

i.e. Mask the password with characters such as ****. Rationale – This is to ensure that a bystander will not be able to read the password being entered by the employee.

- View data details of the employees will be stored in a secure database

Issues

View data details given wrong.It save as it is so be careful

To do

- Identify login id and assume filters and show your details
- And make employees to see the view data

Use Case: DATA REMOVAL PAGE

Id: UC- 06

Description

The employee can use is log in page use by filter constraints enter into the data removal page

Level: Medium

Primary Actor

Employee

Supporting Actors

Nil

Stakeholders and Interests

Employees can navigate to removal data

Pre-Conditions

- Employee should have Internet connection to access the webpage
- Employee have login id to and REMOVAL the data

Post Conditions

Success end condition

- Employee can successfully remove the data

Failure end condition:

- If the provided login credentials is wrong then user will not be able to log in
- Once remove the data it cannot be undo

Minimal Guarantee

- Employee can see the details and they remove easily

Trigger

- Employee constraints navigate to data removal page

Main Success Scenario

Employees can see the removal icon
Employees can remove the data successful

Extensions

incorrect:

- By filter constraints use properly they can see the remove data icon
- In case any error in filter constraints create new login id use properly

Frequency: Once a while in two to three days .

Assumptions:

- Internet Connection is available without any interrupt

Special Requirements

Performance

- The website have to verify and direct the employee to main page within a soon as of user affirmation

User Interface

- The website shall display all options and messages in English language.

Security

- Data remove details of the employees will be stored in a secure database

Issues

Data will once remove it cannot be undo it must be issues

To do

- Employees can successfully enter into the data removal page
- They can successfully remove data .They didn't need

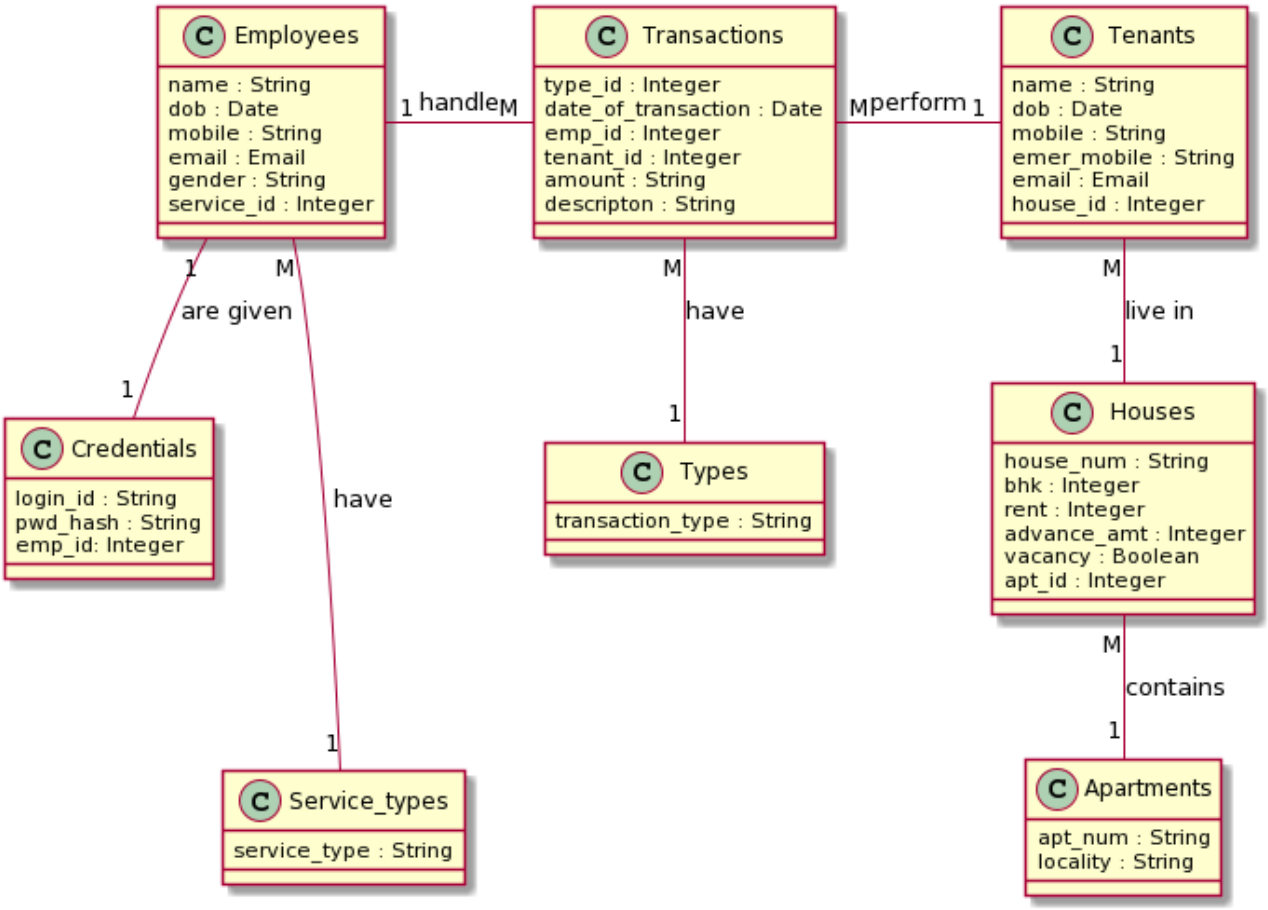
Project Design -High Level Design (HLD), and Low - Level Design (LLD)

For Real Estate Management System

By

Pigeons

High Level Design(HLD)

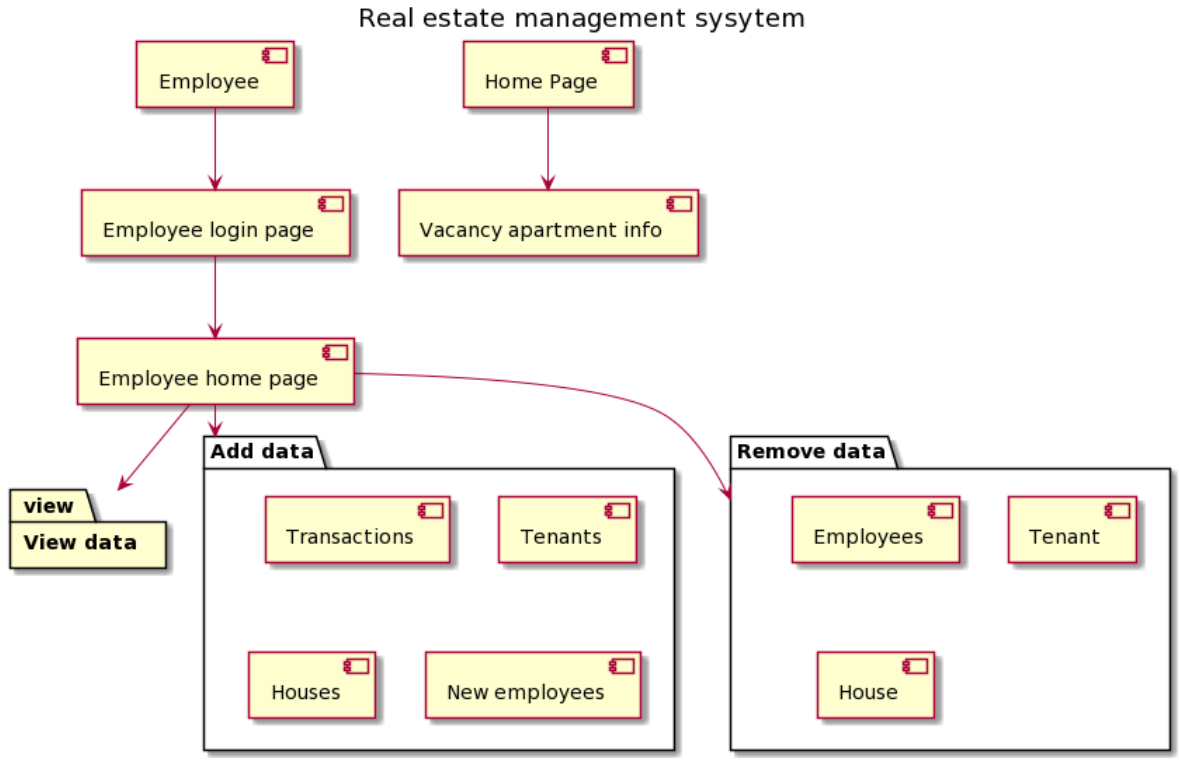


Script:

```
@startuml
class Employees {
   name : String
   dob : Date
   mobile : String
   email : Email
   gender : String
   service_id : Integer
}
class Tenants {
   name : String
   dob : Date
   mobile : String
   emer_mobile : String
   email : Email
   house_id : Integer
}

class Credentials {
  login_id : String
  pwd_hash : String
  emp_id: Integer
```

```
}
class Transactions {
  type_id : Integer
  date_of_transaction : Date
  emp_id : Integer
  tenant_id : Integer
  amount : String
  description : String
}
class Apartments {
  apt_num : String
  locality : String
}
class Houses {
  house_num : String
  bhk : Integer
  rent : Integer
  advance_amt : Integer
  vacancy : Boolean
  apt_id : Integer
}
class Service_types {
  service_type : String
}
class Types {
  transaction_type : String
}
Employees "M" --- "1" Service_types : have
Employees "1" -- "1" Credentials: are given
Employees "1"-right-"M" Transactions : handle
Transactions "M" -down- "1" Types : have
Tenants "1" -left- "M" Transactions : perform
Tenants "M" -- "1" Houses : live in
Apartments "1" -up- "M" Houses : contains
@enduml
```

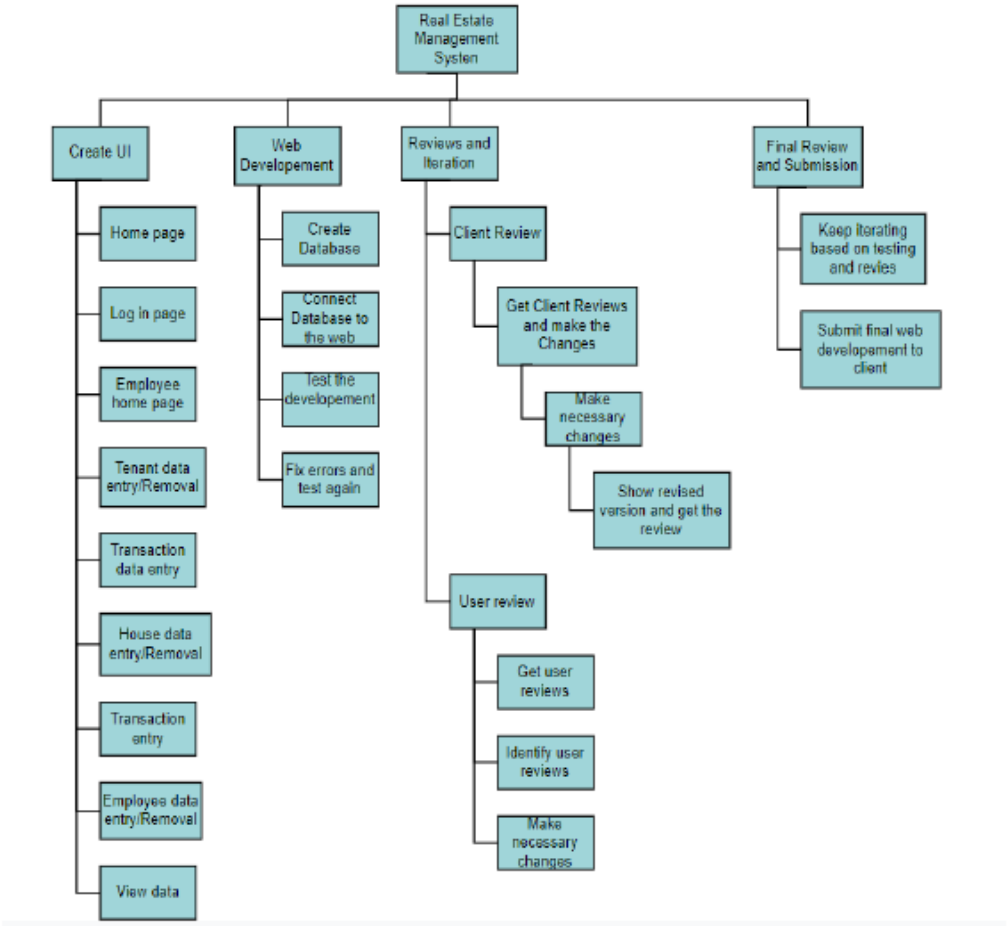
LOW LEVEL DESIGN (LLD):



Script;

```
@startuml
title Real estate management sysytem
[Home Page]-->[Vacancy apartment info]
component Employee as e
e-->[Employee login page]
[Employee login page]-->[Employee home page]
package "Add data" as a {
[Transactions]
[Tenants]
[Houses]
[New employees]
}
package "Remove data" as b {
[Employees]
[Tenant]
[House]
}
package "View data" as view
[Employee home page]-->a
[Employee home page]-->b
[Employee home page]-->view
@enduml
```

Work BreakDown Structure



Team Description

As per the above breakdown, the work was divided among the team members. The UI of the pages were designed together with inputs from all team members. Then the Front end of pages was handled by Nandini, Naveen and Alagappan, whereas the back end was handled by Meganathan.

As further sprints progressed, all members became more involved and worked on both front and back end. The entire project was developed in accordance with Agile principles, with regular meetings and user issues taken into consideration.

Implementation of Real Estate Management System

-Pigeons

Apartment Advertisement Page:

Code:

Front-end:

Nav Bar

```
<section class="coloured" id="title">
    <div class="container-fluid">

        <nav class="navbar navbar-expand-lg navbar-dark fixed-top">
            <a class="navbar-brand" href="">Lalitha apartments</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarTogglerDemo02"
                    aria-controls="navbarTogglerDemo01" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarTogglerDemo02">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="#title">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#footer">Contact</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#pricing">Vacancy list</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#apartment-info">Apt-info</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('login') }}">Log in</a>
                    </li>
                </ul>
            </div>
```

Carousel

```
</nav>
<div id="testimonial-carousel" class="carousel slide" data-ride="carousel" data-interval="false">
    <div class="carousel-inner">
        <div class="carousel-item active container-fluid">
            <div class=" col-12">
                <img class="testimonial-img" src="{{ url_for('static', filename= 'images/img-1.jpeg') }}"
                    alt="apartment pic 1">
            </div>
        </div>

        <div class="carousel-item container-fluid">

            <div class=" col-12">
                <img class="testimonial-img" src="{{ url_for('static', filename= 'images/img-2.jpeg') }}"
                    alt="apartment pic 2">
            </div>


        </div>

        <div class="carousel-item container-fluid">

            <div class=" col-12">
                <img class="testimonial-img" src="{{ url_for('static', filename= 'images/img-3.JPG') }}"
                    alt="apartment pic 3">
            </div>

        </div>
```

Apartment Information

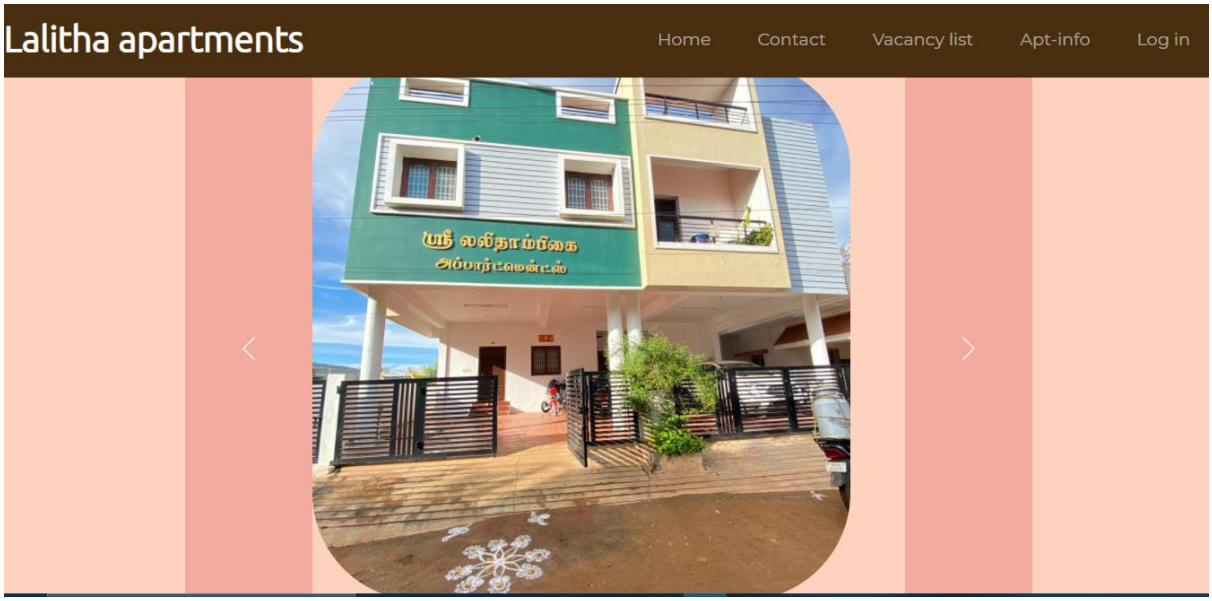
```
<!-- Apartment info -->

<section class="coloured" id="apartment-info">
    <h2 class="section-heading">Apartment INFO</h2>
    <ul>
        <li id="rule-list">Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
        <li id="rule-list">Integer sed eros lobortis, feugiat felis sit amet, mollis turpis.</li>
        <li id="rule-list">Nunc in tortor elementum, ultrices urna eu, porta urna.</li>
        <li id="rule-list">Vestibulum condimentum tellus eget risus molestie, non porta urna viverra.</li>
        <li id="rule-list">Suspendisse vehicula lacus et fermentum auctor.</li>
    </ul>
</section>

<!-- Vacancy list -->
```

Back-end:

Routing

```
@app.route('/')
@app.route('/home')
def index():
    headings = ("BHK", "Locality", "Rent(in Thousands)", "Advance(in Thousands)", " ")
    ap = Apartment.query
    ids = map(lambda x: x.id, ap)
    places = map(lambda x: x.locality, ap)
    locs = dict(zip(ids, places))
    houses = House.query.filter(
        House.id.not_in(map(lambda x: x[0], Tenant.query.with_entities(Tenant.house_id).all()))).all()

    return render_template('homepage.html', headings=headings, houses=houses, apts=locs)


@app.route('/login', methods=['GET', 'POST'])
```

Output:



Vacancy List:

Code:

Front-end:

```
<!-- Vacancy list -->

<section class="white-section" id="pricing">

    <h2 class="section-heading">Vacancy List of all Apartments</h2>
    <p> Affordable price for houses</p>

    <table class="table table-hover">
        <thead class="table-dark">
        <tr>
            {% for header in headings %}
                <th>{{ header }}</th>
            {% endfor %}
        </tr>
        </thead>
        <tbody>
        {% for house in houses %}
            <tr>
                <td> {{ house.bhk }}</td>
                <td> {{ apts[house.apt_id] }} </td>
                <td> {{ house.rent }} </td>
                <td> {{ house.advance }} </td>
                <td>
                    <button type="button" class="btn btn-danger ">More info</button>
                </td>
            </tr>
        {% endfor %}
        </tbody>
```

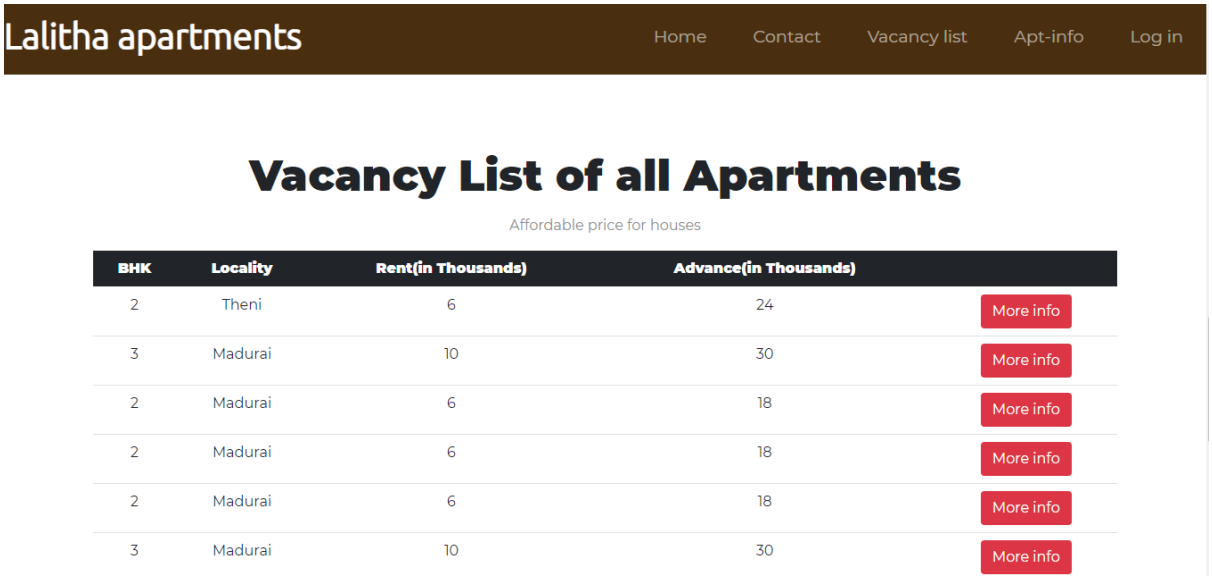
Back-end:

Table model

```
class House(db.Model):
    __tablename__ = 'house'
    id = db.Column(db.Integer, primary_key=True)
    house_num = db.Column(db.String(5), index=True, unique=False)
    bhk = db.Column(db.String(3), index=True, unique=False)
    rent = db.Column(db.String(4), index=True, unique=False)
    advance = db.Column(db.String(4), index=True, unique=False)
    vacancy = db.Column(db.Boolean, index=True, nullable=False, unique=False)
    apt_id = db.Column(db.Integer, db.ForeignKey('apartment.id'))
    tenant = db.relationship("Tenant", uselist=False, backref="house")

    def __repr__(self):
```

Query(in routes.py)

Output:


Lalitha apartments
Home
Contact
Vacancy list
Apt-info
Log in
Vacancy List of all Apartments
Affordable price for houses
BHK
Locality
Rent(in Thousands)
Advance(in Thousands)
2 Theni 6 24 More info
3 Madurai 10 30 More info
2 Madurai 6 18 More info
2 Madurai 6 18 More info
2 Madurai 6 18 More info
3 Madurai 10 30 More info

Multiple User Functionality:

Code:

Front-end:

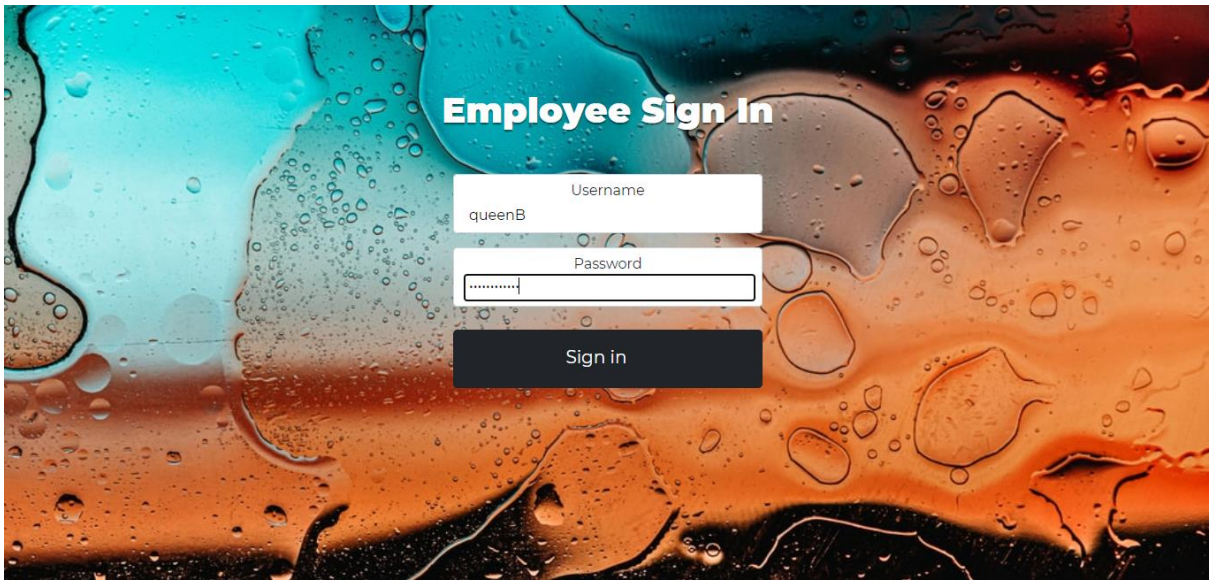
```
<body>

<div id="fullscreen_bg" class="fullscreen_bg"/>

<div class="container">
    <h1 class="form-signin-heading">Employee Sign In</h1>
    <form action="" method="post" novalidate class="form-signin">
        {{ form.hidden_tag() }}
        <p class="form-control">
            {{ form.username.label}}<br>
            {{ form.username(size=32) }}<br>
            {% for error in form.username.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p class="form-control">
            {{ form.password.label }}<br>
            {{ form.password(size=32) }}<br>
            {% for error in form.password.errors %}
            <span style="...">[{{ error }}]</span>
            {% endfor %}
        </p>
        <button class="btn btn-lg btn-dark btn-block col-12">Sign in{{ form.submit(class="btn") }}</button>
    </form>
</div>
```

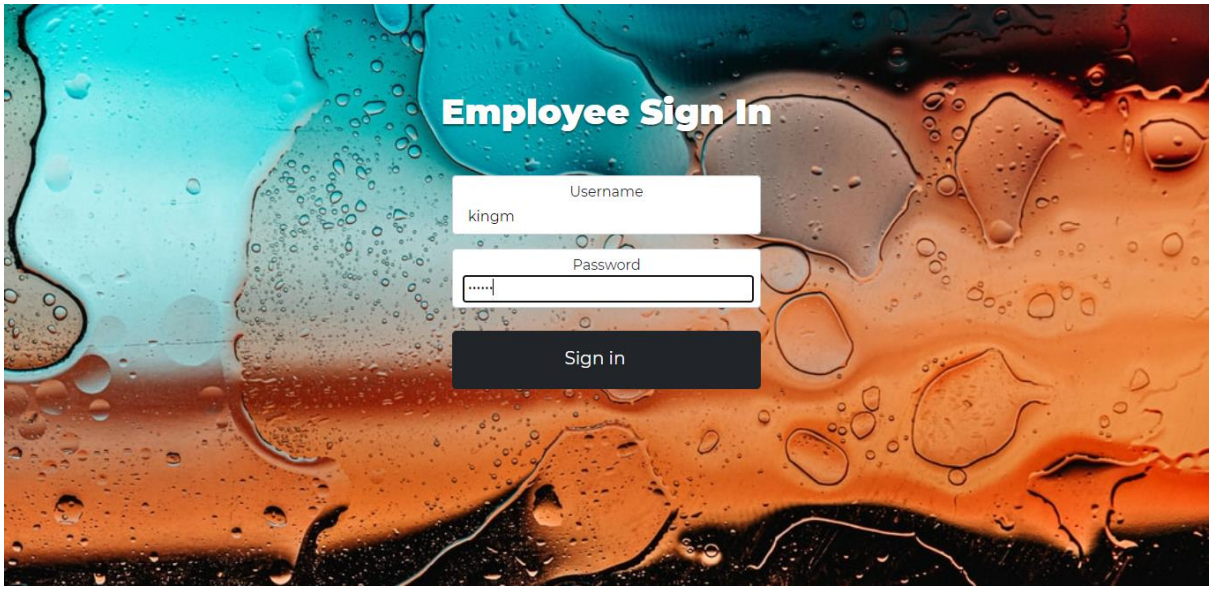
Back-end: Table Model

```
class User(UserMixin, db.Model):
    __tablename__ = 'user'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), index=True, unique=True)
    password_hash = db.Column(db.String(128))
    emp_id = db.Column(db.Integer, db.ForeignKey('employee.id'))

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

    def __repr__(self):
        return '<User {}>'.format(self.username)


@login.user_loader
def load_user(id):
    return User.query.get(int(id))
```

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home2'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password')
            return redirect(url_for('login'))
        login_user(user)
        next_page = request.args.get('next')
        if not next_page or url_parse(next_page).netloc != '':
            next_page = url_for('home2')
        return redirect(next_page)
    return render_template('login.html', title='Sign In', form=form)
```

Output:





Employee Home Page:

Code:

Front-end:

```
<section id="Add_Data">
    <h2>Add Data</h2>
    <div class="row">
        <div class="col-lg-8">
            <div class="d-grid gap-5 d-md-flex justify-content-md-end">
                <button type="button" class="btn btn-dark btn-lg"><a class="anchor" href="{{ url_for('add_trans') }}
                </button>
                <button type="button" class="btn btn-dark btn-lg"><a class="anchor" href="{{ url_for('add_tenant') }
                </button>
                <button type="button" class="btn btn-dark btn-lg"><a class="anchor" href="{{ url_for('add_house') }}
                </button>
                <button type="button" class="btn btn-dark btn-lg"><a class="anchor"
                                                                     href="{{ url_for('add_employee') }}">New
                    Employees</a></button>
            </div>
        </div>
    </div>
</section>
```

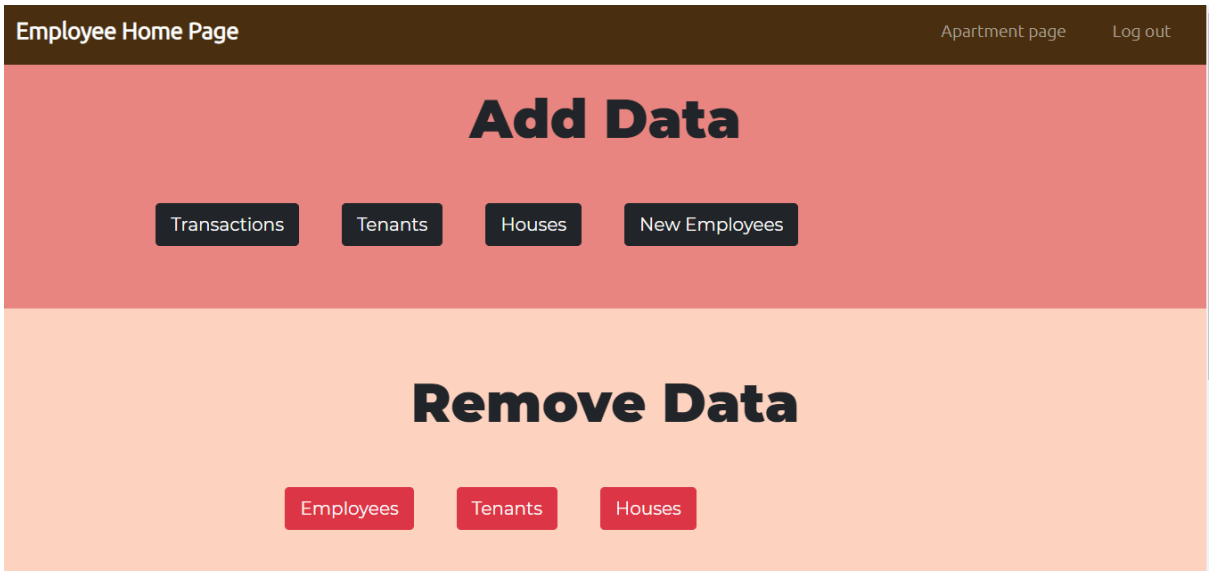
```
<section id="Remove_Data">
    <h2>Remove Data</h2>
    <div class="row">
        <div class="col-lg-7">
            <div class="d-grid gap-5 d-md-flex justify-content-md-end">
                <button type="button" class="btn btn-danger btn-lg"><a class="anchor" href="{{ url_for('rem_employee
                <button type="button" class="btn btn-danger btn-lg" ><a class="anchor" href="{{ url_for('rem_tenant'
                <button type="button" class="btn btn-danger btn-lg"><a class="anchor" href="{{ url_for('rem_house')
            </div>
        </div>
    </div>
</section>
```

Back-end:

Routing

```
@app.route('/home2')
@login_required
def home2():
    return render_template('index.html')
```

Output:



Employee Entry Page:

Code:

Front-end:

```
{% extends 'base.html' %}
{% block heading %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/employee.css') }}">
<title>Employee entry page</title>
<!-- Bootstrap DatePicker -->
<script type="text/javascript">
  $(function() {
    $('#txtDate').datepicker({
      format: "dd/mm/yyyy"
    });
  });

</script>

{% endblock %}

{% block content %}
<div class="container register">
    <div class="row">
        <div class="col-md-3 register-left">
            <h3 class="register-heading" style="...">Employees data entry page</h3>
        </div>
        <div class="col-md-9 register-right">

            <div class="tab-content" id="myTabContent">
                <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">
                    <h3 class="register-heading2">Enter Employee Data</h3>
```

Back-end:

Routing and Table updation

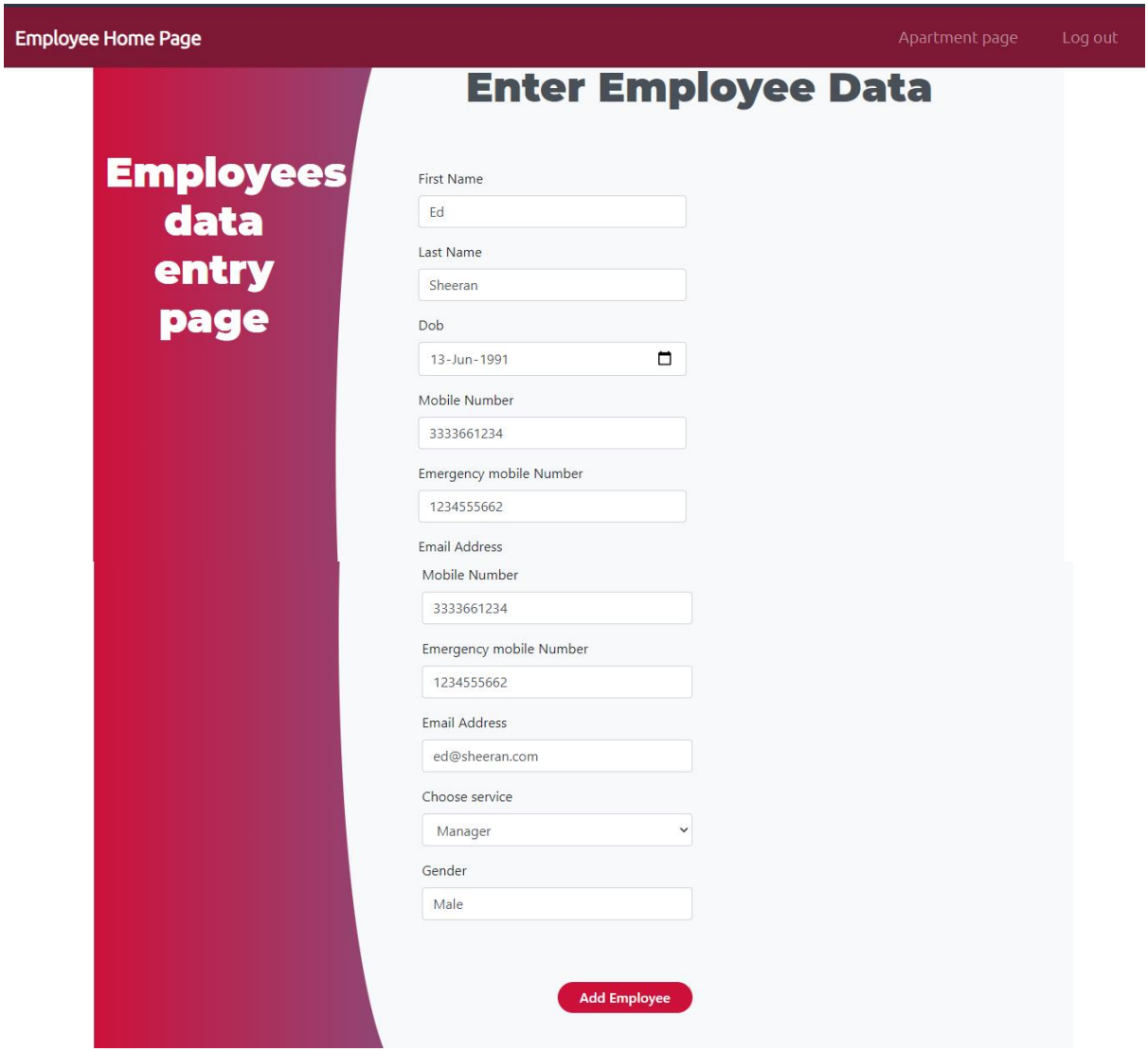
```
@app.route('/addemp', methods=['GET', 'POST'])
def add_employee():
    form = EmployeeAddForm()
    if form.validate_on_submit():
        emp = Employee(fname=form.firstname.data,
                       lname=form.lastname.data,
                       mobile=form.mobile.data,
                       emer_num=form.emer_mobile.data,
                       dob=form.DoB.data,
                       email=form.email.data,
                       gender=form.gender.data,
                       service_id=form.service_list.data.id)
        db.session.add(emp)
        db.session.commit()
        return render_template("ack1.html")
    return render_template('employee.html', form=form)
```

```python
class Employee(db.Model):
    __tablename__ = 'employee'
    id = db.Column(db.Integer, primary_key=True)
    fname = db.Column(db.String(25), index=True, unique=False)
    lname = db.Column(db.String(25), index=True, unique=False)
    mobile = db.Column(db.String(10), index=True, unique=True)
    emer_num = db.Column(db.String(10), index=False, unique=False)
    dob = db.Column(db.DateTime, nullable=False)
    email = db.Column(db.String(120), index=True, unique=True)
    gender = db.Column(db.String(10), index=True, unique=False)
    service_id = db.Column(db.Integer, db.ForeignKey('service.id'))
    user = db.relationship("User", uselist=False, backref="employee")
    receiver = db.relationship("Transaction", uselist=False, backref="employee")

    def __repr__(self):
        return '<Employee {}>    '.format(self.fname)
```

Form

```python
class EmployeeAddForm(FlaskForm):
    firstname = StringField('First Name', validators=[DataRequired()])
    lastname = StringField('Last Name', validators=[DataRequired()])
    DoB = DateField(validators=[DataRequired()])
    mobile = StringField('Mobile Number', validators=[DataRequired(), Length(min=7, max=10,
                                                                              message='Name length must be between %(
    emer_mobile = StringField('Emergency mobile Number', validators=[DataRequired(), Length(min=7, max=10,
                                                                                             message='Name length mus
    email = StringField('Email Address', validators=[DataRequired(), Email(), Length(max=60)])
    service_list = QuerySelectField('Choose service', query_factory=lambda: Service.query, allow_blank=False,
                                    get_label='service_type')
    gender = StringField('Gender', validators=[DataRequired(), AnyOf(values=['Male', 'Female'],
                                                                     message='Gender must be male or female')])
    submit = SubmitField('Add Employee')

    def validate_email(self, email):
        emp = Employee.query.filter_by(email=email.data).first()
        if emp is not None:
            raise ValidationError('Please use a different email address.')
```

Output:


Employee Home Page
Apartment page
Log out
Employees data entry page
Enter Employee Data
First Name
Ed
Last Name
Sheeran
Dob
13-Jun-1991
Mobile Number
3333661234
Emergency mobile Number
1234555662
Email Address
Mobile Number
3333661234
Emergency mobile Number
1234555662
Email Address
ed@sheeran.com
Choose service
Manager
Gender
Male
Add Employee

House Entry Page:

Code:

Front End:

```
{% extends 'base.html' %}
{% block heading %}
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/houses.css') }}">
<title>Houses Page</title>

{% endblock %}
{% block content %}
<div class="container register">
    <div class="row">
        <div class="col-md-3 register-left">
            <h3 class="register-heading" style="...">Houses data entry page</h3>
        </div>
        <div class="col-md-9 register-right">

            <div class="tab-content" id="myTabContent">
                <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">
                    <h3 class="register-heading2">Enter the data here</h3>
                    <div class="row register-form">
                        <div class="col-md-6">
                            <form action="{{ url_for('add_house') }}" method="POST">
                                {{ form.hidden_tag() }}
```

Back End:

Routing and Table Updation

```
@app.route('/addhouse', methods=['GET', 'POST'])
def add_house():
    form = HouseAddForm()
    if form.validate_on_submit():
        house = House(house_num=form.house_num.data,
                      bhk=form.bhk.data,
                      rent=form.rent.data,
                      advance=form.advance.data,
                      vacancy=False,
                      apt_id=form.apt_num.data.id,
                      )
        db.session.add(house)
        db.session.commit()
        return render_template("ack1.html")
    return render_template('houses.html', form=form)
```

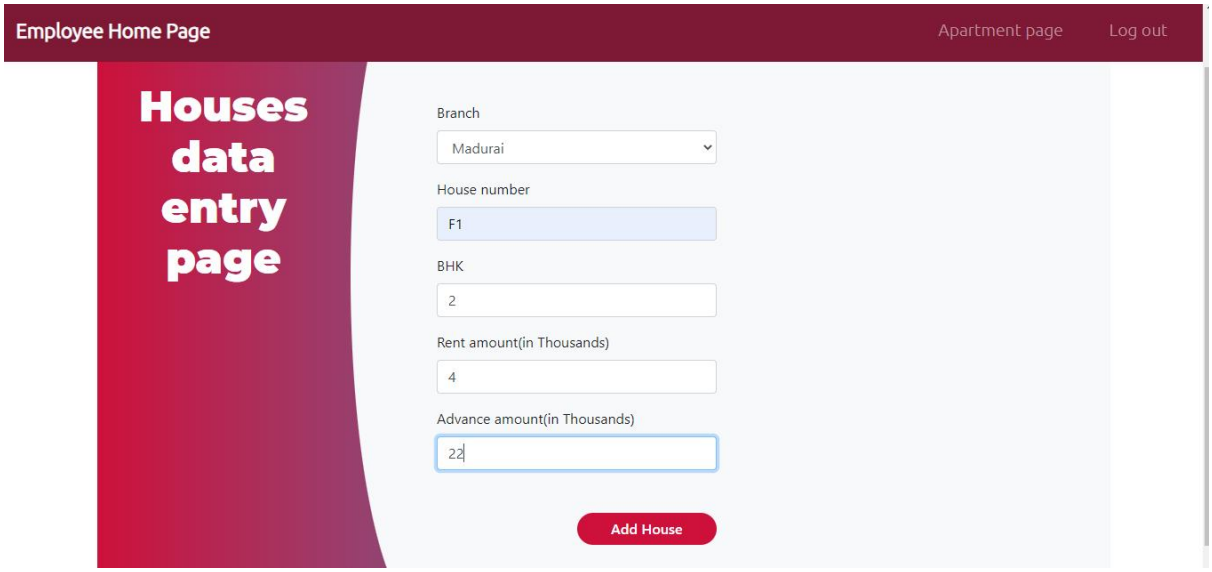
Table Model

```
class House(db.Model):
    __tablename__ = 'house'
    id = db.Column(db.Integer, primary_key=True)
    house_num = db.Column(db.String(5), index=True, unique=False)
    bhk = db.Column(db.String(3), index=True, unique=False)
    rent = db.Column(db.String(4), index=True, unique=False)
    advance = db.Column(db.String(4), index=True, unique=False)
    vacancy = db.Column(db.Boolean, index=True, nullable=False, unique=False)
    apt_id = db.Column(db.Integer, db.ForeignKey('apartment.id'))
    tenant = db.relationship("Tenant", uselist=False, backref="house")

    def __repr__(self):
        return '<House {}>'.format(self.house_num)
```

Form

```
class HouseAddForm(FlaskForm):
    apt_num = QuerySelectField('Branch', query_factory=lambda: Apartment.query, allow_blank=False,
                               get_label='locality')
    house_num = StringField('House number', validators=[DataRequired()])
    bhk = StringField('BHK', validators=[DataRequired(),Length(max=2)])
    rent = StringField('Rent amount(in Thousands)', validators=[DataRequired(),Length(max=2,message='Enter amount in
    advance = StringField('Advance amount(in Thousands)', validators=[DataRequired(),Length(max=2,message='Enter amou
    submit = SubmitField('Add House')
```

Output:



Tenants Addition Page:

Code

Front-end:

```
{% extends 'base.html' %}
{% block heading %}
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/tenants.css') }}">

    <!-- Bootstrap DatePicker -->
    <script type="text/javascript">
        $(function () {
            $('#txtDate').datepicker({
                format: "dd/mm/yyyy"
            });
        });


    </script>

    <script type="text/javascript">
        $(function () {
            $('#txtDate2').datepicker({
                format: "dd/mm/yyyy"
            });
        });
```

Back-end:

Routing and Table Updation

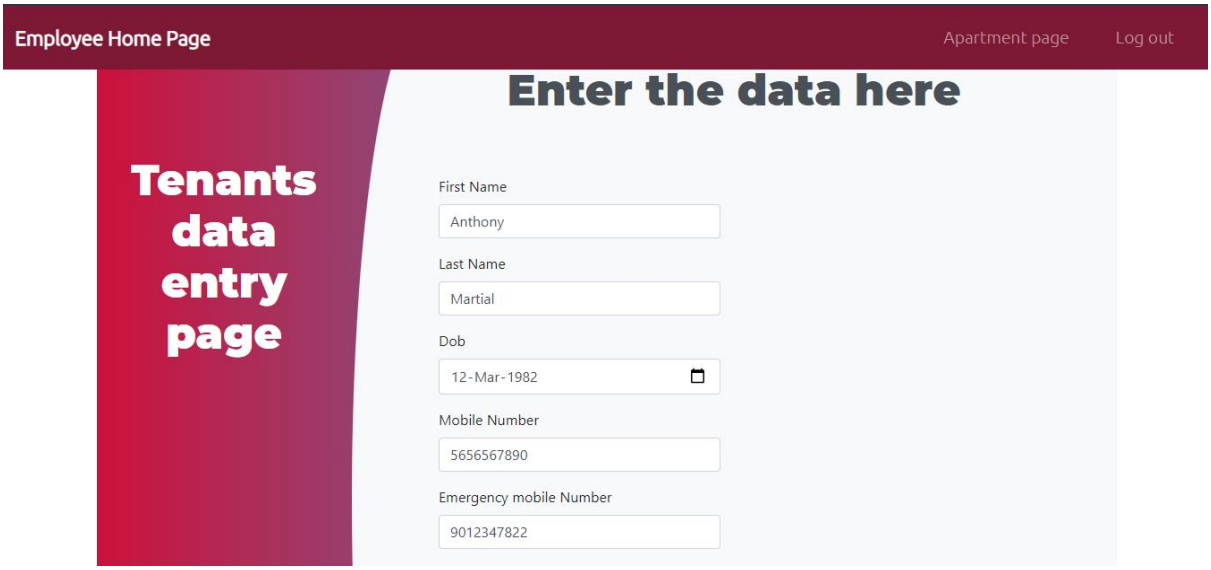
```
@app.route('/addtenant', methods=['GET', 'POST'])
def add_tenant():
    form = TenantAddForm()
    form.house_num.choices = [(house.id, house.house_num) for house in House.query.filter_by(apt_id=1).all()]
    if form.validate_on_submit():
        tenant = Tenant(fname=form.firstname.data,
                        lname=form.lastname.data,
                        mob_num=form.mobile.data,
                        emer_num=form.emer_mobile.data,
                        email=form.email.data,
                        dob=form.DoB.data,
                        Spouse_num=form.spouse_mob.data,
                        house_id=form.house_num.data
                        )
        db.session.add(tenant)
        db.session.commit()
        return render_template('ack1.html')
    return render_template('tenants.html', form=form)
```

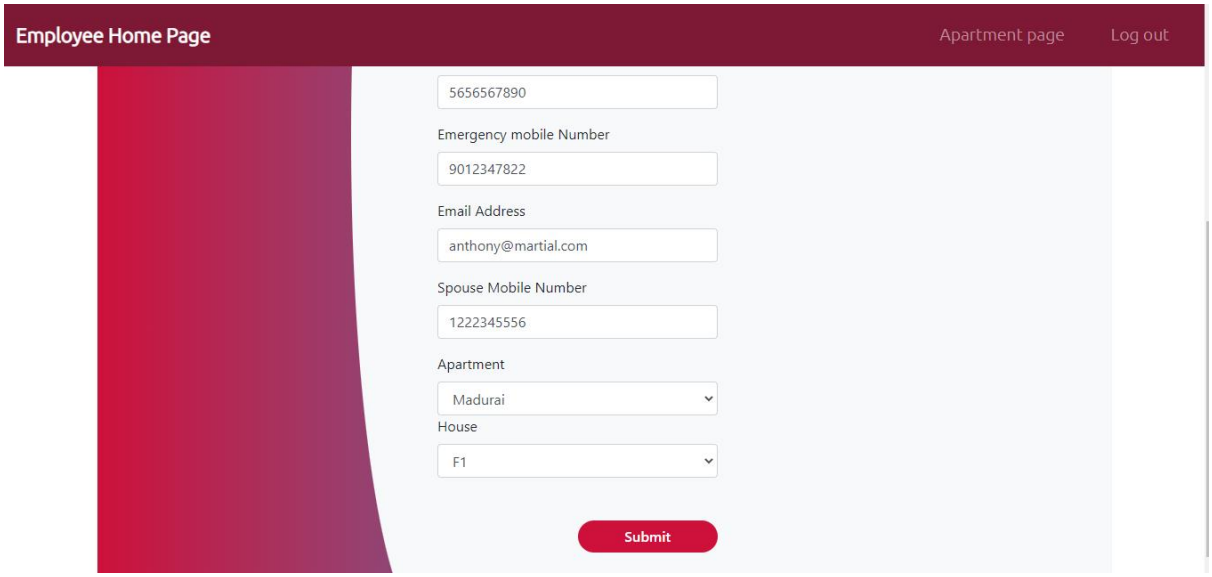
Table Model

```
class Tenant(db.Model):
    __tablename__ = 'tenant'
    id = db.Column(db.Integer, primary_key=True)
    fname = db.Column(db.String(25), index=True, unique=False)
    lname = db.Column(db.String(25), index=True, unique=False)
    mob_num = db.Column(db.String(10), index=True, unique=True)
    emer_num = db.Column(db.String(10), index=True, unique=False)
    email = db.Column(db.String(120), index=False, unique=True)
    dob = db.Column(db.DateTime, nullable=False)
    Spouse_num = db.Column(db.String(10), unique=False, nullable=True)
    house_id = db.Column(db.Integer, db.ForeignKey('house.id'))
    sender = db.relationship("Transaction", uselist=False, backref="tenant")

    def __repr__(self):
        return '<Tenant {}>'.format(self.fname)
```

Form

```
class TenantAddForm(FlaskForm):
    firstname = StringField('First Name', validators=[DataRequired()])
    lastname = StringField('Last Name', validators=[DataRequired()])
    DoB = DateField(validators=[DataRequired()])
    mobile = StringField('Mobile Number', validators=[DataRequired(), Length(min=7, max=10,
                                                                              message='phone number length must b
    emer_mobile = StringField('Emergency mobile Number', validators=[DataRequired(), Length(min=7, max=10,
                                                                                            message='phone numbe
    email = StringField('Email Address', validators=[DataRequired(), Email(), Length(max=60)])
    spouse_mob = StringField('Spouse Mobile Number')
    apt_num = SelectField('Apartment', choices=['Theni', 'Madurai', 'Dindigul'], validate_choice=False)
    house_num = SelectField('House', validate_choice=False)
    submit = SubmitField('Add Tenant')
```


Employee Home Page
Apartment page
Log out
Tenants data entry page
Enter the data here
First Name
Anthony
Last Name
Martial
Dob
12-Mar-1982
Mobile Number
5656567890
Emergency mobile Number
9012347822


Employee Home Page
Apartment page
Log out
5656567890
Emergency mobile Number
9012347822
Email Address
anthony@martial.com
Spouse Mobile Number
1222345556
Apartment
Madurai
House
F1
Submit

Transaction Addition Page

Code

Front-end

```
{% extends 'base.html' %}
{% block heading %}
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/transaction.css') }}">

    <!-- Bootstrap DatePicker -->
    <script type="text/javascript">
        $(function () {
            $('#txtDate').datepicker({
                format: "dd/mm/yyyy"
            });
        });
    </script>


{% endblock %}

{% block content %}

    <div class="container register">
    <div class="row">
    <div class="col-md-3 register-left">
        <h4 class="register-heading" style="...">Transactions entry page</h4>
    </div>
    <div class="col-md-9 register-right">

    <div class="tab-content" id="myTabContent">
    <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">
        <h3 class="register-heading2">Enter the data here</h3>
```

Back-end:

Routing and Table Updation:

```
@app.route('/addtrans', methods=['GET', 'POST'])
def add_trans():
    form = TransactionAddForm()
    form.house_num.choices = [(house.id, house.house_num) for house in House.query.filter_by(apt_id=1).all()]
    form.tenant_id.choices = [(tenant.id, tenant.fname) for tenant in Tenant.query.filter_by(house_id=1).all()]
    if form.validate_on_submit():
        transaction = Transaction(type_id=form.types_list.data.id,
                                  dot=form.Dot.data,
                                  ten_id=form.tenant_id.data,
                                  emp_id=form.employee_list.data.id,
                                  amt=form.amount.data,
                                  desc=form.description.data
                                  )
        db.session.add(transaction)
        db.session.commit()
        return render_template("ack1.html")
    return render_template('transactions.html', form=form)
```

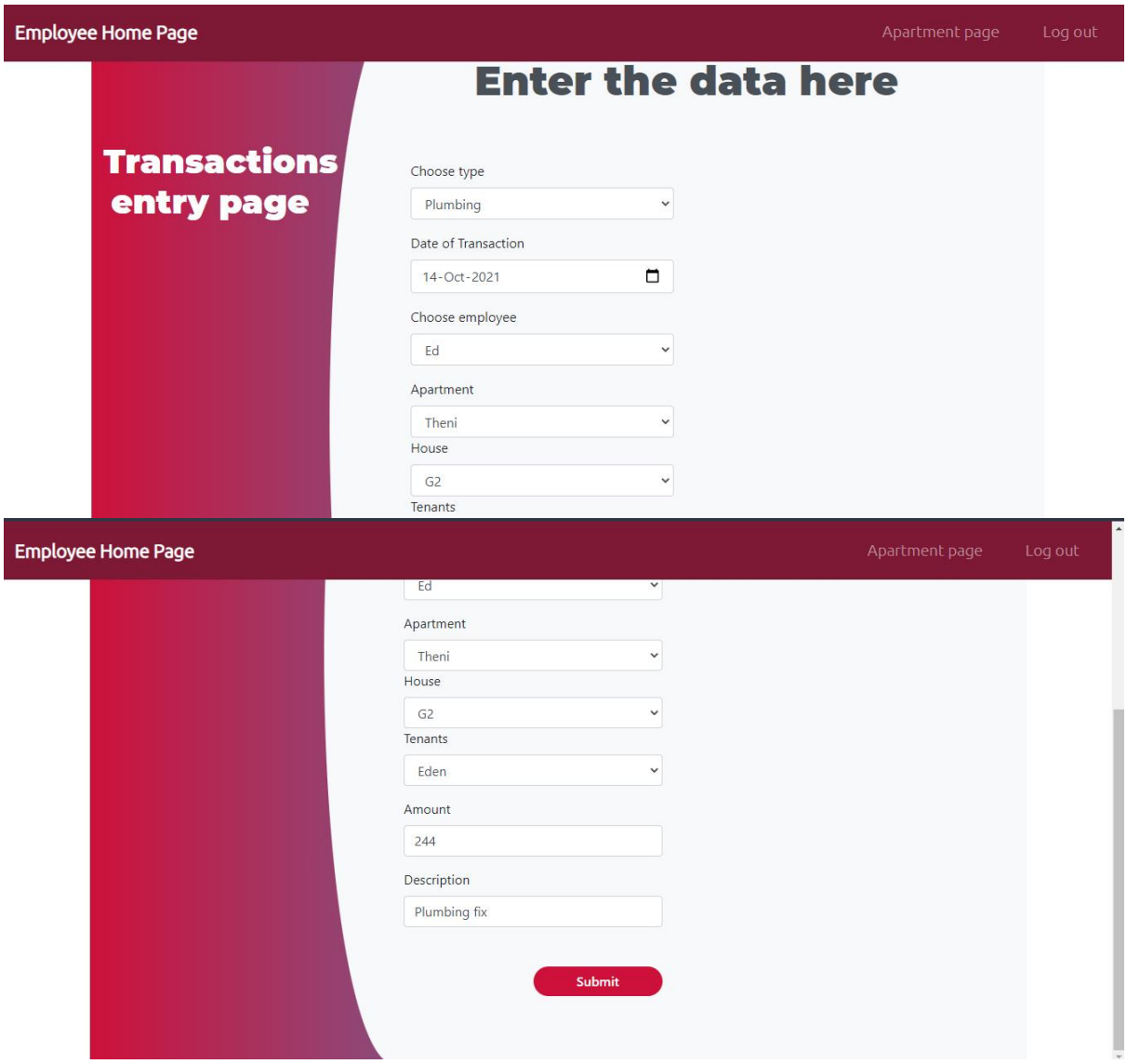
Table Model

```
class Transaction(db.Model):
    __tablename__ = 'transaction'
    id = db.Column(db.Integer, primary_key=True)
    type_id = db.Column(db.Integer, db.ForeignKey('types.id'))
    dot = db.Column(db.DateTime, nullable=False)  # date of transaction
    emp_id = db.Column(db.Integer, db.ForeignKey('employee.id'))
    ten_id = db.Column(db.Integer, db.ForeignKey('tenant.id'))
    amt = db.Column(db.String(4), unique=False)
    desc = db.Column(db.String(40), unique=False)

    def __repr__(self):
        return '<Sent {}>    '.format(self.amt)
```

Form

```
class TransactionAddForm(FlaskForm):
    types_list = QuerySelectField('Choose type', query_factory=lambda: Types.query, allow_blank=False,
                                  get_label='transaction_type')
    Dot = DateField(validators=[DataRequired()], label='Date of Transaction')
    employee_list = QuerySelectField('Choose employee',
                                     query_factory=lambda: Employee.query.filter(
                                         Employee.service_id.in_((6, 7, 8))).all(),
                                     allow_blank=False,
                                     get_label='fname')
    apt_num = SelectField('Apartment', choices=['Theni', 'Madurai', 'Dindigul'], render_kw={'class': 'form-control'
                          validate_choice=False)
    house_num = SelectField('House', render_kw={'class': 'form-control'}, validate_choice=False)
    tenant_id = SelectField('Tenants', render_kw={'class': 'form-control'}, validate_choice=False)
    amount = StringField('Amount', validators=[DataRequired(),Length(min=1, max=2,message='Enter amount in thousand
    description = StringField('Description', validators=[DataRequired()])
    submit = SubmitField('Add Transaction')
```

Output


Employee Home Page
Apartment page
Log out
Transactions entry page
Enter the data here
Choose type
Plumbing
Date of Transaction
14-Oct-2021
Choose employee
Ed
Apartment
Theni
House
G2
Tenants
Employee Home Page
Apartment page
Log out
Ed
Apartment
Theni
House
G2
Tenants
Eden
Amount
244
Description
Plumbing fix
Submit

Employee Removal Page

Code

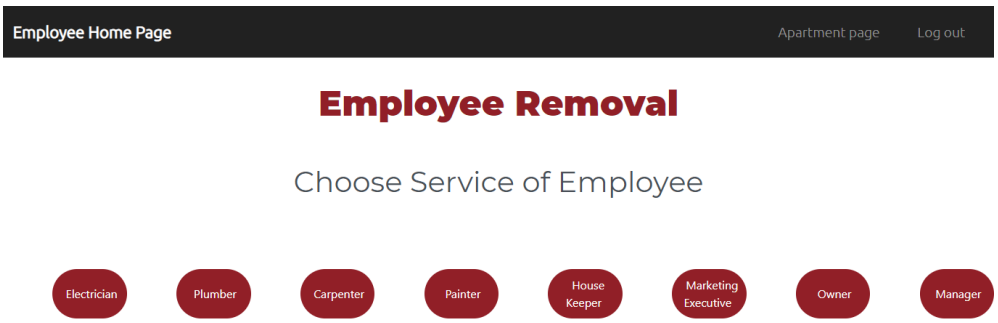
Front-end:

```
{% extends 'base.html' %}
{% block heading %}
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/rem_employee.css') }}">
    <!-- <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'list.html') }}"> -->
    <title>Employee Remove page</title>

{% endblock %}

{% block content %}
    <div class="container register">
        <h3 class="register-heading">Employee Removal</h3>
    </div>

    <div class="tab-content" id="myTabContent">
        <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">
            <h3 class="register-heading2">Choose Service of Employee</h3>
            <div class="row register-form">
                <div class="col-md-12">
                    <div class="form-group">
                        <div class="row">
                            <div class="col-12 ">
                                <div class="d-grid gap-5 d-md-flex justify-content-md-end">
                                    {% for service in services %}
                                    <button type="button" class="btnRegister"><a class="anchor"
                                                                                 href="{{ url_for('emp_list',id=se
                                    </button>
                                    {% endfor %}
                                </div>

>
id gap-5 d-md-flex justify-content-md-end">
ce in services %}
="button" class="btnRegister"><a class="anchor"
                                 href="{{ url_for('emp_list',id=service.id) }}">{{ service.service_type }}</a>


</head>
<body>
<section id="title">
    <div class="container-fluid">
        <nav class="navbar navbar-expand-lg navbar-dark fixed-top">
            {% block nav %}
            {% endblock %}
            <a class="navbar-brand " href="{{ url_for('home2') }}">Employee Home Page</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                    data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
                    aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('index') }}">Apartment page</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('logout') }}">Log out</a>
                    </li>

                </ul>
            </div>
```

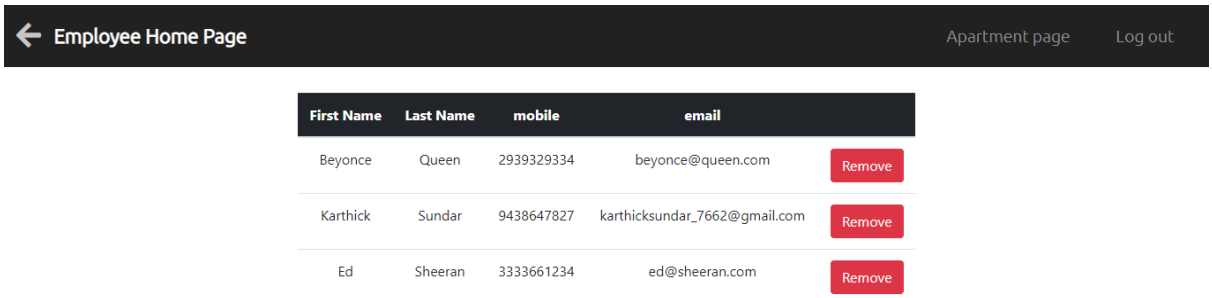
Back-end

Routing and Link allocation

```python
@app.route('/rememployee')
def rem_employee():
    services = Service.query
    return render_template('rem_employee.html', services=services)


@app.route('/rem_list/<int:id>')
def emp_list(id):
    headings = ("First Name", "Last Name", "mobile", "email", " ")
    emps = Employee.query.filter_by(service_id=id).all()
    return render_template('emp_rlist.html', headings=headings, data=emps)


@app.route('/delete_emp/<int:id>')
def delete_emp(id):
    emp_todel = Employee.query.filter_by(id=id).first()
    db.session.delete(emp_todel)
    db.session.commit()
    return render_template("ack2.html")
```



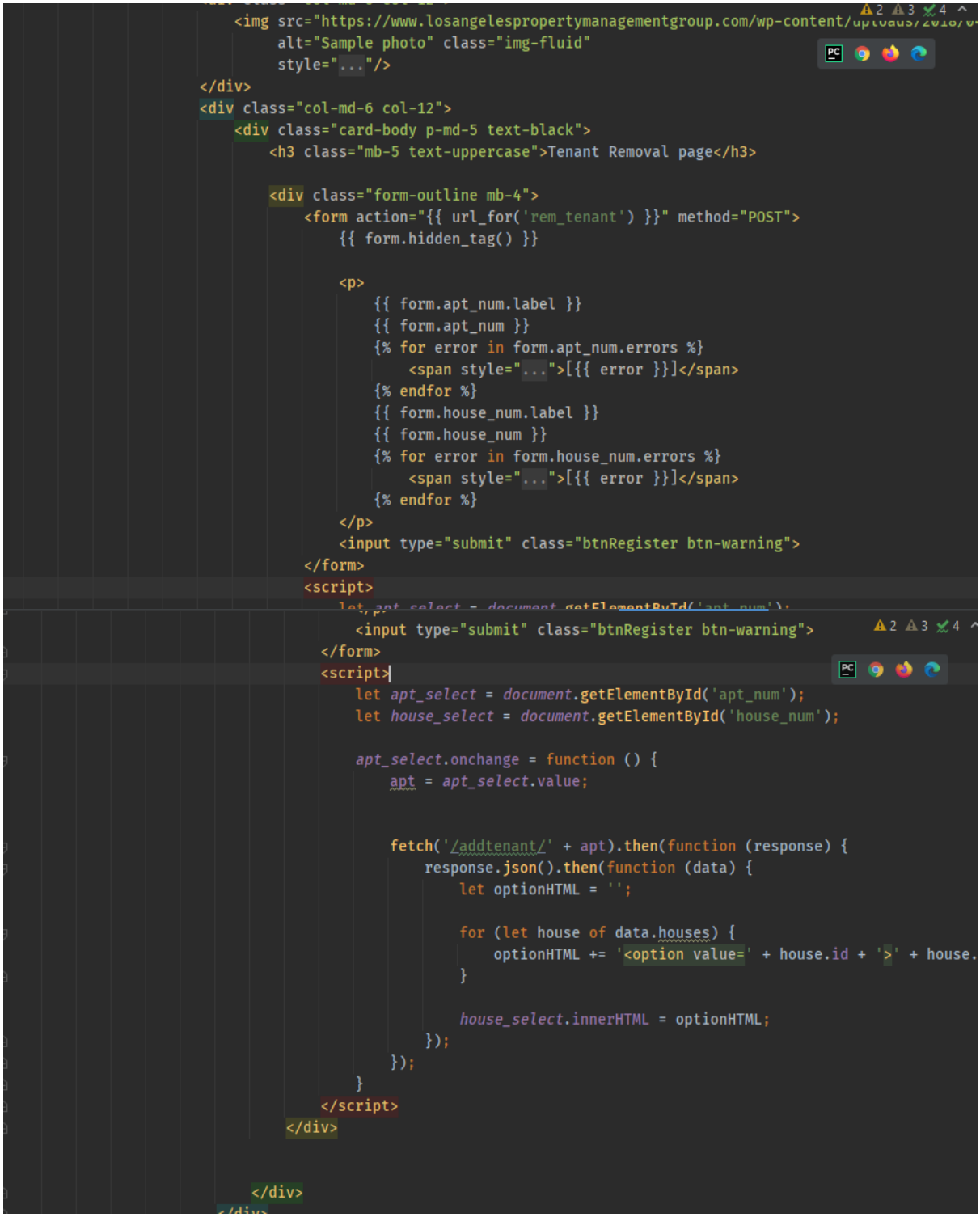
Output



Tenant Removal Page

Code

Front-End



```
    <img src="https://www.losangelespropertymanagementgroup.com/wp-content/uploads/2018/0
         alt="Sample photo" class="img-fluid"
         style="..."/>
</div>
<div class="col-md-6 col-12">
    <div class="card-body p-md-5 text-black">
        <h3 class="mb-5 text-uppercase">Tenant Removal page</h3>

        <div class="form-outline mb-4">
            <form action="{{ url_for('rem_tenant') }}" method="POST">
                {{ form.hidden_tag() }}

                <p>
                    {{ form.apt_num.label }}
                    {{ form.apt_num }}
                    {% for error in form.apt_num.errors %}
                        <span style="...">[{{ error }}]</span>
                    {% endfor %}
                    {{ form.house_num.label }}
                    {{ form.house_num }}
                    {% for error in form.house_num.errors %}
                        <span style="...">[{{ error }}]</span>
                    {% endfor %}
                </p>
                <input type="submit" class="btnRegister btn-warning">
            </form>
            <script>
                let apt_select = document.getElementById('apt_num');
                <input type="submit" class="btnRegister btn-warning">
            </form>
            <script>
                let apt_select = document.getElementById('apt_num');
                let house_select = document.getElementById('house_num');

                apt_select.onchange = function () {
                    apt = apt_select.value;


                    fetch('/addtenant/' + apt).then(function (response) {
                        response.json().then(function (data) {
                            let optionHTML = '';

                            for (let house of data.houses) {
                                optionHTML += '<option value=' + house.id + '>' + house.
                            }

                            house_select.innerHTML = optionHTML;
                        });
                    });
                }
            </script>
        </div>


    </div>
</div>
```

Back-End:

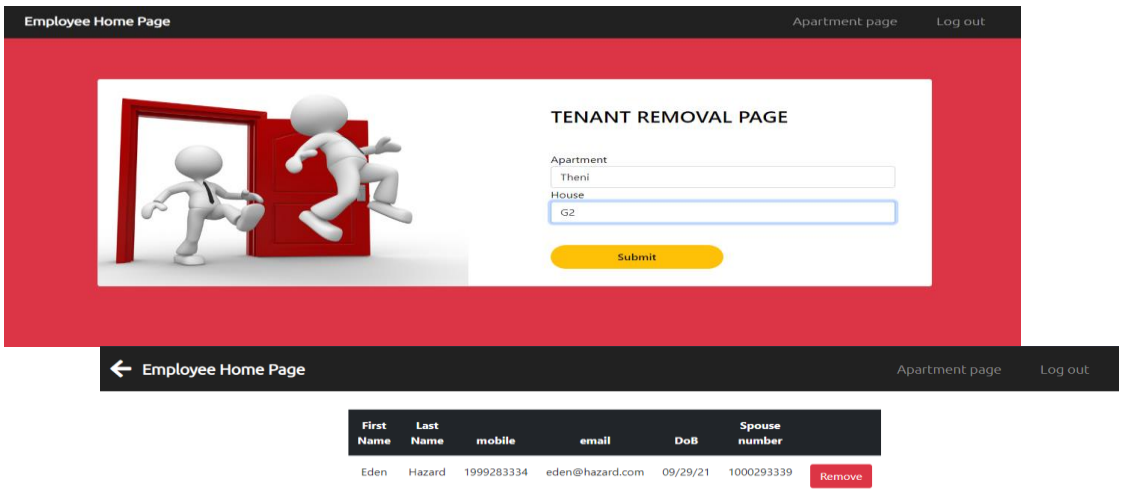
Routing

```
@app.route('/remtenant', methods=['GET', 'POST'])
def rem_tenant():
    form = TenantRemoveForm()
    if form.validate_on_submit():
        id = form.house_num.data
        headings = ("First Name", "Last Name", "mobile", "email", "DoB", "Spouse number ", " ")
        tens = Tenant.query.filter_by(house_id=id).all()
        return render_template('ten_rlist.html', headings=headings, data=tens)
    return render_template('tenants_rm.html', form=form)


@app.route('/remhouses', methods=['GET', 'POST'])
```

Deletion

```
@app.route('/delete_ten/<int:id>')
def delete_ten(id):
    ten_todel = Tenant.query.filter_by(id=id).first()
    db.session.delete(ten_todel)
    db.session.commit()
    return render_template("ack2.html")


@app.route('/delete_house/<int:id>')
```

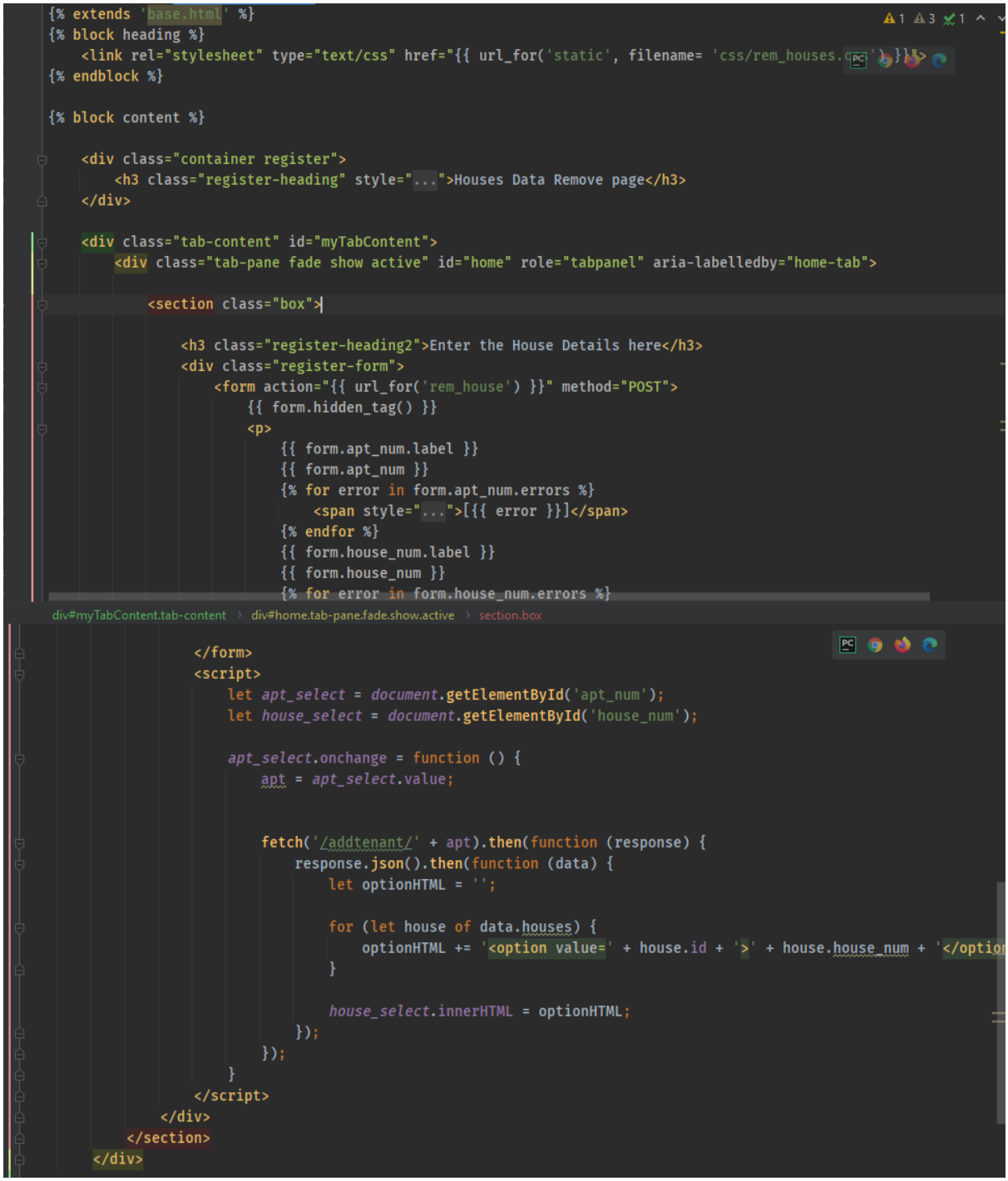
Output:



House Removal Page

Code:

Front-End:



```
{% extends 'base.html' %}
{% block heading %}
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/rem_houses.css') }}">
{% endblock %}

{% block content %}

    <div class="container register">
        <h3 class="register-heading" style="...">Houses Data Remove page</h3>
    </div>

    <div class="tab-content" id="myTabContent">
        <div class="tab-pane fade show active" id="home" role="tabpanel" aria-labelledby="home-tab">

            <section class="box">

                <h3 class="register-heading2">Enter the House Details here</h3>
                <div class="register-form">
                    <form action="{{ url_for('rem_house') }}" method="POST">
                        {{ form.hidden_tag() }}
                        <p>
                            {{ form.apt_num.label }}
                            {{ form.apt_num }}
                            {% for error in form.apt_num.errors %}
                                <span style="...">[{{ error }}]</span>
                            {% endfor %}
                            {{ form.house_num.label }}
                            {{ form.house_num }}
                            {% for error in form.house_num.errors %}
```

div#myTabContent.tab-content › div#home.tab-pane.fade.show.active › section.box

```
                </form>
                <script>
                    let apt_select = document.getElementById('apt_num');
                    let house_select = document.getElementById('house_num');

                    apt_select.onchange = function () {
                        apt = apt_select.value;


                        fetch('/addtenant/' + apt).then(function (response) {
                            response.json().then(function (data) {
                                let optionHTML = '';

                                for (let house of data.houses) {
                                    optionHTML += '<option value=' + house.id + '>' + house.house_num + '</optio
                                }

                                house_select.innerHTML = optionHTML;
                            });
                        });
                    }
                </script>
            </div>
        </section>
    </div>
```

Back-End:

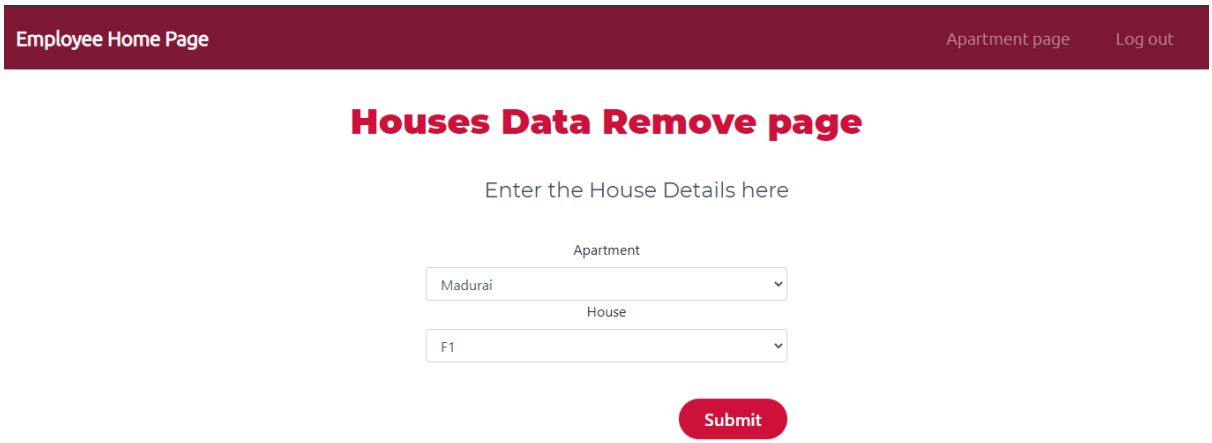
Routing

```
@app.route('/remhouses', methods=['GET', 'POST'])
def rem_house():
    form = TenantRemoveForm()
    form.house_num.choices = [(house.id, house.house_num) for house in House.query.filter_by(apt_id=1).all()]
    if form.validate_on_submit():
        id = form.house_num.data
        house_todel = House.query.filter_by(id=id).first()
        db.session.delete(house_todel)
        db.session.commit()
        return render_template("ack2.html")
    return render_template('rem_house.html', form=form)
```

Deletion

```
@app.route('/delete_house/<int:id>')
def delete_house(id):
    house_todel = House.query.filter_by(id=id).first()
    db.session.delete(house_todel)
    db.session.commit()
    return render_template("ack2.html")
```

Output:



Apartment Information Page

Code

Front-End

```
<!doctype html>
<html>

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- font letter-->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@1,100&display=swap" rel="stylesheet"
  <link rel="stylesheet" href="{{ url_for('static', filename= 'css/STYLE.css') }}">

  <!-- font icons -->
  <script src="https://use.fontawesome.com/ae44ae53a6.js"></script>

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" integrit
  <link rel="stylesheet" href="C:\web development\Bootstrap-Installation\styles.css">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-U

  <!-- Option 2: Separate Popper and Bootstrap JS -->

  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js" integrity="sha384-eMNCO
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.min.js" integrity="sha384-cn7l7gDp
  <title>MORE&INFO</title>
</head>
```

```
<body>
  <!-- Optional JavaScript; choose one of the two! -->

  <!-- Option 1: Bootstrap Bundle with Popper -->

  <section class="colored-section" id="title">

    <div class="container-fluid">



      <nav class="navbar navbar-expand-lg navbar-dark">
        <a class="navbar-brand" href="">MORE&INFO</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedConte
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav ms-auto">
            <li class="nav-item">
              <a class="nav-link" href="#footer">HOMEPAGE</a>
            </li>

          </ul>
        </div>
      </nav>
```

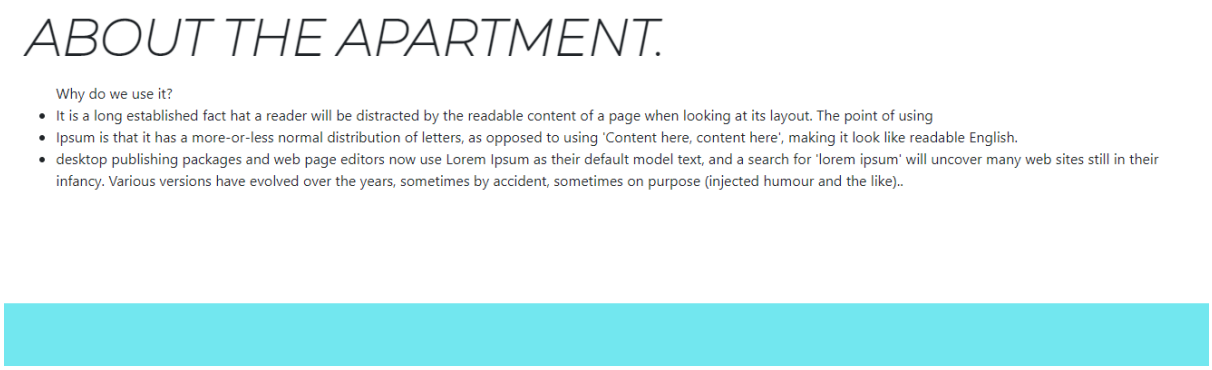
```
<section class="white-section" id="features">
  <div class="container-fluid">
    <div class="feature-box">

      <h3 class="feature-title">ABOUT THE APARTMENT.</h3>
      <p class="feature-text">
      <ul>Why do we use it?
        <li>It is a long established fact hat a reader will be distracted by the readable content of a page when
          The point of using </li>
        <li> Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content her
          making it look like readable English.</li>
        <li>desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and
          by accident, sometimes on purpose (injected humour and the like)..</li>
      </ul>
      </p>
    </div>


  </div>

  </div>
</section>

<section class="colored-section" id="testimonials">
  <div id="testimonial-carousel" class="carousel slide" data-ride="false">
```

Back-end:

Routing

```
@app.route('/moreinfo')
def more_info():
    return render_template("more_info.html")
```

Output:



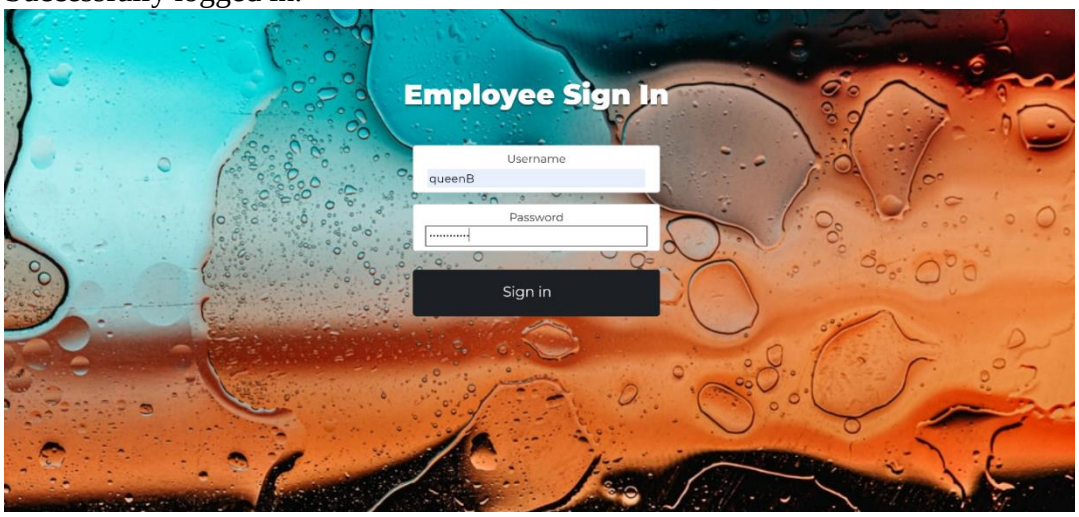
UIT1511---Software Design Lab

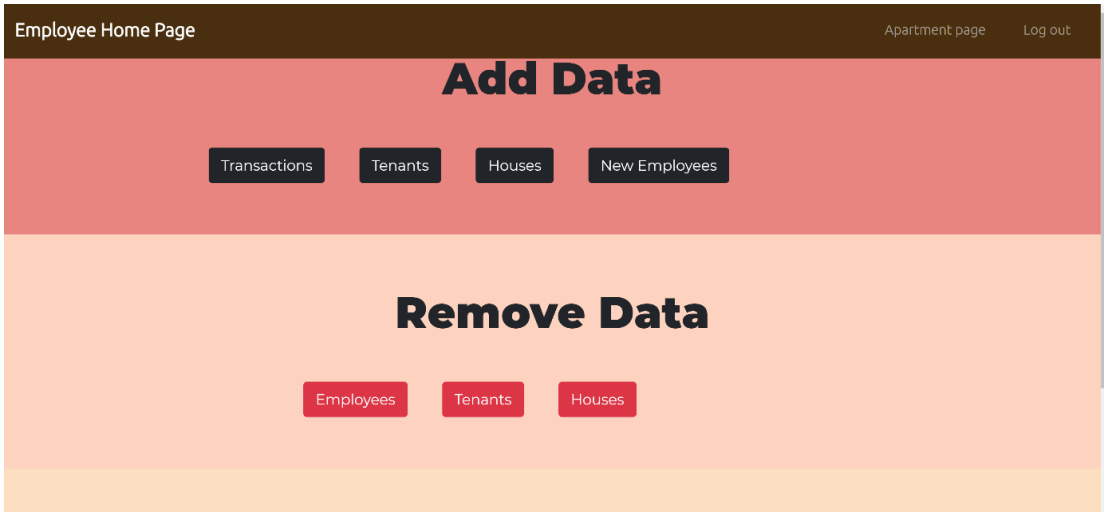
Test Case Analysis
for Real Estate Management System

-Team Pigeons

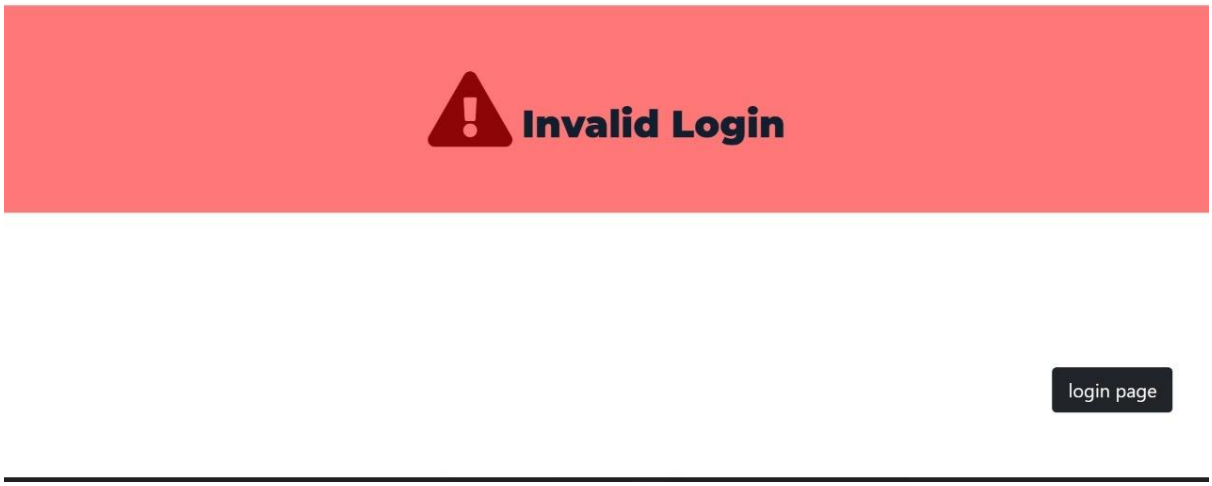
Login Module:

Input: Username and password
Explanation: This module takes two inputs i.e Username and password as input. The backend checks if the credentials are correct if so it successfully navigates to the employee home page.
If not it shows error
Output:
Successfully logged in.





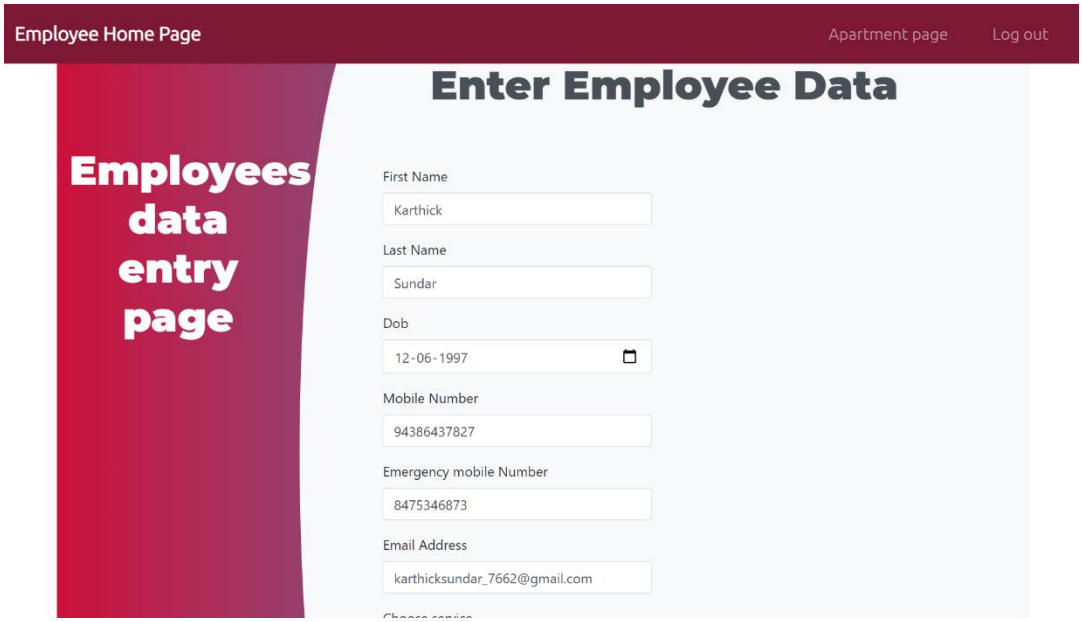
Shows error on invalid credentials:



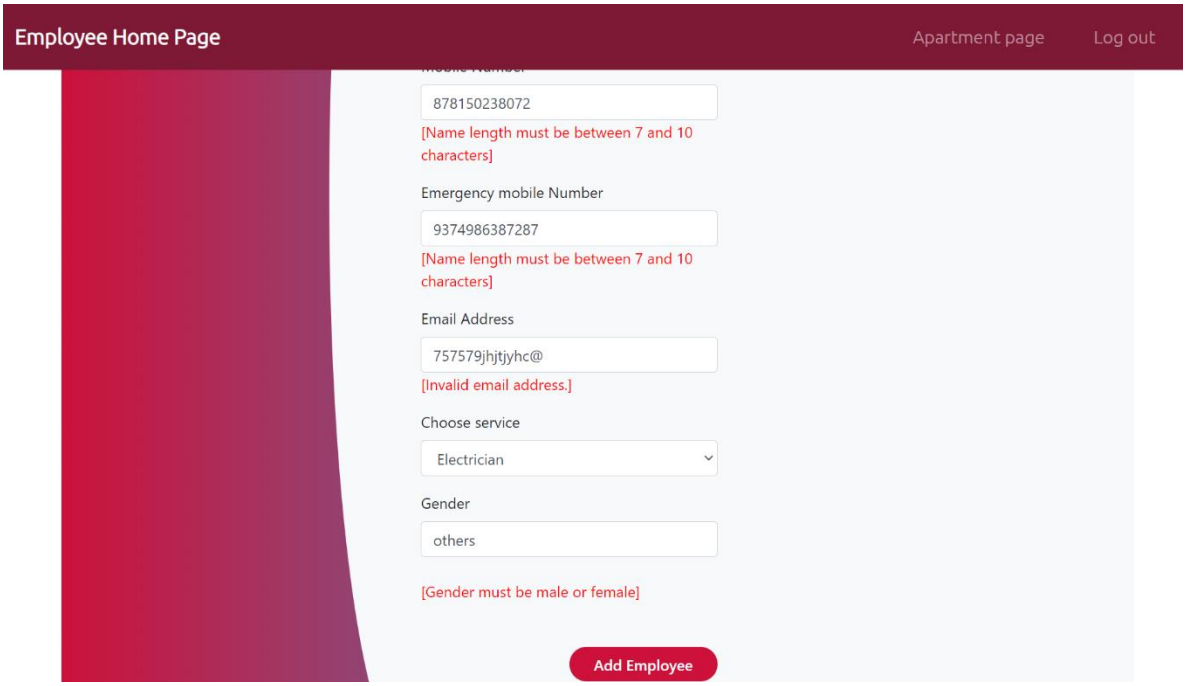
Employee entry page:

Input: need enter all details required for an employee to register.
Explanation: On entering all the details the backend checks if the entered data is in correct format it accepts the input and successfully navigates to the acknowledgement page.
Output:

On entering correct format of data:



On entering data in incorrect format:

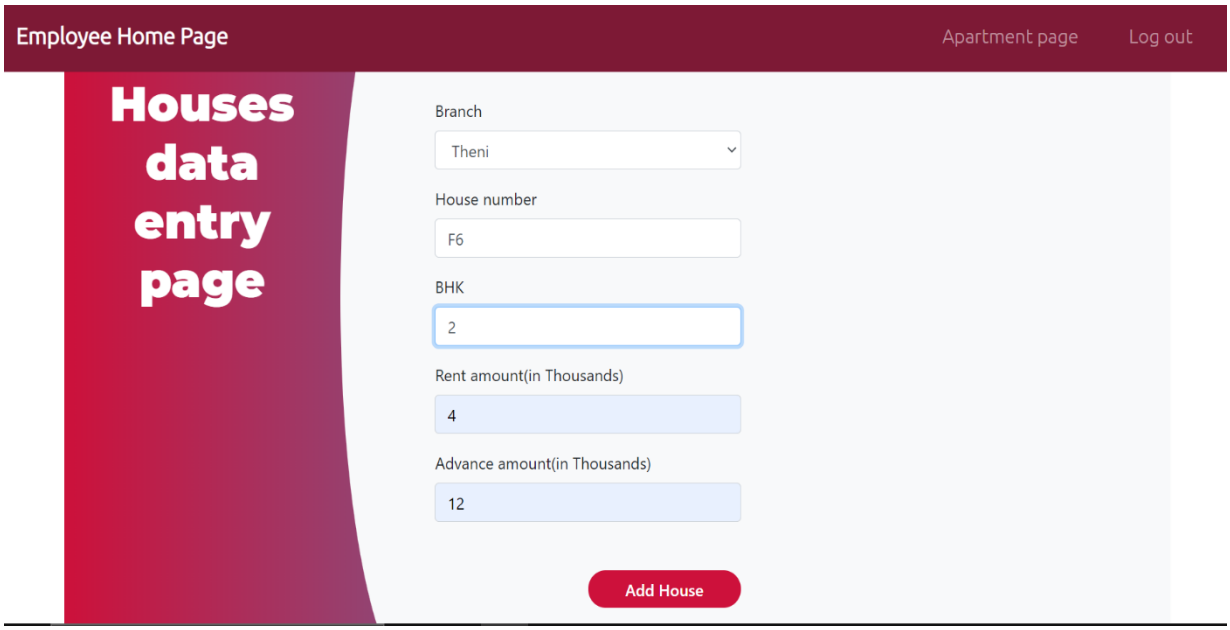


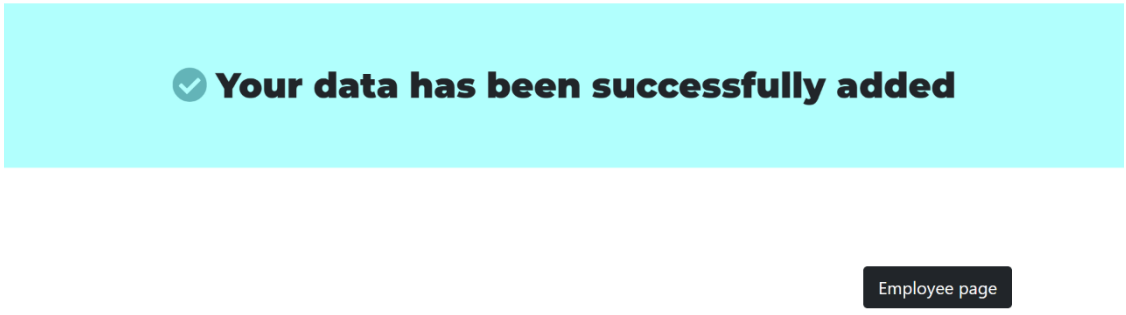
Houses entry page:

Input: need enter all details required to add a house.
Explanation: On entering all the details the backend checks if the entered data is in correct format it accepts the input and successfully navigates to the acknowledgement page.

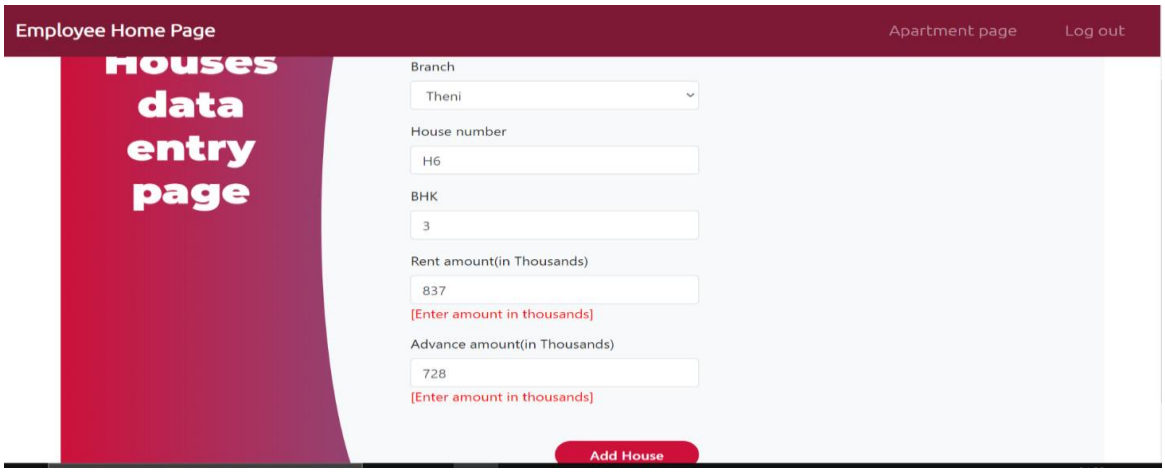
Output:

On entering correct format of data:





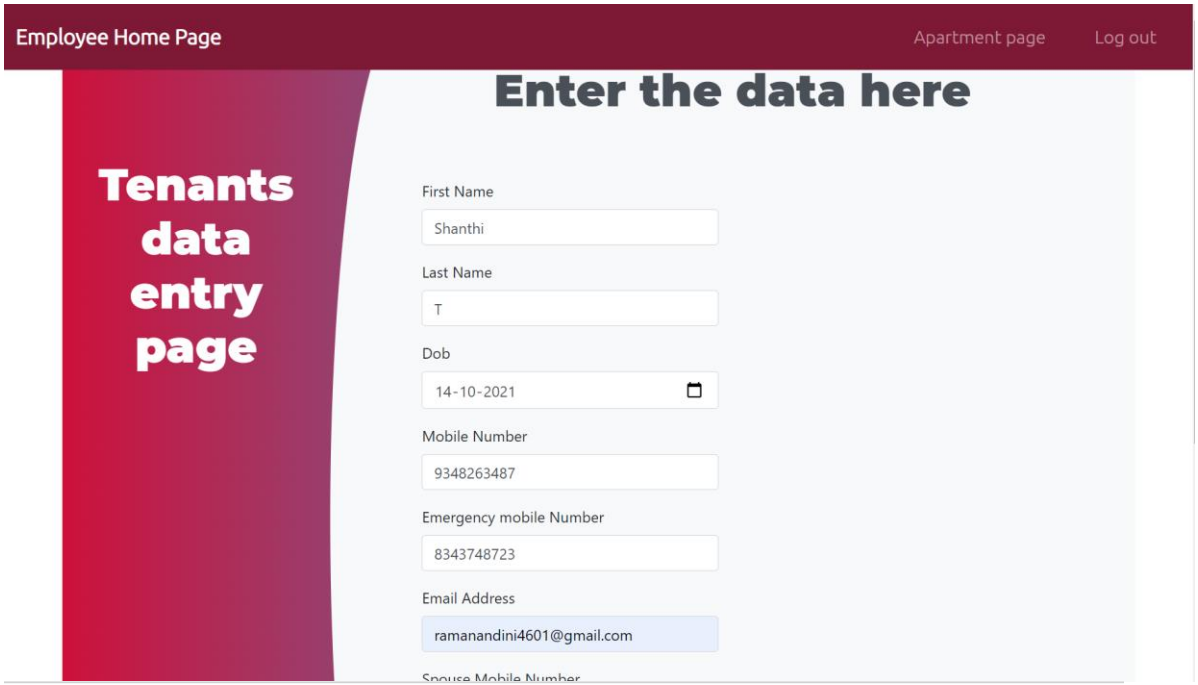
On entering data in incorrect format:

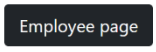


Tenants entry page:

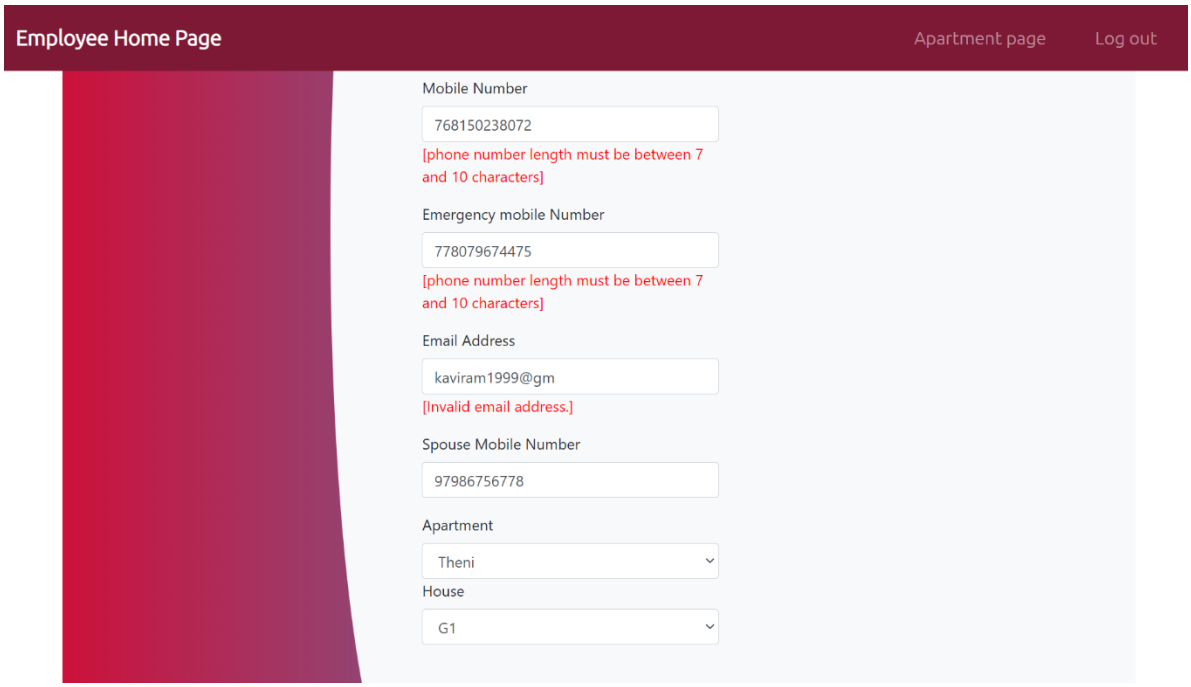
Input: need enter all details required for a tenant to register.
Explanation: On entering all the details the backend checks if the entered data is in correct format it accepts the input and successfully navigates to the acknowledgement page.
Output:

On entering correct format of data:





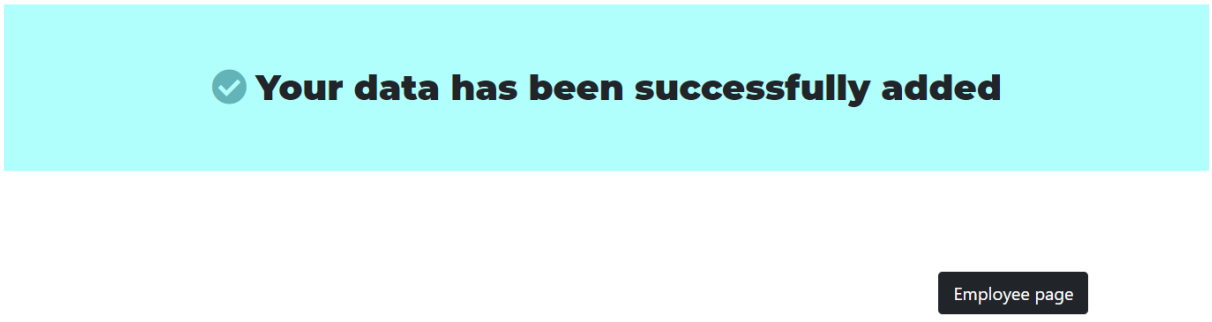
On entering data in incorrect format:



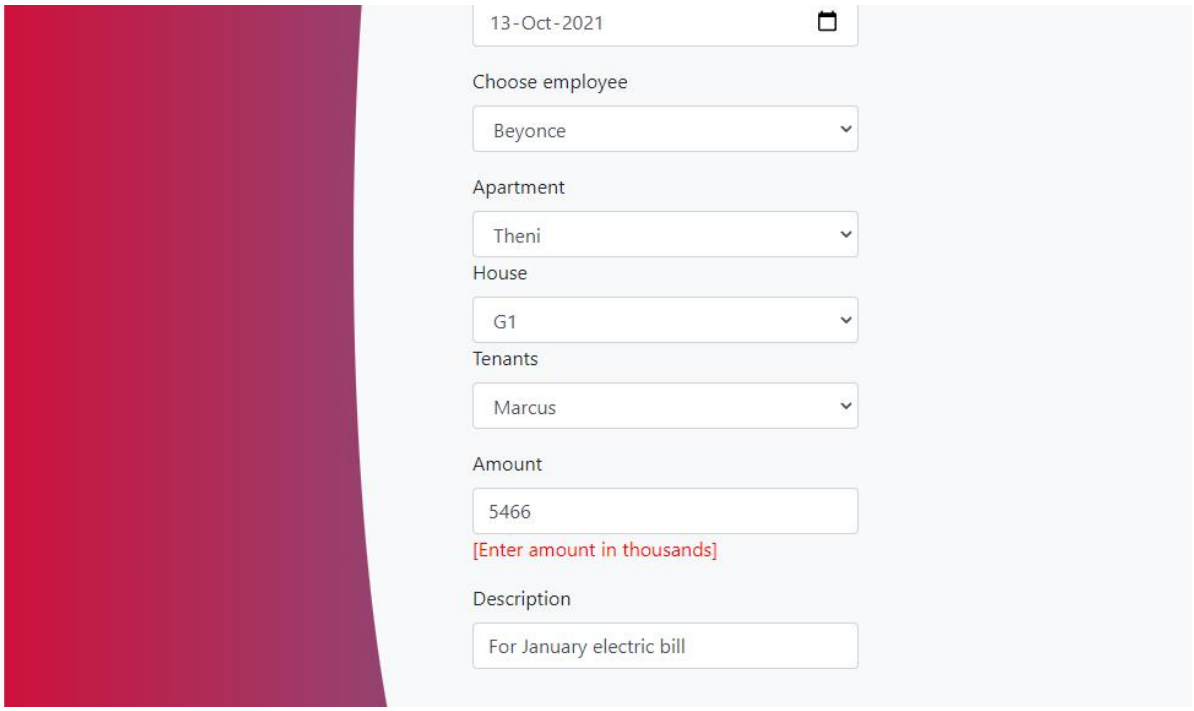
Transactions entry page:

Input: Enter all details required for a Transaction to be registered.
Explanation: On entering all the details the backend checks if the entered data is in correct format it accepts the input and successfully navigates to the acknowledgement page.
Output:

On entering correct format of data:



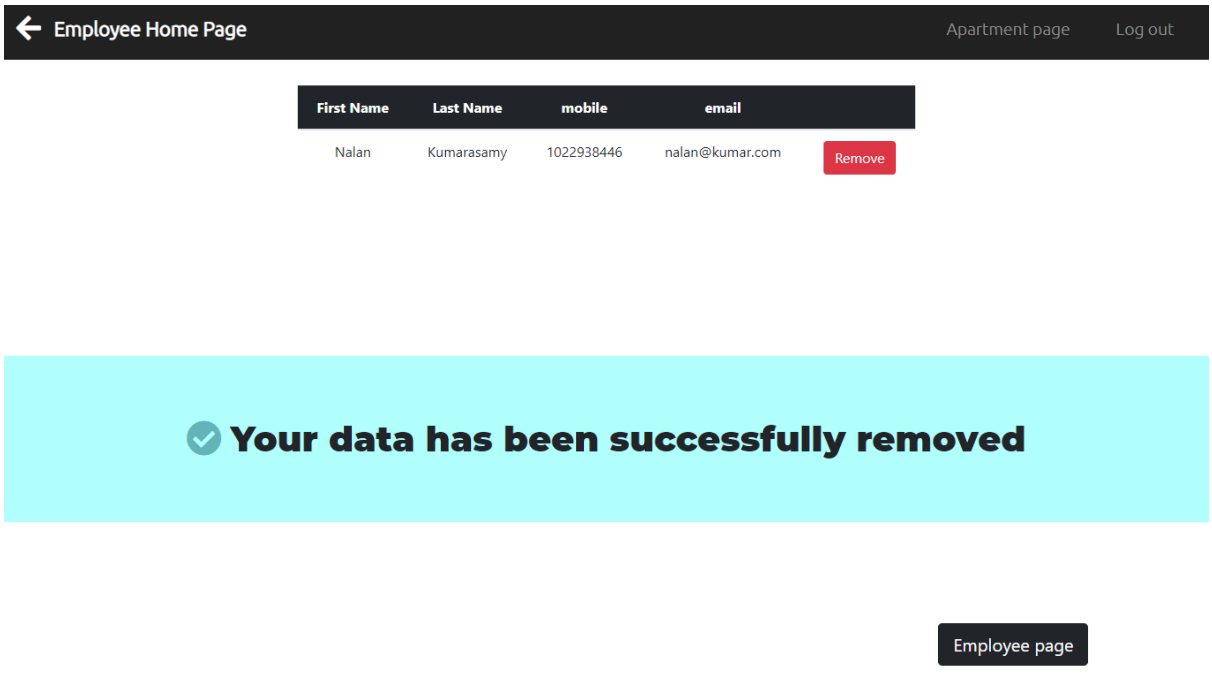
On entering data in incorrect format:



Employee Removal page:

Input: Select the employee record to remove.
Explanation: The selected employee record will be removed from the database.

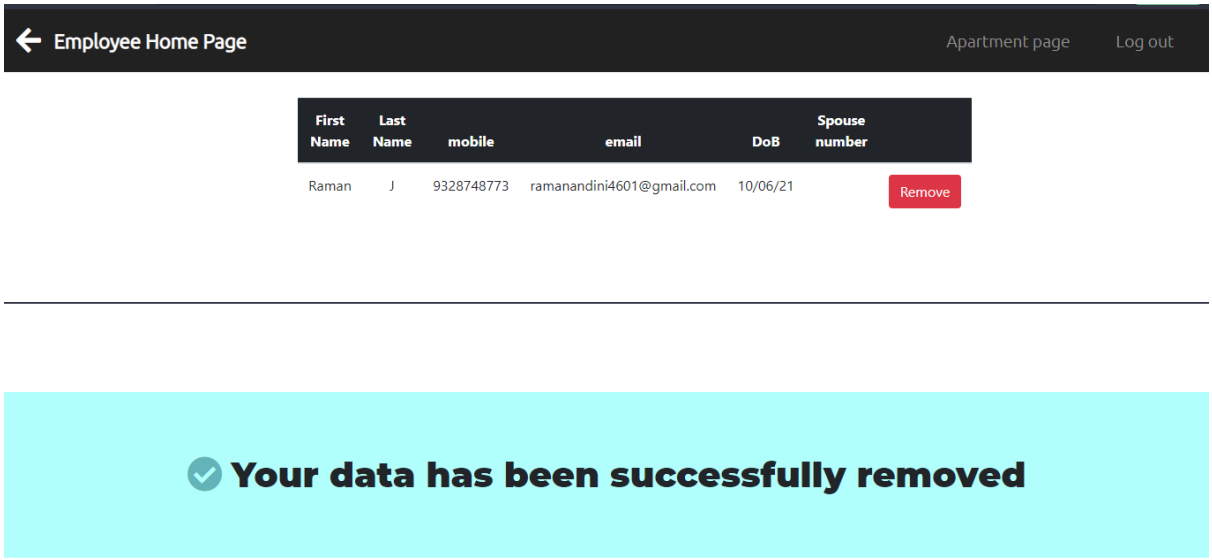
On successful deletion:



Tenant Removal Page:

Input: Select the desired tenant record to be deleted.
Explanation: The selected tenant record will be deleted from the database.
Output:

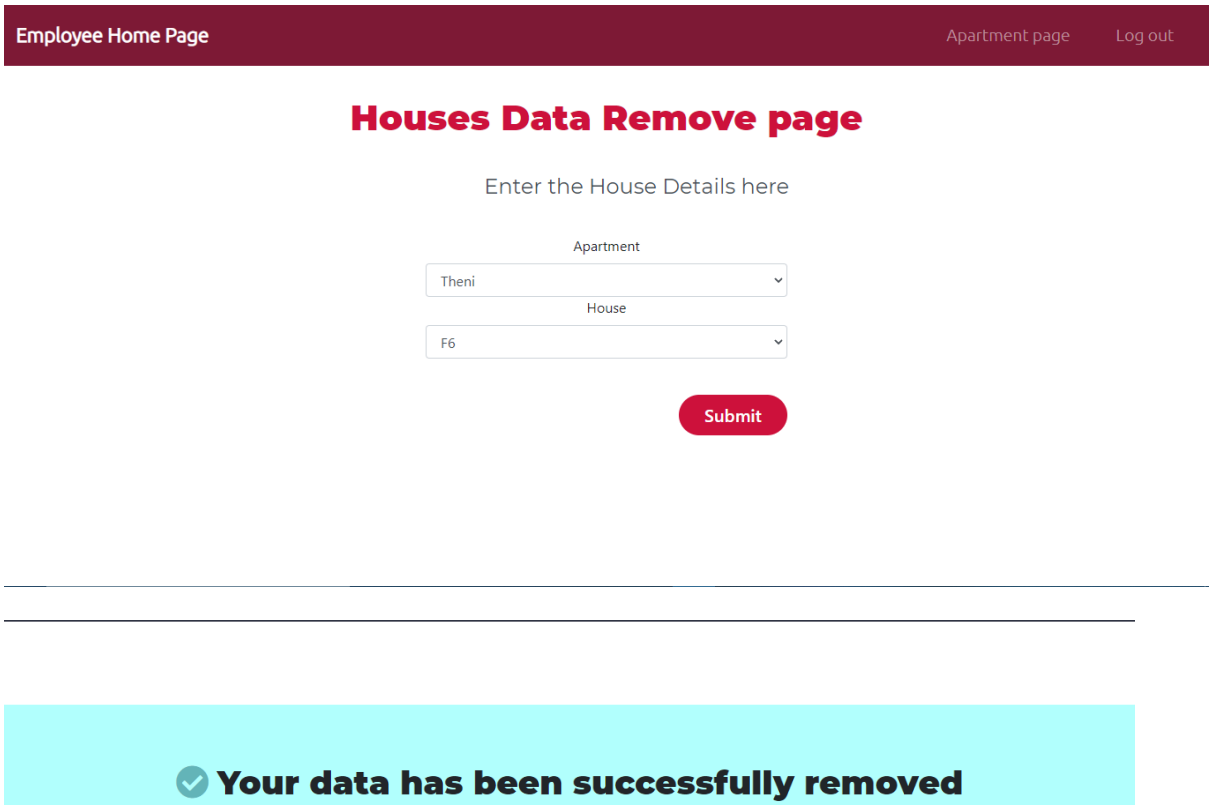
Upon successful deletion:

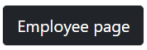


House Removal Page:

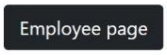
Input: Select the desired house record to be deleted
Explanation: The selected house record will be deleted from the database.
Output:

Upon successful deletion :





Invalid Removal:



The above scenario is caused when a house which has pre existing tenants is removed.

Client Evaluation Report

29th October 2021,
Chennai, Tamil Nadu

Shree Lalithambigai Apartments

To whomsoever it may concern,

This document is to certify that the below students from SSN College of Engineering, Kalavakkam - 603 110, Chennai have successfully completed and delivered the promised project upto its discussed and agreed upon specifications.

1. V.Meganathan
2. S.Navaneetha Krishnan
3. Nandini R
4. Alagappan S

Lalithambigai Apartments is an apartment building, privately owned by Mr . R V Vijayakumar, who's seeking to actively expand into a housing chain . The above students from SSN College of Engineering, put us closer to that vision, by delivering a Real Estate Management System, which abstracts and simplifies the record storage of apartment details without compromising on the quality of data .

The product also doubles as a worthy advertisement page, for future tenants or investors for the Lalithambigai Apartments housing chain. I am extremely satisfied with the product that was delivered by the students, and wish them all well for their future.

Ratings for different Features:

Ratings vary from, Excellent, Good, satisfactory, and unsatisfactory.

- *Apartment advertisement page:* **Excellent.** All the features and highlights of the apartments can be listed in an eye-catching manner.
- *Multiple User functionality :* **Excellent.** Completely Functional
- *Data addition Pages :* **Good.** There could be more customisable options, like the capability of adding more buildings.

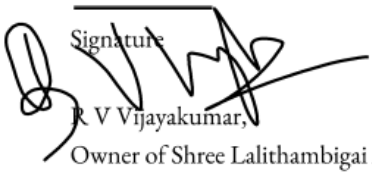
29th October 2021,
Chennai, Tamil Nadu

- *Data Removal and Viewing Pages:* **Good.**It was a smart idea to combine data removal and viewing pages together. Scenario of tenant removal when a house is deleted had been overlooked.
- *UI:* **Excellent.** Simple UI that is easy and intuitive to use.
- *Buffer time:* **Good.** Buffer time is reasonable in most cases.
- Overall Project: **Good.**

The students showed a real knack of professionalism and a good understanding of the problems faced by the everyday user during the development of this product. Their reactions to hurdles and way of communication with the client was impressive and showed great promise.

Throughout the duration of the project, they were disciplined, dedicated and never faltered on deadlines. They maintained a healthy work ethic and kept good time on promised deliverables.

I am satisfied with the product and am happy to accept it on behalf of Lalithambigai Apartments.



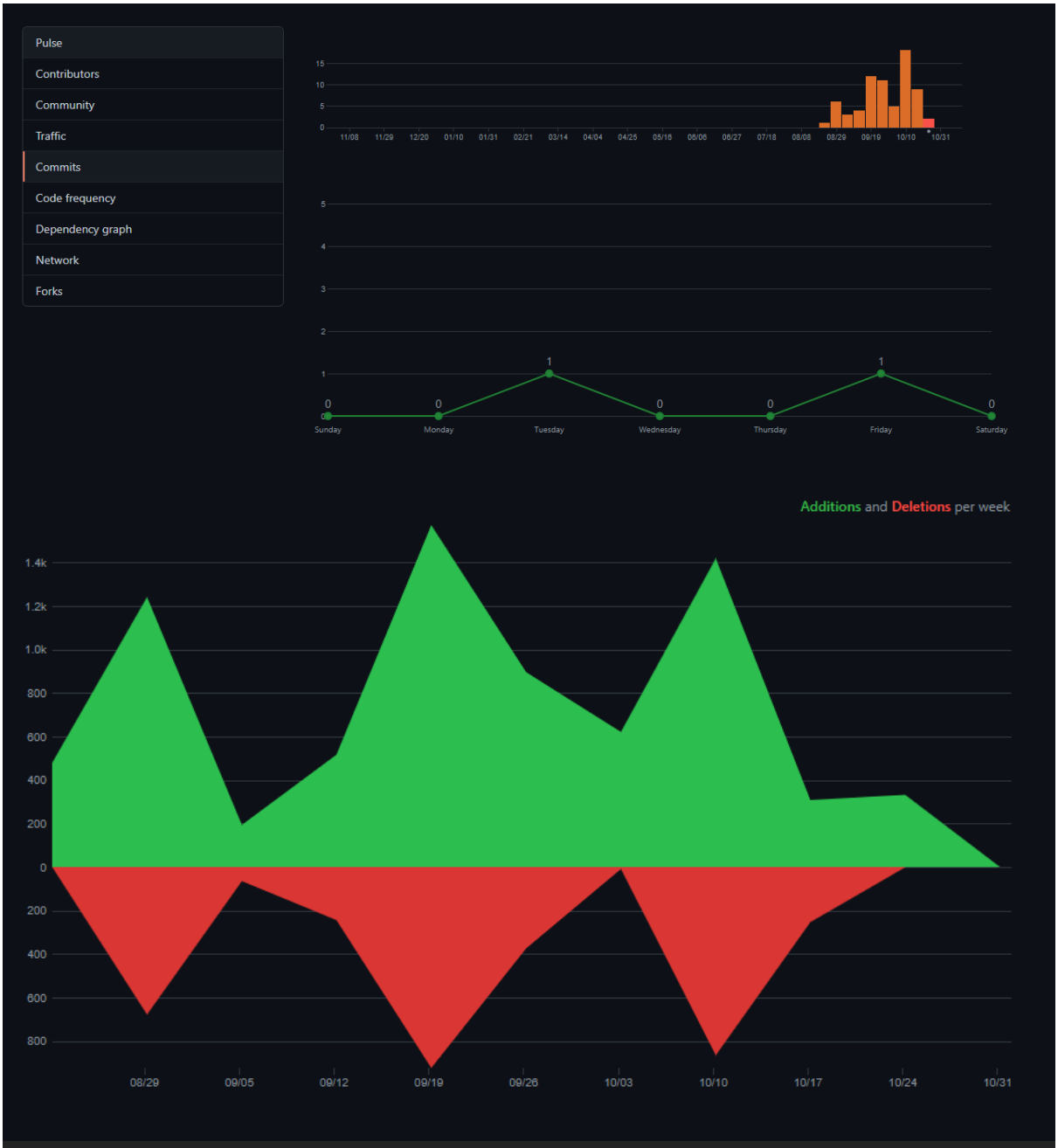
Signature

R V Vijayakumar,
Owner of Shree Lalithambigai Apartments,
Email id : vijairv@gmail.com

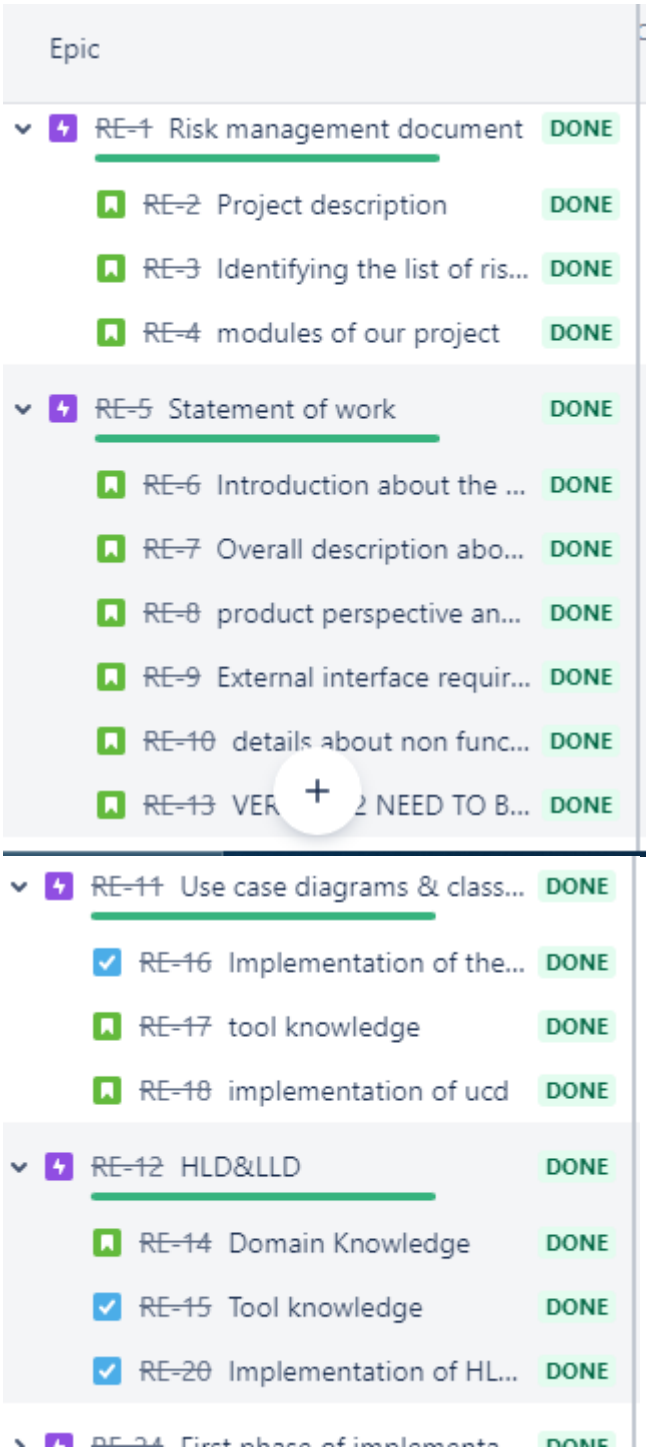
Appendix

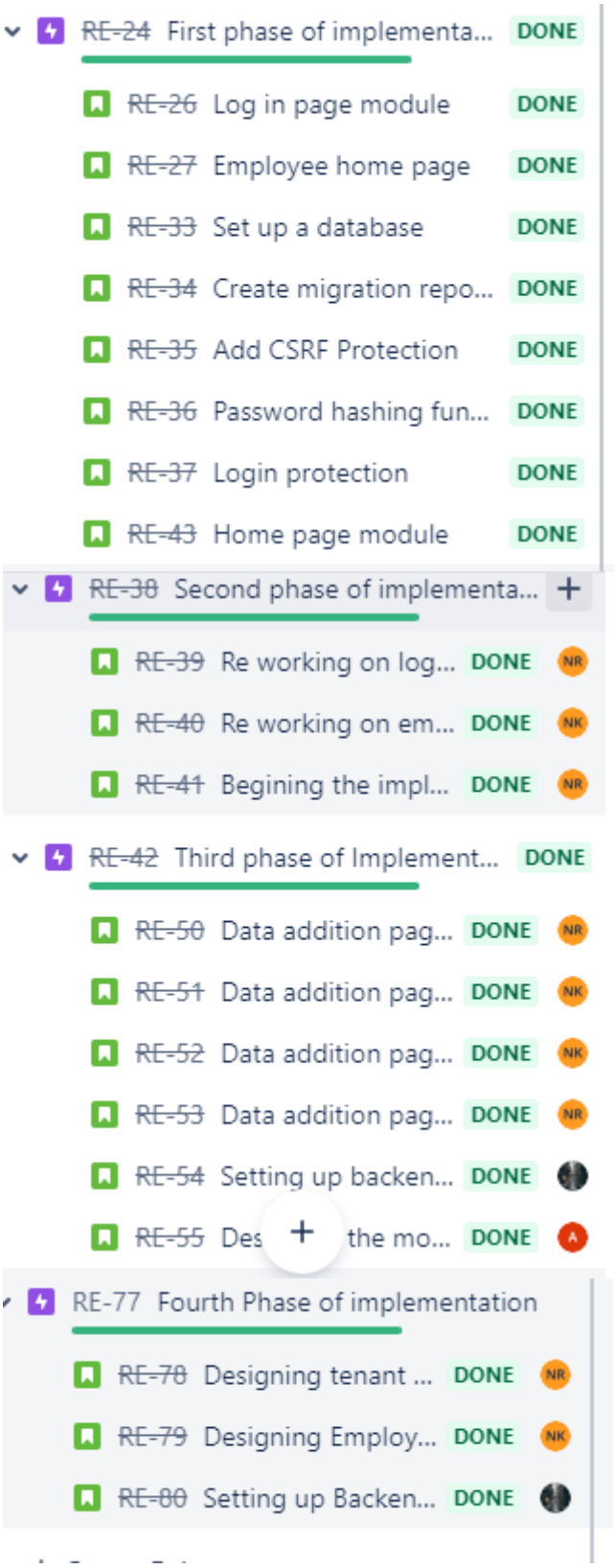
Jira methodology was rigorously followed in the implementation of the project.
Regular meetings were held everyday. All meetings were documented.
All members were made aware of the other person's role in the spirit.
All planning of future sprints and all project decisions were taken with the inputs of all team members.
Client was regularly updated during the project development. Changes were made after every suggestion made by the client.

Workflow is updated in JIRA.



Epic	JAN '22	FEB '22	MAR
~~RE-1~~ Risk management document DONE			
~~RE-5~~ Statement of work DONE			
~~RE-11~~ Use case diagrams & class diagrams DONE			
~~RE-12~~ HLD&LLD DONE			
~~RE-24~~ First phase of implementation DONE			
~~RE-38~~ Second phase of implementation DONE			
~~RE-42~~ Third phase of Implementation DONE			
RE-77 Fourth Phase of implementation			


Epic
RE-1 Risk management document DONE
RE-2 Project description DONE
RE-3 Identifying the list of ris... DONE
RE-4 modules of our project DONE
RE-5 Statement of work DONE
RE-6 Introduction about the ... DONE
RE-7 Overall description abo... DONE
RE-8 product perspective an... DONE
RE-9 External interface requir... DONE
RE-10 details about non func... DONE
RE-13 VER 2 NEED TO B... DONE
RE-11 Use case diagrams & class... DONE
RE-16 Implementation of the... DONE
RE-17 tool knowledge DONE
RE-18 implementation of ucd DONE
RE-12 HLD&LLD DONE
RE-14 Domain Knowledge DONE
RE-15 Tool knowledge DONE
RE-20 Implementation of HL... DONE



Minutes of Meetings

Sri Sivasubramaniya Nadar College of Engineering,
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Information Technology,
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110.

26-07-2021

Minutes of the First Online SD lab Team Meeting

Class: V Sem IT **Meeting held on:** 26-07-2021

Academic Year: 2021-22 **Batch:** 2019-2023

The following members were present for the Team meeting:
Team Members

S.NO	NAME	ROLE	ATTENDED
1	V.MEGANATHAN	PRODUCT MANAGER	YES
2	ALAGAPPAN.S	DEVELOPER	YES
3	NANDINI.R	SCRUM MASTER	YES
4	NAVANEETHA KRISHNAN.S	DEVELOPER	YES

POINTS DISCUSSED IN THIS MEETING ARE:
*Brief about the project.
*Meganathan finalised who is the client and what he demands in the project.
*In the project how to data entry and how it works.
*Diagram and create the circuit.
*Risk management created by the team.
* It was Discussion about modules.
*How to create the link between apartment and project and rules and regulations.
*Discussion about vacancy and data entry page and experiment page and report generation page.
*JIRA created the team meet how.

Sri Sivasubramaniya Nadar College of Engineering,
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Information Technology,
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110.

Minutes of the online SD lab team meeting

CLASS:V SEM IT ***Meeting held on:31-07-2020***

ACADEMIC YEAR:2021-22
BATCH:2019-2023

The following members were present for Team meeting:

TEAM MEMBERS

S.NO	NAME	ROLE	ATTENDED
1	V.MEGANATHAN	PRODUCT MANAGER	YES
2	ALAGAPPAN.S	DEVELOPER	YES
3	NANDINI.R	SCRUM MASTER	YES
4	NAVANEETHA KRISHNAN.S	DEVELOPER	YES

POINTS DISCUSSED IN THIS MEETING ARE:
*Brief about the project.
*Meganathan finalised who the client is and what he demands in the project.
*In the project how to data entry and how it works.
*Diagram and create the circuit.
*Risk management created by the team.
* It was Discussion about modules.
*How to create the link between apartment and project and rules and regulations.
*Discussion about vacancy and data entry page and experiment page and report generation page.
*JIRA created the team meet how.

Sri Sivasubramaniya Nadar College of Engineering,
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Information Technology,
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110.

Minutes of the online SD lab team meeting

CLASS:V SEM IT ***Meeting held on:02-08-2020***

ACADEMIC YEAR:2021-22
BATCH:2019-2023

The following members were present for Team meeting:

TEAM MEMBERS

S.NO	NAME	ROLE	ATTENDED
1	V.MEGANATHAN	PRODUCT MANAGER	YES
2	ALAGAPPAN.S	DEVELOPER	YES
3	NANDINI.R	SCRUM MASTER	YES
4	NAVANEETHA KRISHNAN.S	DEVELOPER	YES

POINTS DISCUSSED IN THIS MEETING ARE:
*By the team change the correction in SOW
*It is discussion about introduction in the project
*Priority and details explanation correct the correction and discussion
*This project who can deal with client and connection between the client discussion by scrum master
*Brief about rdms ,scale data involve and product function
*what is use of sql data discussion by developer
*who can client create user manual discussion by developer
*It is discussion about assumption by developer
*What was focus to develop the project
*Brief about vacancy list

Sri Sivasubramaniya Nadar College of Engineering,
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Information Technology,
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110.

Minutes of the online SD lab team meeting

CLASS:V SEM IT ***Meeting held on:04-08-2020***

ACADEMIC YEAR:2021-22
BATCH:2019-2023

The following members were present for Team meeting:

TEAM MEMBERS

S.NO	NAME	ROLE	ATTENDED
1	V.MEGANATHAN	PRODUCT MANAGER	YES
2	ALAGAPPAN.S	DEVELOPER	YES
3	NANDINI.R	SCRUM MASTER	YES
4	NAVANEETHA KRISHNAN.S	DEVELOPER	YES

POINTS DISCUSSED IN THIS MEETING ARE:
*It is discussion about open the jira page check login id
*Use case diagram check creating by team
*It is discussion about the create issue and plantuml
*Discussion about SOW version 3 making
*Prepare for submission all documents make and date to submission
*It is discussion HLD and LLD
*JIRA create processing checking by team
*In JIRA checking project planning and domain knowledge ,tool knowledge, implementation of the Ucd
*EDIT SPIRIT and spirit goal edit by team
*In JIRA, how to make meeting note checking by team

Sri Sivasubramaniya Nadar College of Engineering,
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Information Technology,
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110.

Minutes of the online SD lab team meeting

CLASS:V SEM IT ***Meeting held on:19-08-2020***

ACADEMIC YEAR:2021-22
BATCH:2019-2023

The following members were present for Team meeting:

TEAM MEMBERS

S.NO	NAME	ROLE	ATTENDED
1	V.MEGANATHAN	PRODUCT MANAGER	YES
2	ALAGAPPAN.S	DEVELOPER	YES
3	NANDINI.R	SCRUM MASTER	YES
4	NAVANEETHA KRISHNAN.S	DEVELOPER	YES

POINTS DISCUSSED IN THIS MEETING ARE:
*Start with HLD and LLD in the meeting
*Discussion going components diagram and class diagram
*Use diagram flow details can be explain by scrum master
*Doubt are clear by the scrum master
*Details about employees given. It can clear explain by scrum master
*Discussion about level case diagram
*Relationship in use case diagram explain details
*Implementation starts processing discussion held on.

Sri Sivasubramaniya Nadar College of Engineering,
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Department of Information Technology,
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110.

Minutes of the online SD lab team meeting

CLASS:V SEM IT ***Meeting held on:26-08-2020***

ACADEMIC YEAR:2021-22
BATCH:2019-2023

The following members were present for Team meeting:

TEAM MEMBERS

S.NO	NAME	ROLE	ATTENDED
1	V.MEGANATHAN	PRODUCT MANAGER	YES
2	ALAGAPPAN.S	DEVELOPER	YES
3	NANDINI.R	SCRUM MASTER	YES
4	NAVANEETHA KRISHNAN.S	DEVELOPER	YES

POINTS DISCUSSED IN THIS MEETING ARE:
*Start with SOW error and mistake discuss
*Discussion about login page
*Discussion about Home page and Add data page
*Back end discussion held on.
*Editorial branch discuss by scrum master
*HTML rework discuss by scrum master
*Discussion about SOW update version