ASS2_BIT302_E1700882_E1700873_GroupAssignment

by Rivaldo Soepardhy

Submission date: 04-Apr-2020 11:36PM (UTC+0800)
Submission ID: 1289513869
File name: ASS2_BIT302_E1700882_E1700873_GroupAssignment_DONE.docx (29.26M)
Word count: 3404
Character count: 19672

BIT302
Software Engineering

ASSIGNMENT 2
Design & Test Document

"Web-based Information System for MicroHousing System in Kuala Lumpur"

Team Leader:

Luh Wulandari Maharani

E1700873 / 170030401

luhwulandari@gmail.com

Member:

Rivaldo Bagus Soepardhy

E1700882 / 170030400

aldobagus@hotmail.co.id

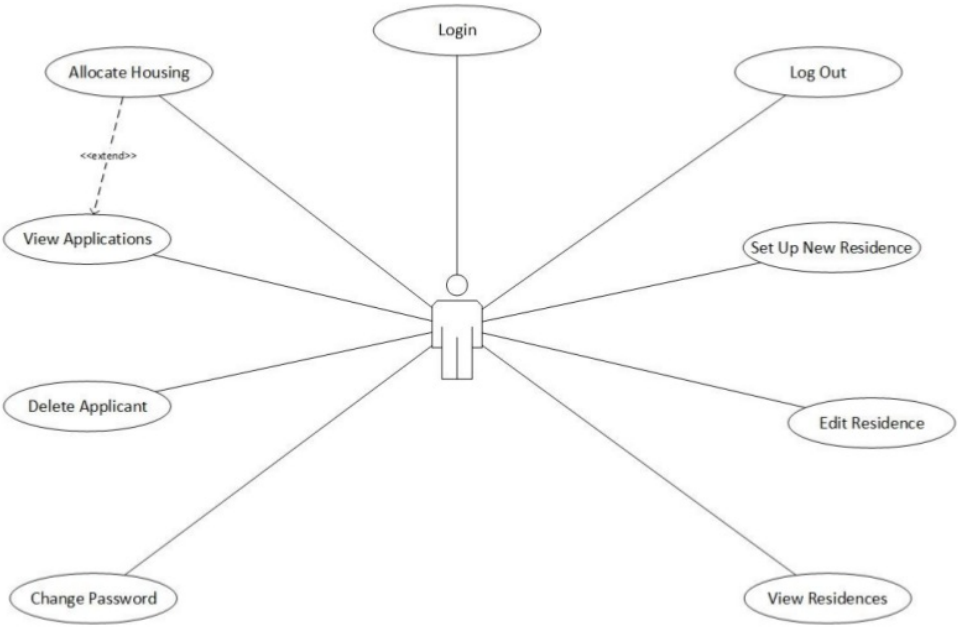
Table of Content

Introduction

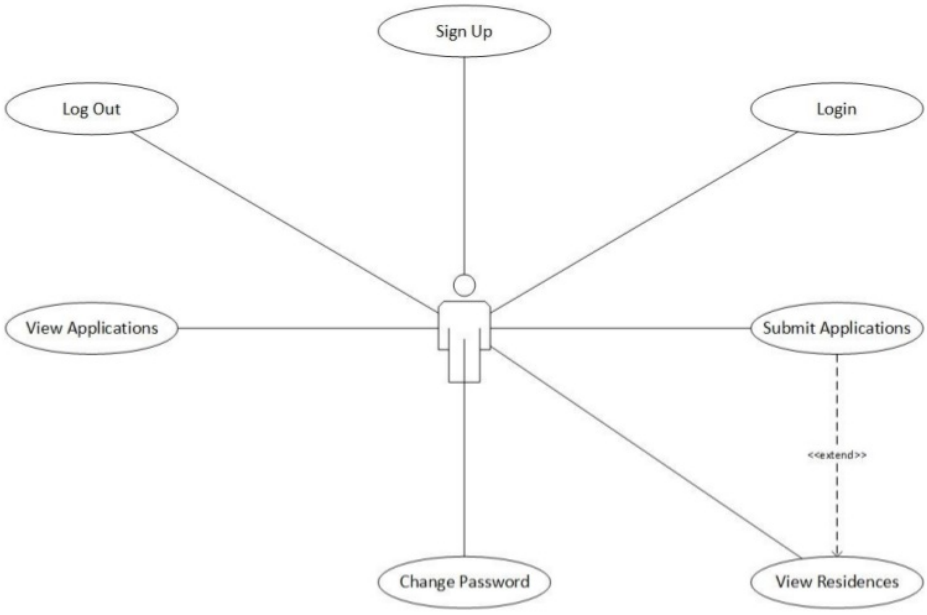
Every software development must have a clear and understandable requirement or specification to ensure developers can keep developing software within the boundary of requirement, while also, developer can adjust to what users need with the software, and developers can optimize the software according to users' need.

Previously, we already created the general requirement of what kind tools and how our web-based information system will be developed. In this document, we will give you the design specification and other deliverables regarding to the project, such as SSD, ERD, Database Design, Wireframe, etc.

Use Case Diagram

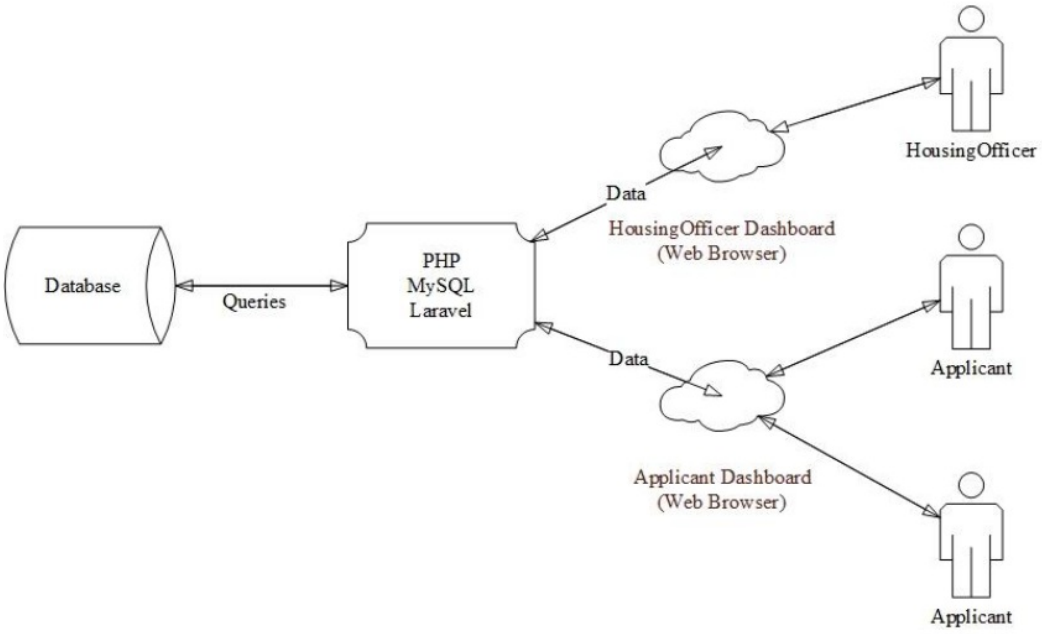


Use Case Diagram with Actor HousingOfficer

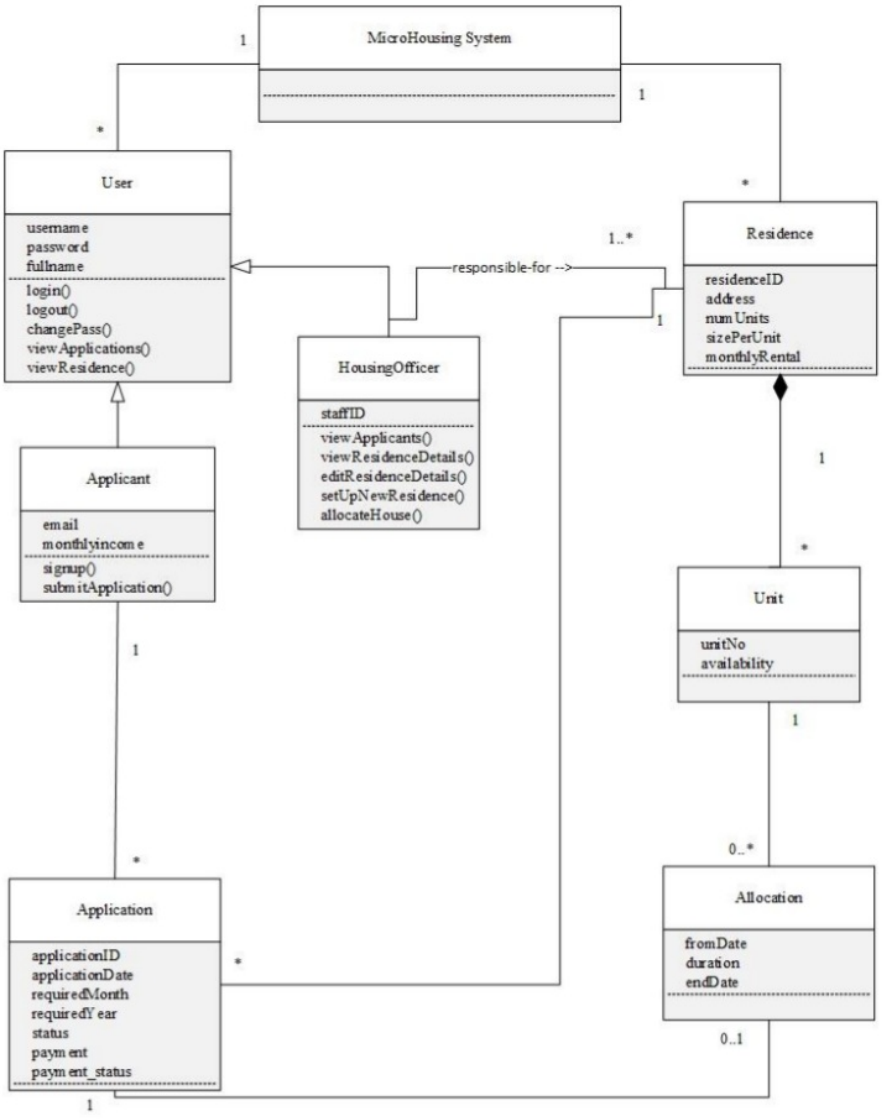


Use Case Diagram with Actor Applicant

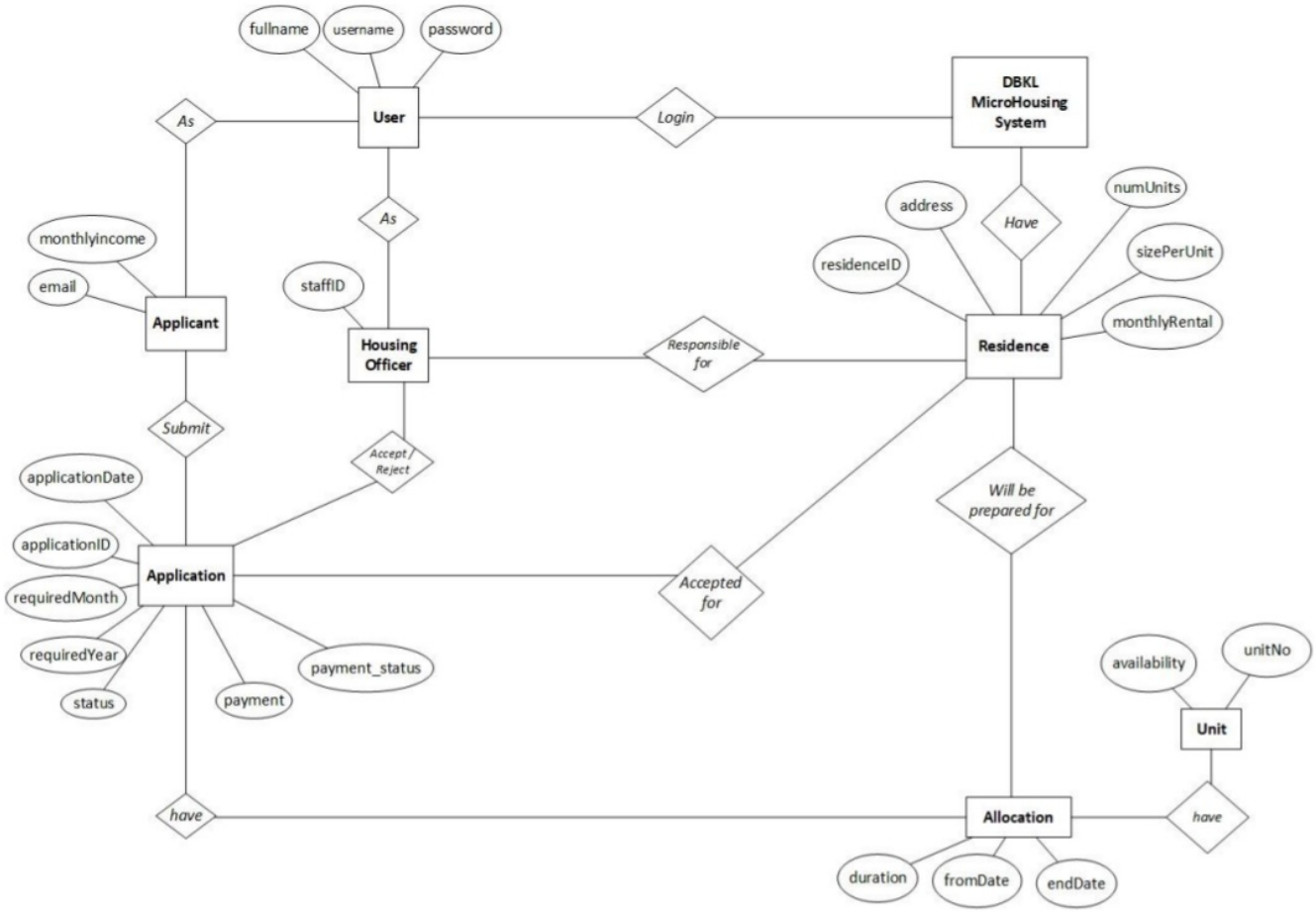
Architectural Design



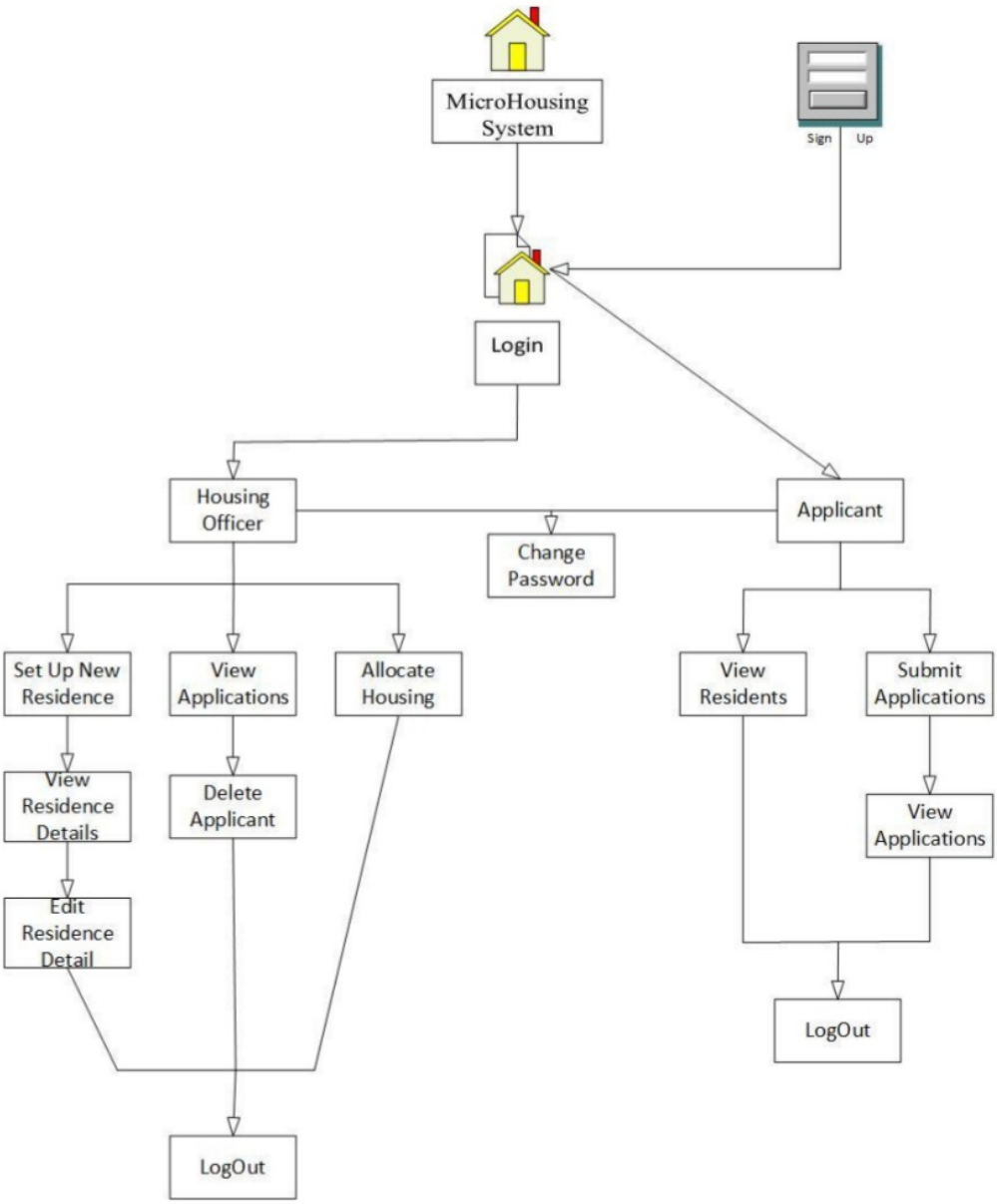
Class Diagram Design



Entity Relationship Diagram (ERD)

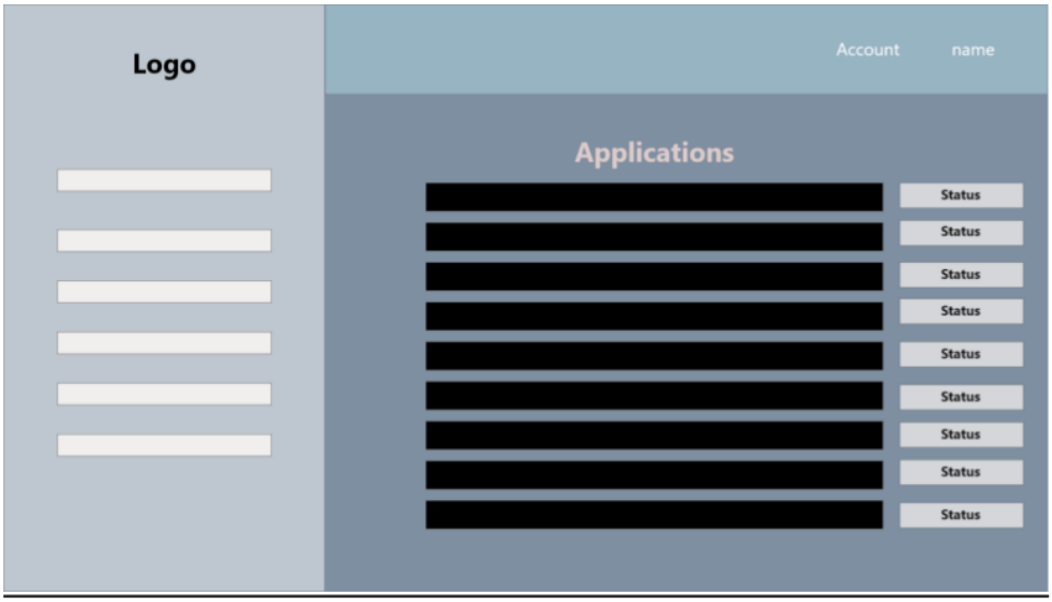


Sitemap

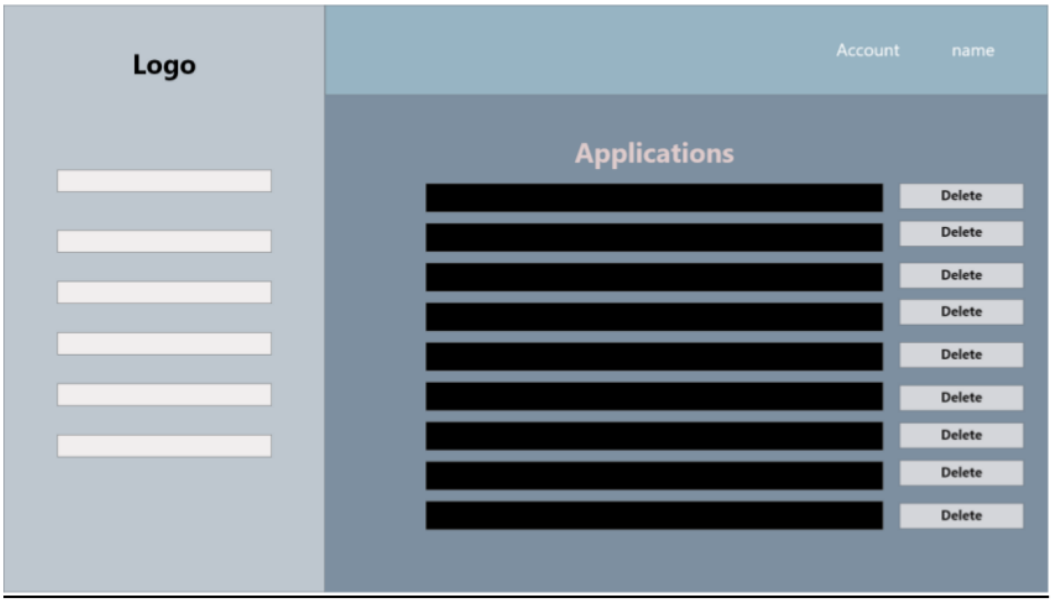

MicroHousing System
Sign
Up
Login
Housing Officer
Applicant
Change Password
Set Up New Residence
View Applications
Allocate Housing
View Residents
Submit Applications
View Residence Details
Delete Applicant
View Applications
Edit Residence Detail
LogOut
LogOut

Wireframe

Housing Officer



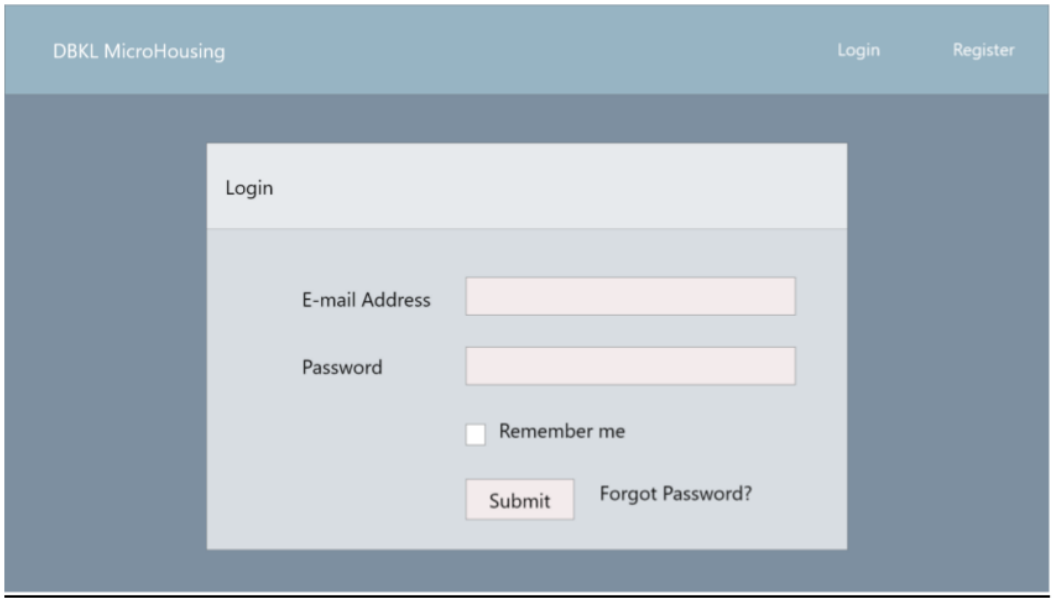
Allocate Housing



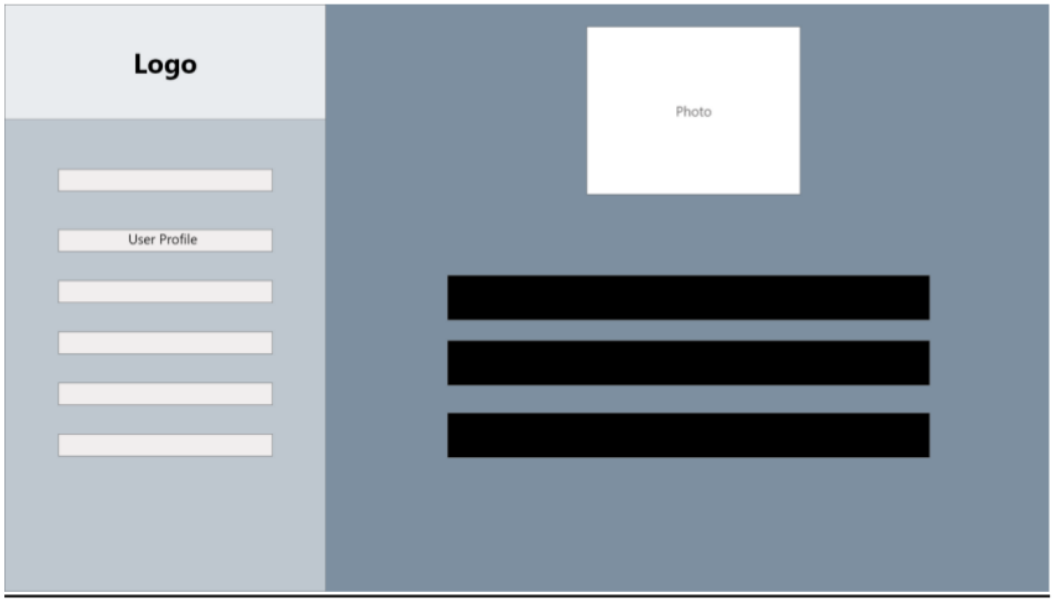
Delete Application

Edit Residence

Logout

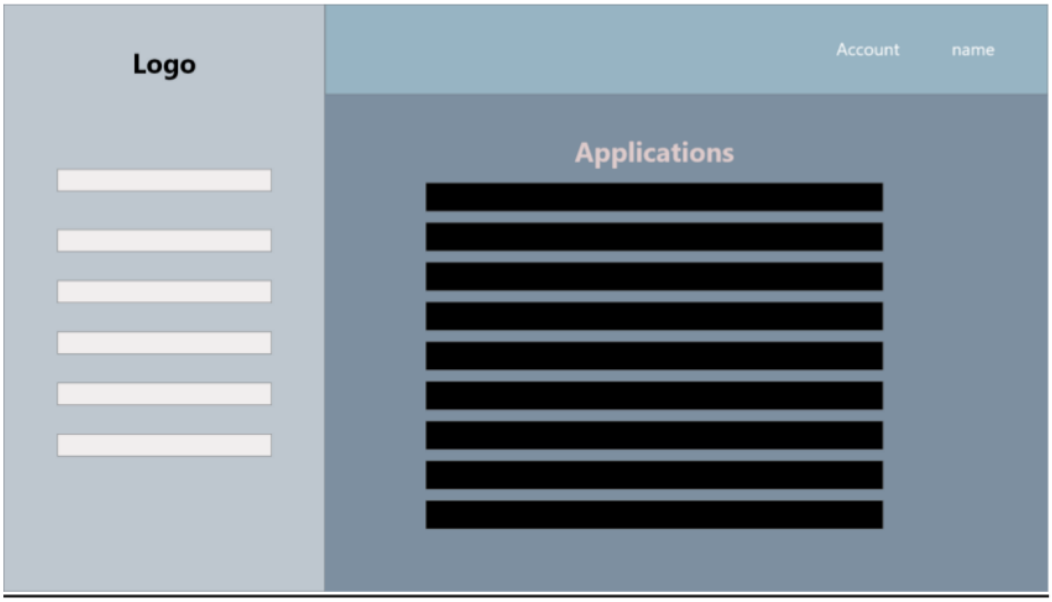


Login



My Profile

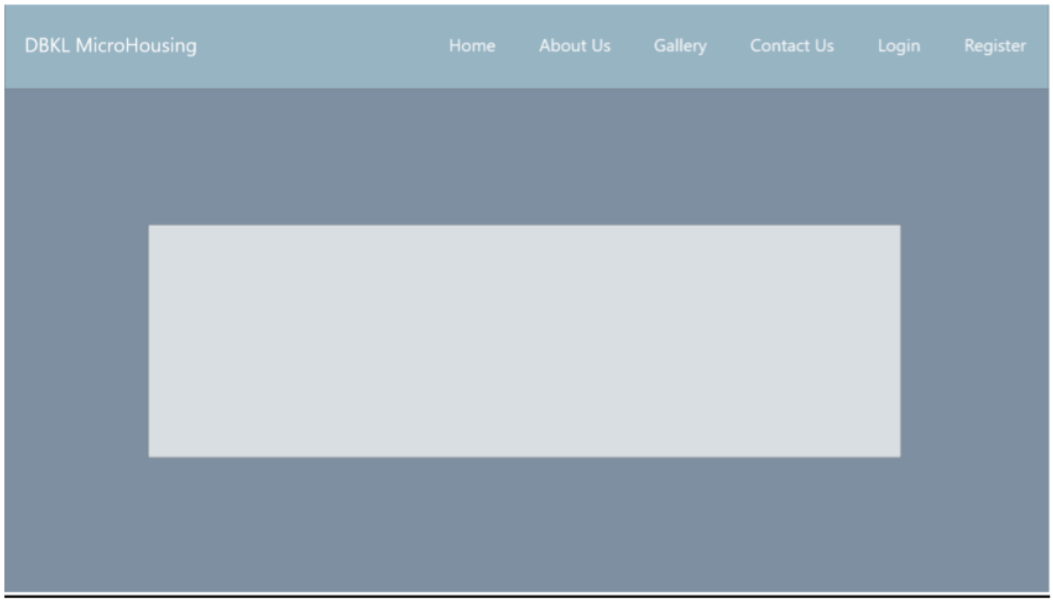
Set Up New Residence



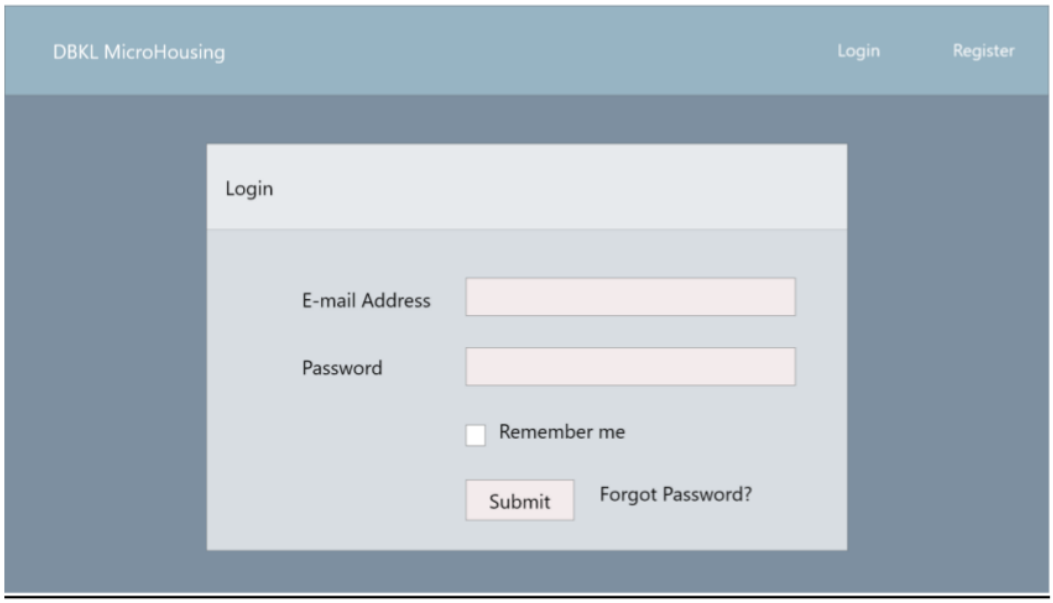
View Applications

View Residence

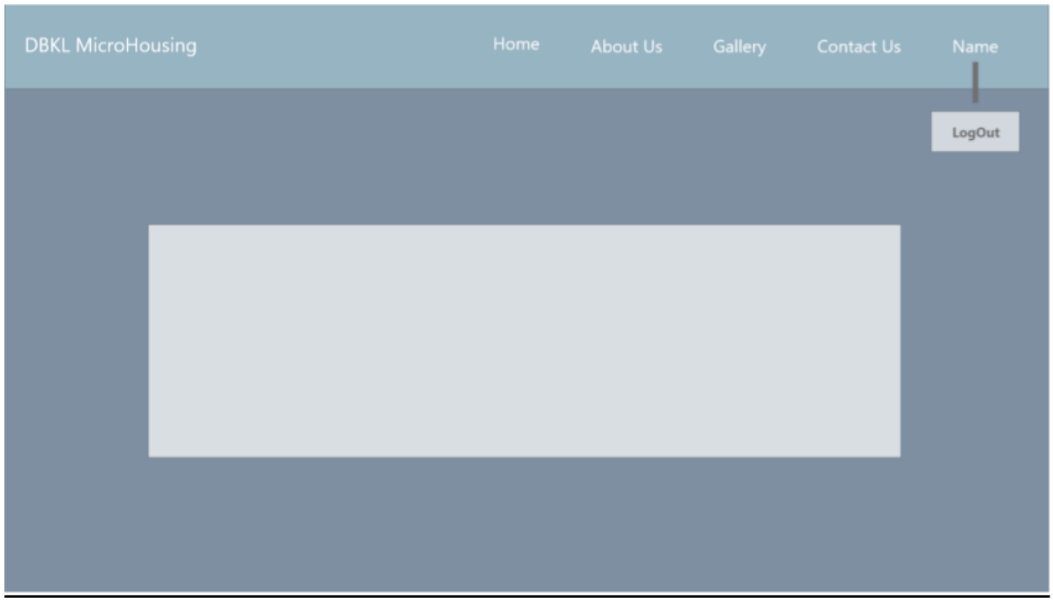
Applicant



Homepage

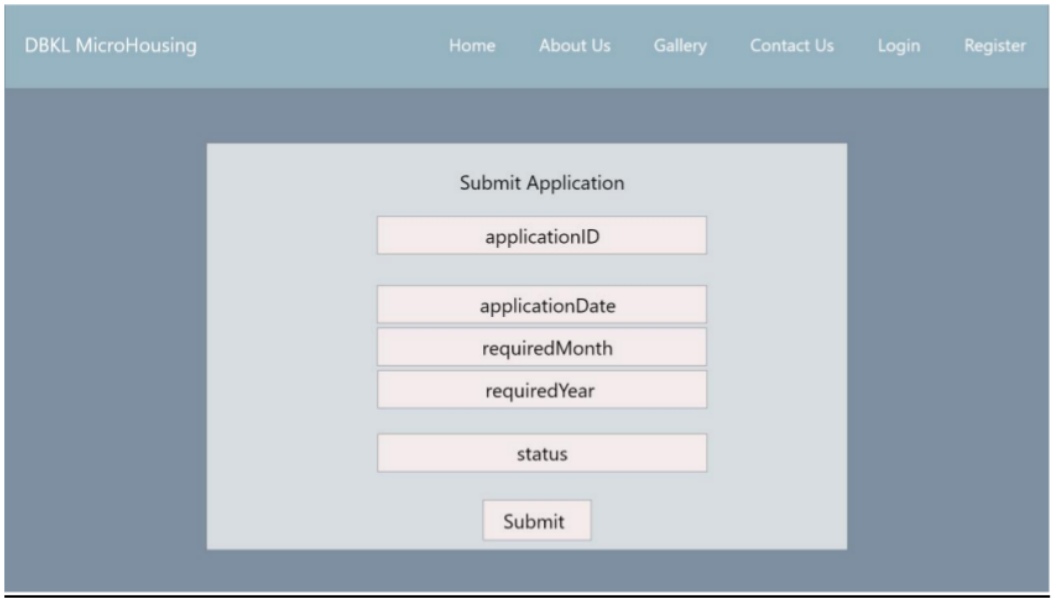


Login

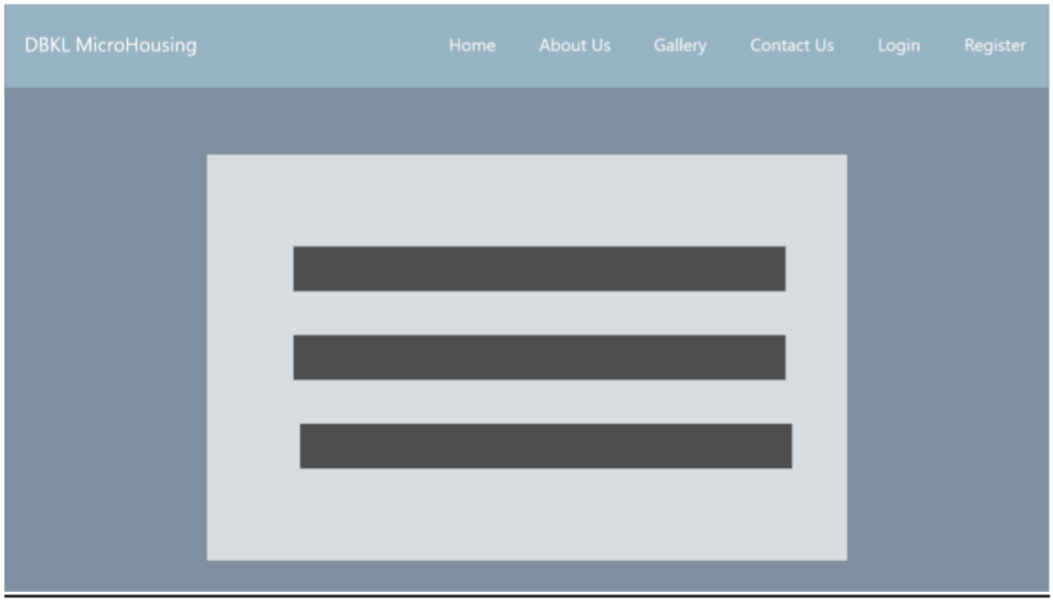


Logout

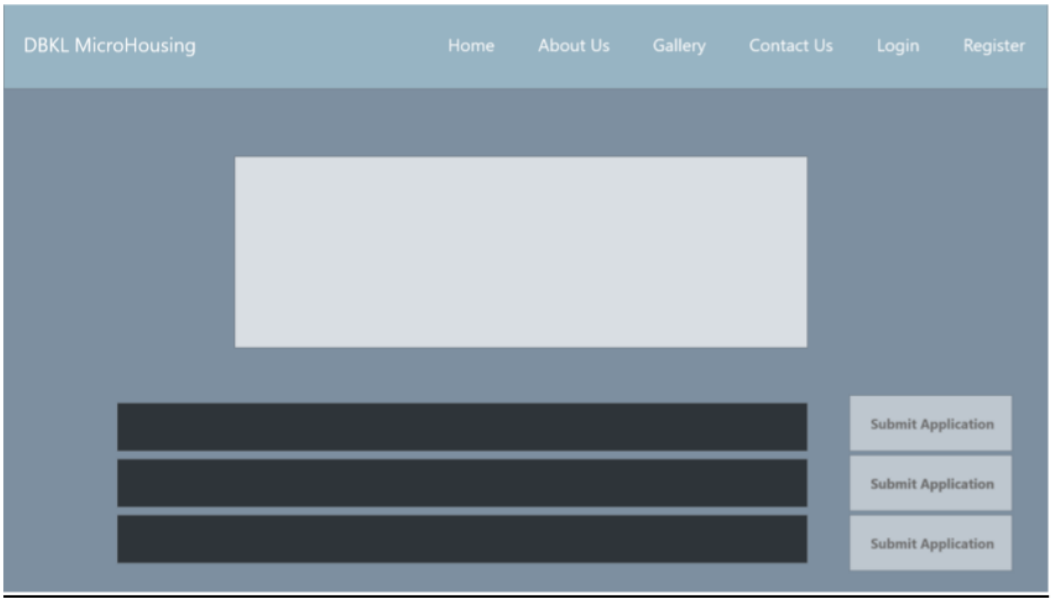
Register



Submit Applications



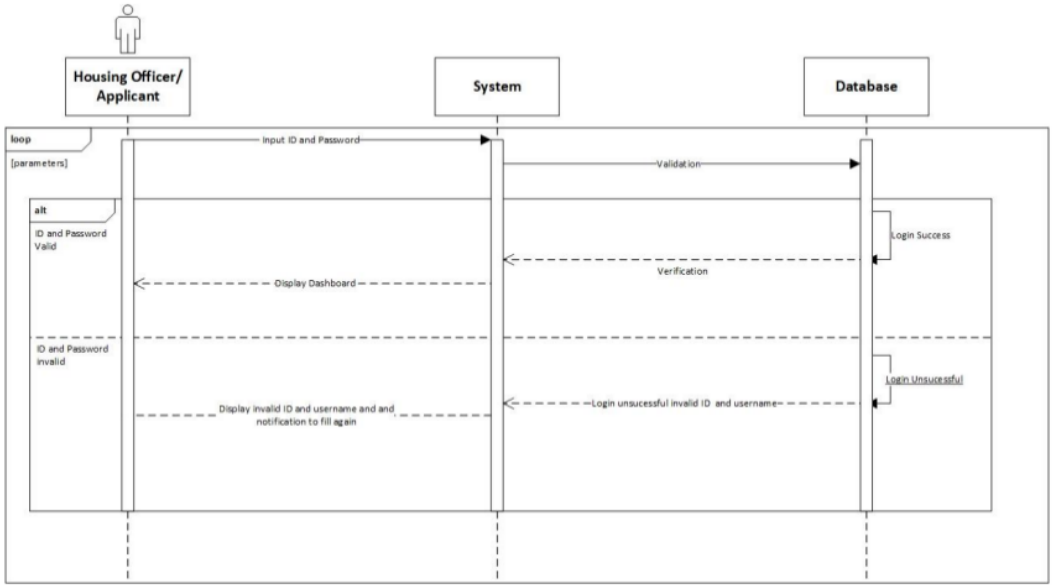
View Applications



View Residence

System Sequence Diagram + Contracts

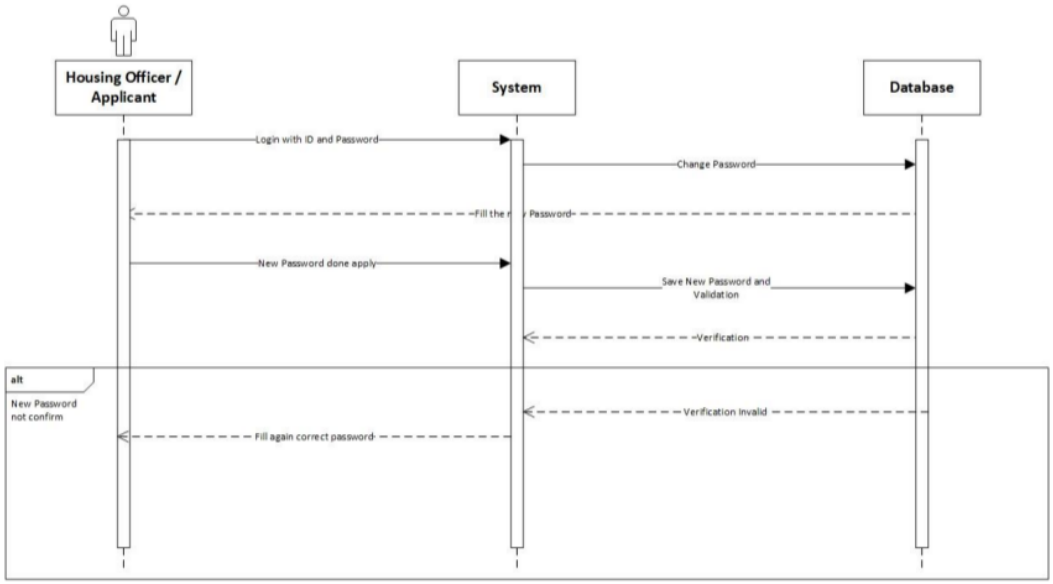
1. **Login**



Prepared by: Luh Wulandari Maharani

Cross References	Login
Operation	Login with ID and username
Responsible	To access the main page
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard

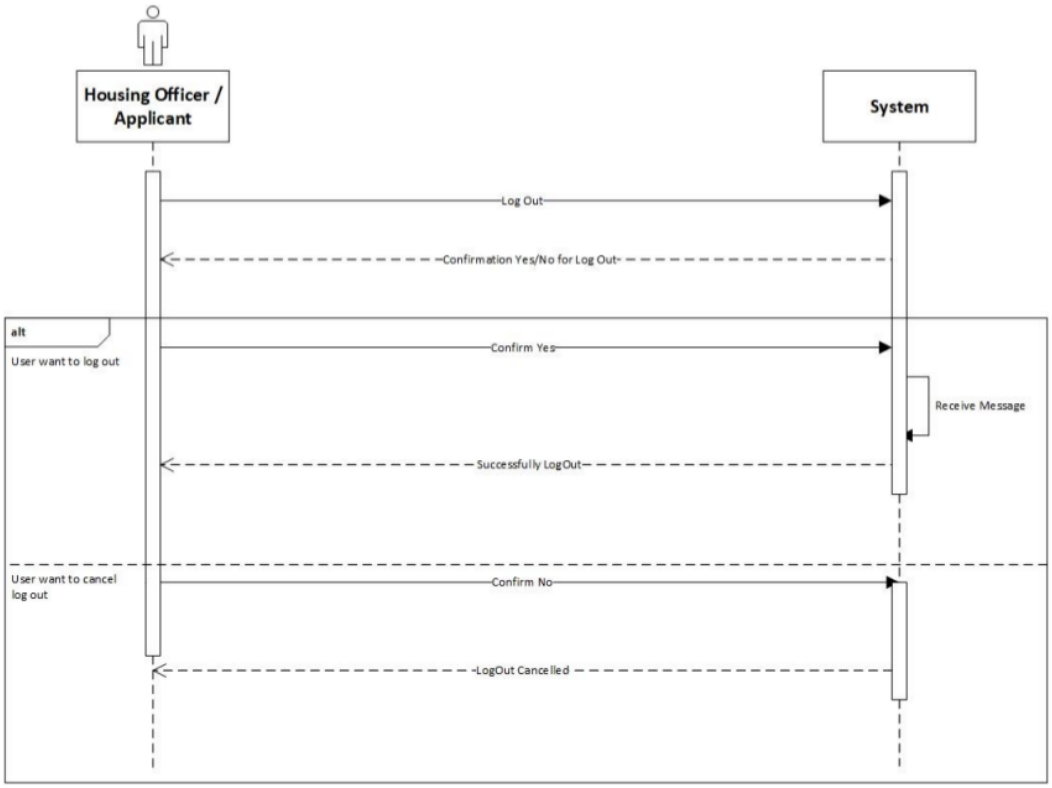
2. Change Password



Prepared by: Luh Wulandari Maharani

Cross References	Change Password
Operation	Enter current password
Responsible	To change the password
Pre-conditions	The new password must be available
Post-conditions	Successfully changed password
Cross References	**Change Password**
Operation	Enter current password
Responsible	To change the password
Pre-conditions	The user must be enter the new password again to confirm
Post-conditions	Fill again the form with the new password

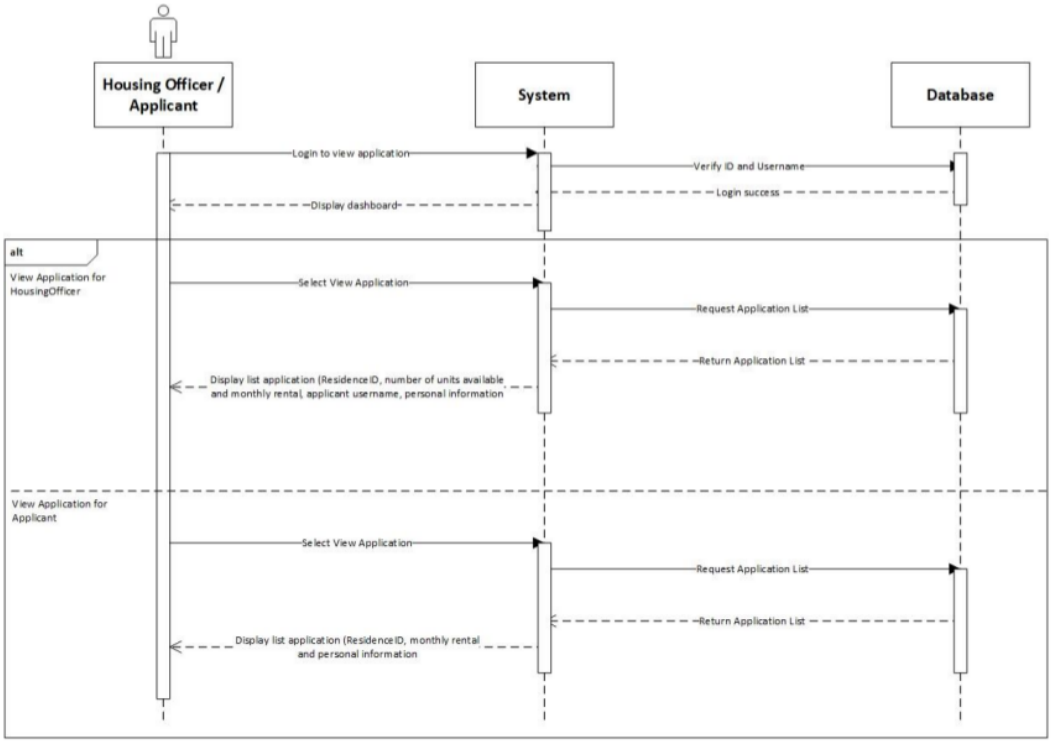
3. Log Out



Prepared by: Luh Wulandari Maharani

Cross References	Log Out
Operation	User want to log out
Responsible	To log out from the system
Pre-conditions	The user accepts that they want to log out
Post-conditions	Successfully log out
Cross References	**Log Out**
Operation	Cancel to log out
Responsible	To cancel log out from the system
Pre-conditions	The user cancel log out by click the “No” option
Post-conditions	Log out cancelled

4. View Applications



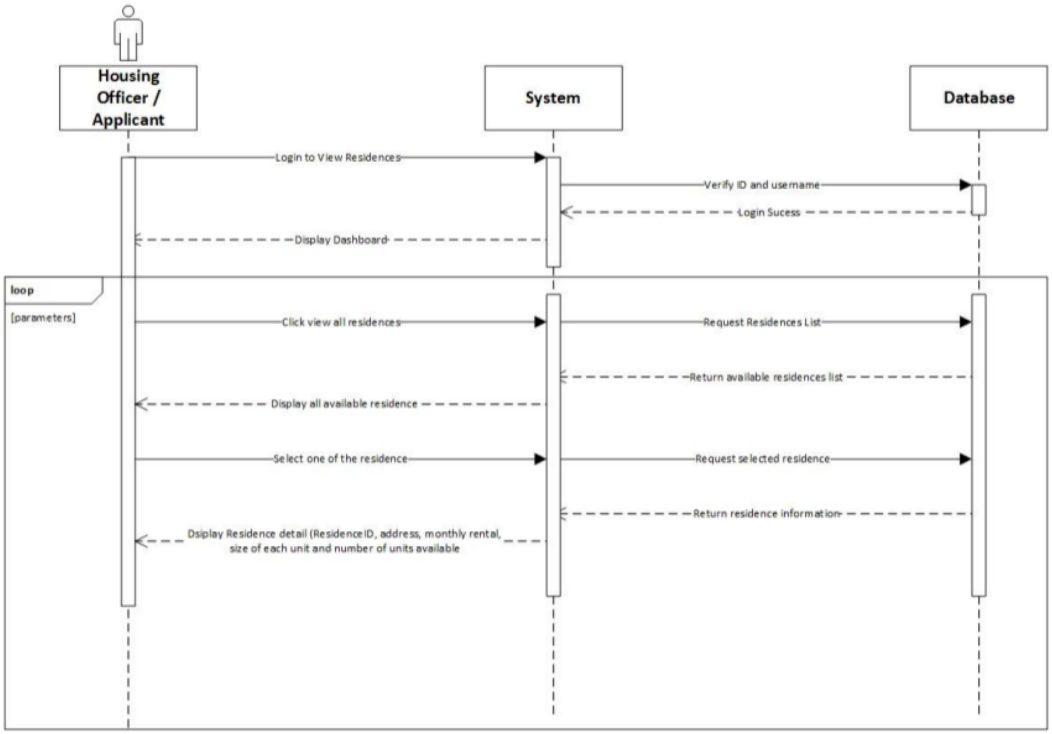
Prepared by: Luh Wulandari Maharani

Cross References	View Applications
Operation	Login with username and password
Responsible	To get access to view the applications
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard

Cross References	View Applications
Operation	View Application in Housing Officer
Responsible	To view a list for the Residence that Housing Officer is responsible
Pre-conditions	The application object must be available
Post-conditions	The list of application with status for the Residence that the Housing Officer is responsible, showing the residence ID, number of units available, monthly rental, application username and personal information

Cross References	View Applications
Operation	View Application in Applicant
Responsible	To view a list of application that have been made for applicant
Pre-conditions	The application object must be available
Post-conditions	The list of application that have been made by the applicant, showing the residence ID, monthly rental and personal information

5. View Residences

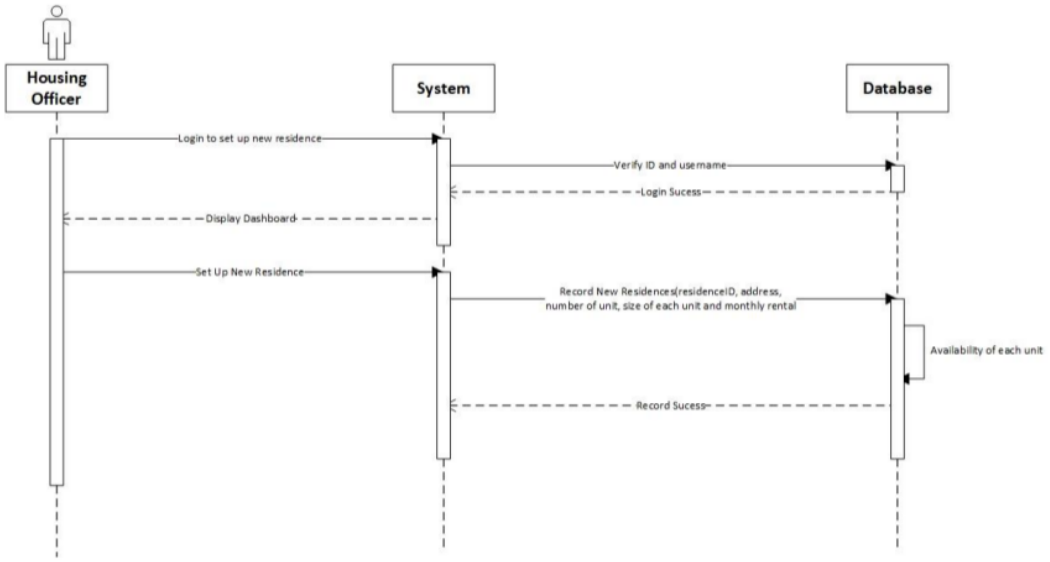


Prepared by: Luh Wulandari Maharani

Cross References	View Residences
Operation	Login with ID and username
Responsible	To access the main page
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard

Cross References	View Residences
Operation	Click view all residences
Responsible	To get all list of the residences
Pre-conditions	The residence object must be available
Post-conditions	Success to display all residences of all residences
Cross References	**View Residences**
Operation	Select the residence to view
Responsible	To get information about selected residences
Pre-conditions	The residence ID must be available
Post-conditions	Success to display detail information about the selected residences

6. Set Up New Residences

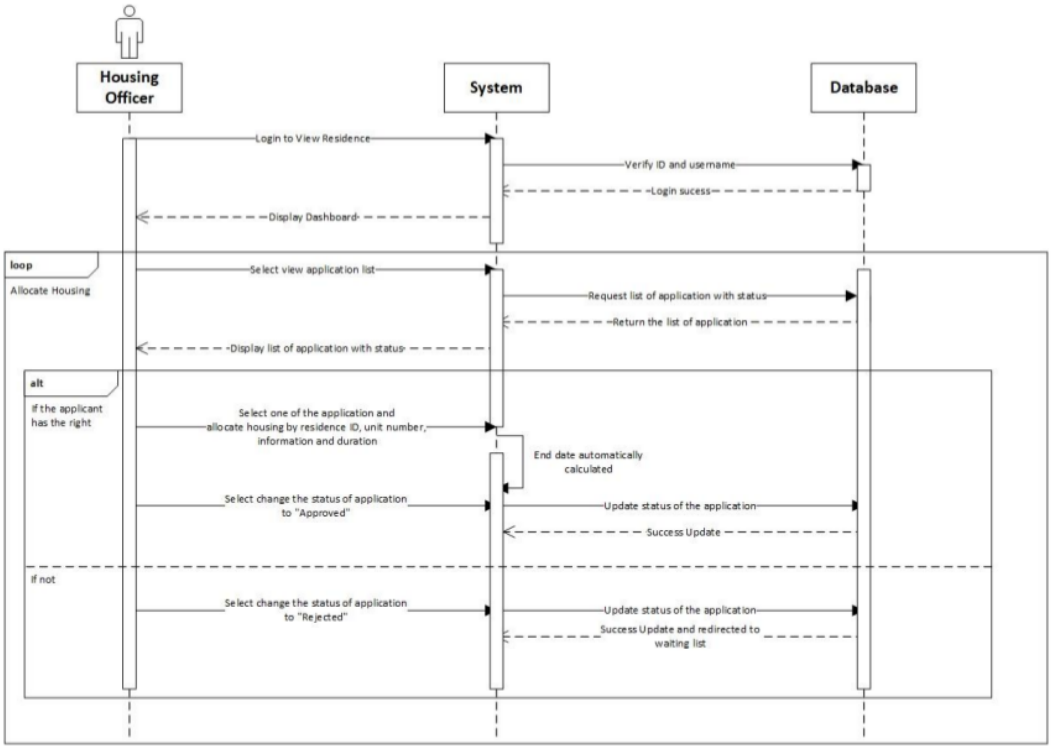


Prepared by: Luh Wulandari Maharani

Cross References	Set Up New Residences
Operation	Login with ID and username
Responsible	To access the main page
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard

Cross References	Set Up New Residences
Operation	Set up new residences
Responsible	To set up the new residences by input the residenceID, address, number of units available, size of each unit and monthly rental
Pre-conditions	Object residenceID, address, number of units available, size of each unit and monthly rental must be available
Post-conditions	New residences was successfully added to the system

7. Allocate Housing

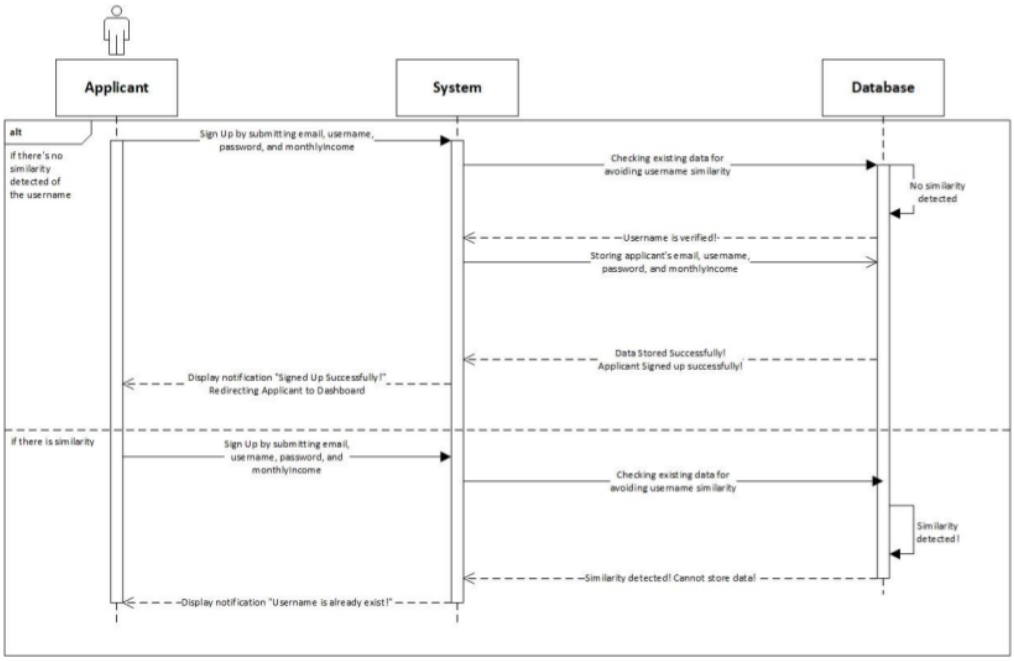


Prepared by: Luh Wulandari Maharani

Cross References	Allocate Housing
Operation	Login with ID and username
Responsible	To access the main page
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard

Cross References	Allocate Housing
Operation	Select view application list
Responsible	To get the list of all application
Pre-conditions	The application object must be available
Post-conditions	Success displayed list of all application with status
Cross References	**Allocate Housing**
Operation	Select one of the application and allocate the housing
Responsible	To allocate the housing for an application
Pre-conditions	Object residenceID, unit number, from date and duration must be available
Post-conditions	Success create allocation object based on data input
Cross References	**Allocate Housing**
Operation	Select to changed status of to be “Approved”
Responsible	To change the status to “Approved”
Pre-conditions	Application object must be available
Post-conditions	Success changed the application status
Cross References	**Allocate Housing**
Operation	Select to changed status of to be “Rejected”
Responsible	To change the status to “Rejected”
Pre-conditions	Application object must be available
Post-conditions	Success changed the application status and redirected to waiting list

8. Sign Up Applicant

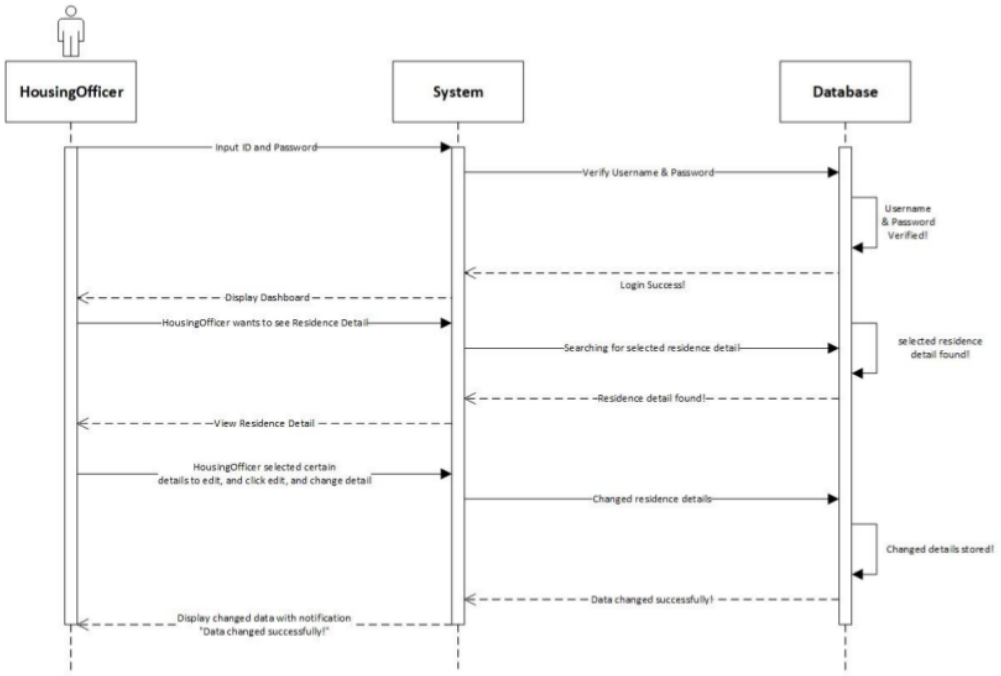


Prepared by: Rivaldo Bagus Soepardhy

Cross References	Sign up Applicant
Operation	Sign up with email, username, password, and monthlyIncome
Responsible	To be registered in system and able to access the main page
Pre-conditions	• Email must be available • Username must be available • Password must be available • MonthlyIncome must be available
Post-conditions	• Username is verified (no similarity detected) • Registered in System • Display dashboard

Cross References	Sign up Applicant
Operation	Sign up with email, username, password, and monthlyIncome
Responsible	To be registered in system and able to access the main page
Pre-conditions	• Email must be available • Username must be available • Password must be available • MonthlyIncome must be available
Post-conditions	• Username is not verified (already existed in system) • Applicant must re-enter new username.

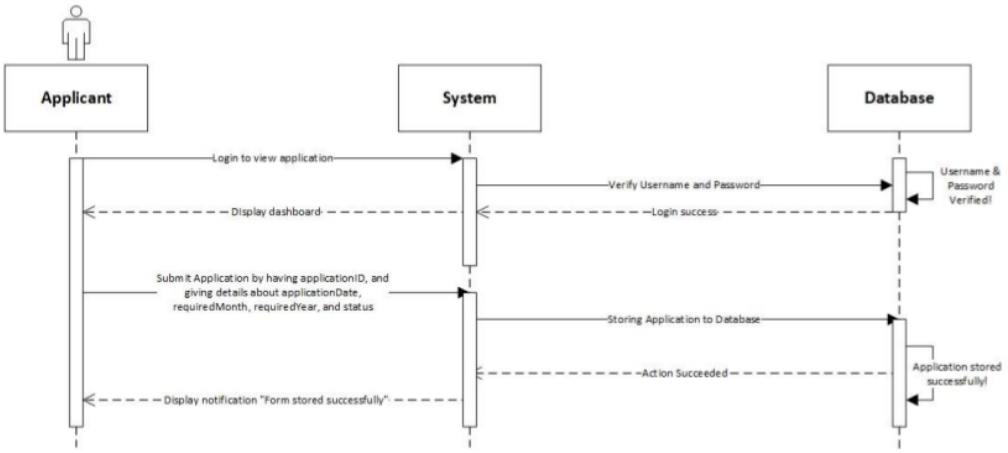
9. Edit Residence



Prepared by: Rivaldo Bagus Soepardhy

Cross References	Edit Residence Detail
Operation	Login with username and password
Responsible	To get access to view the applications
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard
Cross References	**Edit Residence Detail**
Operation	Edit residenceID, address, numUnits, sizePerUnit, monthlyRental
Responsible	To change certain Residence Detail
Pre-conditions	We will need **one of** or **all of** the details below: • Email must be available • Username must be available • Password must be available • MonthlyIncome must be available
Post-conditions	Residence Detail changed successfully

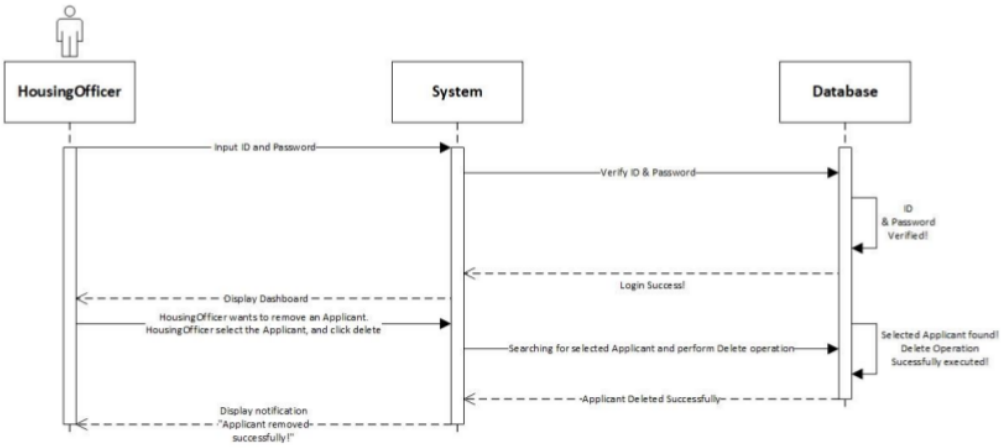
10. Submit Application



Prepared by: Rivaldo Bagus Soepardhy

Cross References	**Submit Application**
Operation	Login with username and password
Responsible	To get access to view the applications
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard
Cross References	**Submit Application**
Operation	View Residence & Select Residence
Responsible	To submit applicant's application to the database.
Pre-conditions	• Applicant is already logged in to the system • Applicant has already determine the desired residence
Post-conditions	User will view the list of the residence, and then select the desired residence.
Cross References	**Submit Application**
Operation	Submit applicationID, applicationDate, requiredMonth, requiredYear, and status
Responsible	To submit applicant's application to the database
Pre-conditions	Applicant has already selected the residence. We will need **all of** the details below: • applicationID will be given by system automatically • applicationDate must be available • requiredMonth must be available • requiredYear must be available • status will be given by system automatically
Post-conditions	Application will be stored successfully, and will be on waiting list for the approval & arrangement by HousingOfficer.

11. Delete Applicant

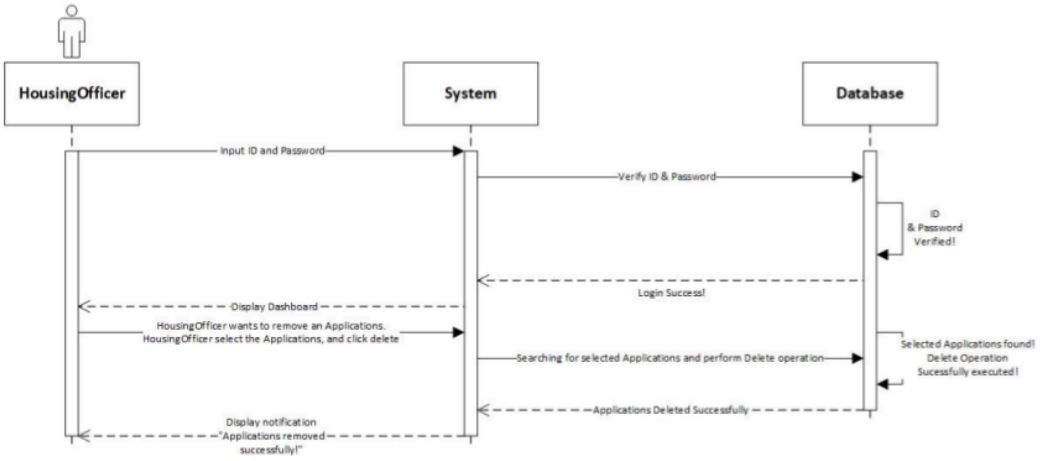


Prepared by: Rivaldo Bagus Soepardhy

Cross References	Delete Applicant
Operation	Login with ID and username
Responsible	To access dashboard, and delete applicant
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard

Cross References	Delete Applicant
Operation	Delete Applicant
Responsible	To remove an applicant
Pre-conditions	The applicant object must be available, and the applicant is no longer in place (checkout)
Post-conditions	Applicant object will be removed successfully

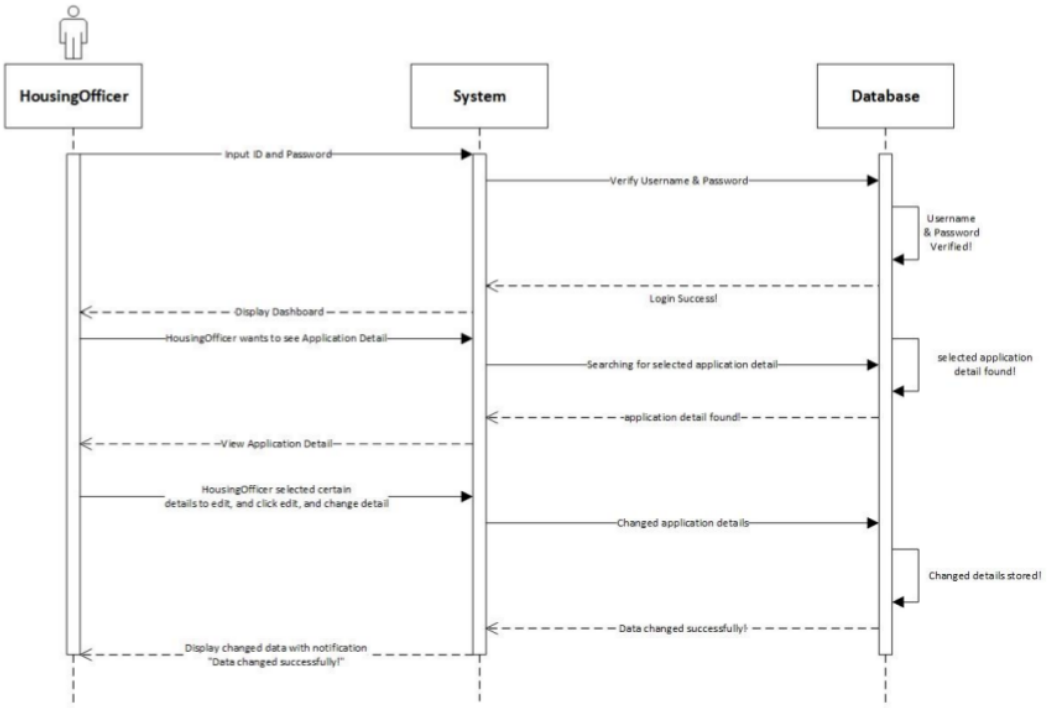
12. Delete Applications



Prepared by: Luh Wulandari Maharani

Cross References	Delete Applications
Operation	Login with ID and username
Responsible	To access dashboard, and delete applicant
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard
Cross References	**Delete Applications**
Operation	Delete Applications
Responsible	To remove an applications
Pre-conditions	The applications object must be available, and the applications is no longer in place (checkout)
Post-conditions	Applications object will be removed successfully

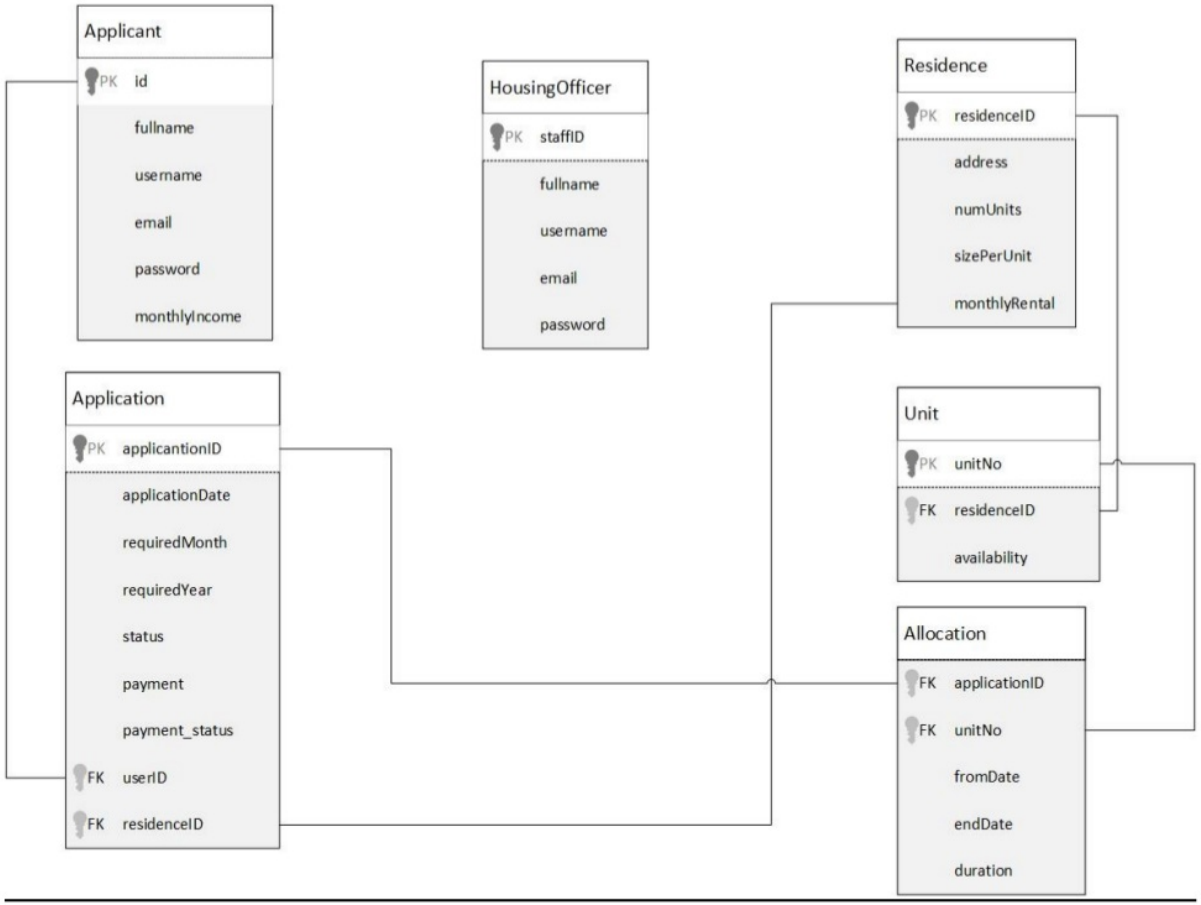
13. Edit Applications



Prepared by: Luh Wulandari Maharani

Cross References	Edit Applications
Operation	Login with username and password
Responsible	To get access to view the applications
Pre-conditions	• Username must be available • Password must be available
Post-conditions	• Username is matched • Password must be match based on user's password input • Display dashboard
Cross References	**Edit Applications Detail**
Operation	Edit status, payment and payment status
Responsible	To change certain Applications Detail
Pre-conditions	We will need **one of** or **all of** the details below: • applicantID must be available • residenceID must be available • requiredMonth must be available • requiredYear must be available
Post-conditions	Applications changed successfully

Database Design



Iteration 1

Use Case

Use Case	Developed By	Testing
Login	Wulan	Iteration 1
SignUp (For Applicant)	Wulan	Iteration 1
Change Password	Wulan	Iteration 2
Log Out	Wulan	Iteration 1
View Application (HousingOfficer)	Wulan	Iteration 1
Delete Application (HousingOfficer)	Wulan	Iteration 2
Edit Application (HousingOfficer)	Wulan	Iteration 2
Allocate Housing(HousingOfficer)	Wulan	Iteration 2
Set Up New Residence(HousingOfficer)	Wulan	Iteration 1
View Residence(HousingOfficer)	Wulan	Iteration 1
Edit Residence(HousingOfficer)	Wulan	Iteration 1
Delete Applicant(HousingOfficer)	Wulan	Iteration 2
Submit Application(Applicant)	Aldo	Iteration 1
View Applications(Applicant)	Aldo	Iteration 1
View Residences(Applicant)	Aldo	Iteration 1

Test Objectives

The main reason of the testing of this system is very obvious and inevitable, to find mistakes or errors inside the system. Although there is no such a "perfect system", it does not mean that we keep making systems without consideration and thorough testing. With this testing and a high frequency of testing, it will build confidence of the software quality, thus making us sure, the software will be very robust. Testing is not only about finding errors or mistake, but also to analyze the system and countering any possibility of errors that can possibly happen in the future.

For DBKL MicroHousing information system, our first objective is to make sure login validation works and password data is successfully encrypted with encryption provided by the framework we use to work. Then we test the application (web pages) across different web browser to ensure it is working properly. Next, we test the system according to the role:

- As HousingOfficer, he/she should be able to do basic CRUD (Create, Read, Update, Delete) operation in the system on certain object, such as residence, allocation, etc. HousingOfficer will be able to do some deciding action regarding application, whether to apply or reject it.
- As Applicant, he/she should be able view data only related to his role and application, such as Application Status, Payment, Residence, etc.

We also have to test all the links, buttons, pages, are working properly and directing to the right pages.

Test Plan

For the information system, we will be testing functions that will be performed by HousingOfficer and Applicant. We want to make sure that system boundaries are working. For example, Applicant cannot edit or delete data. Login system will also be tested, so that HousingOfficer and Applicant will be redirected to their own dashboard. During the testing process, we will test a function with several times of repeat to verify that the function has worked perfectly.

Unit Testing

1. **Register for Applicant**

This create_users_table function is used to make a users table in the microhousing database and declare each variable which will be used for the register applicant.

```php
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('name');
        $table->string('username')->unique();
        $table->string('usertype')->nullable();
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->bigInteger('monthlyincome')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
```

This registerContoller function is use to validate input from user when registering as an Applicant to the DBKL Microhousing. If applicant input invalid information the system will not accept the information and display an error from invalid input from the applicant.

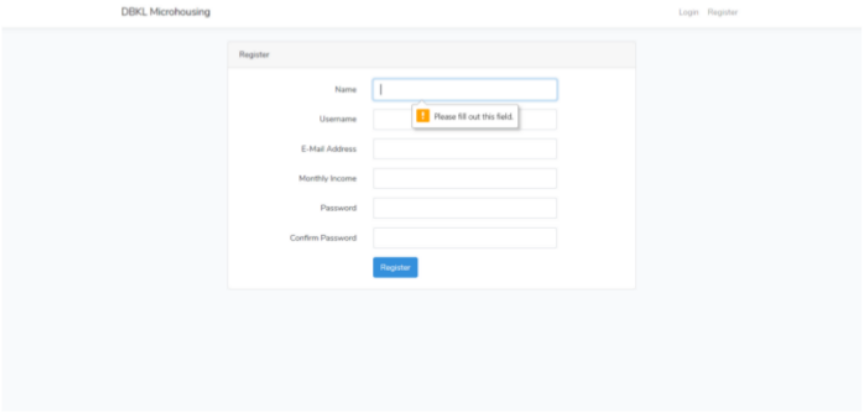
```php
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'username' => ['required', 'string', 'max:255', 'unique:users'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'monthlyincome' => ['required', 'string', 'max:255'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
        ]);
    }
    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        return User::create([
            'name' => $data['name'],
            'username' => $data['username'],
            'email' => $data['email'],
            'monthlyincome' => $data['monthlyincome'],
            'password' => Hash::make($data['password']),
        ]);
    }
}
```

This $fillable is register the attribute (column name) that we can fill in when inserting or updating to the database.

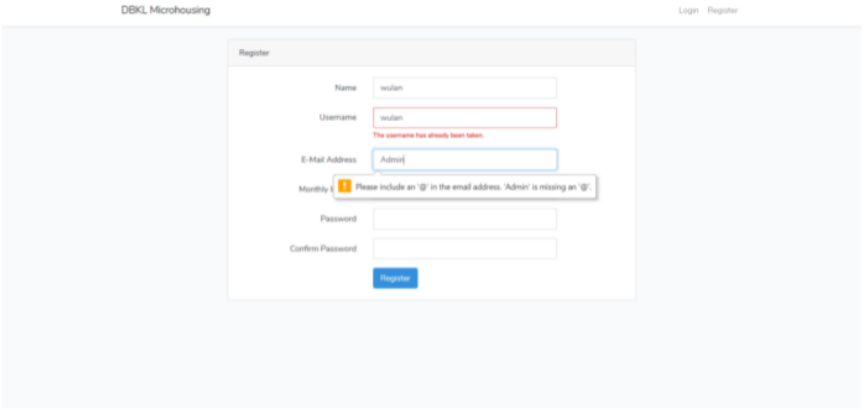
```php
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name','username', 'email','monthlyincome', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
```

a) Applicant input blank information

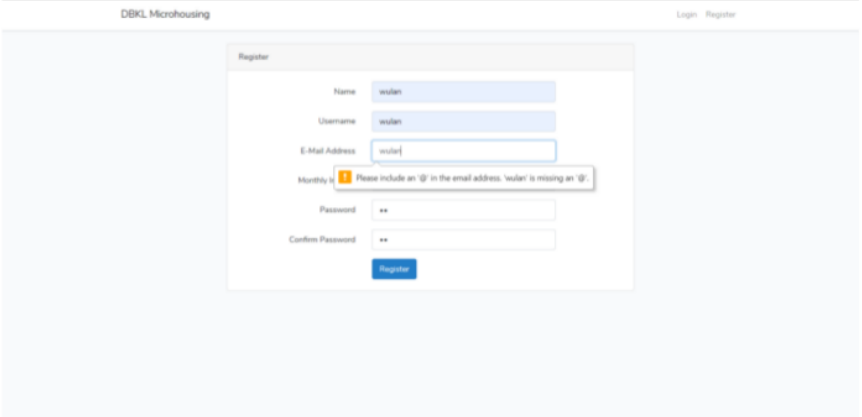
The required rule function was complete successfully.

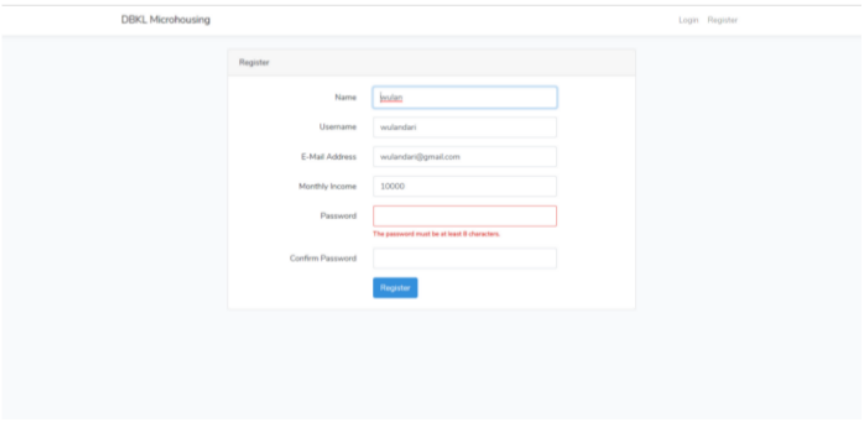


b) Applicant input wrong email or username (username already in the database / not unique) unique() function is for username and email was complete successfully.



c) Applicant input invalid email and password (too short) class email and password that has class for display error which is @enderror were complete successfully.





If applicant input valid information, the system will enter the applicant's information to the database 'microhousing' automatically.

d) Password Hashing

When register, the applicant's password will be saved in the database, and the password will be hashed so that won't be easy to read by others.

username	usertype	password 1
aldobagus	Housingofficer	$2y$10$raEeBv7zHvNbISqH0bNiOegCjPfVFefJg7m5iNk6Yir...
wulan	Applicant	$2y$10$KLgd5f6Lf6hqx1Ts4UbEp.L1YWlBu.ghU.bSpmSNwy4...

2. **Login**

This loginController function is to validate and separated according to the role as housing officer and as applicant which is will be input manual in usertype (to directed each web page).

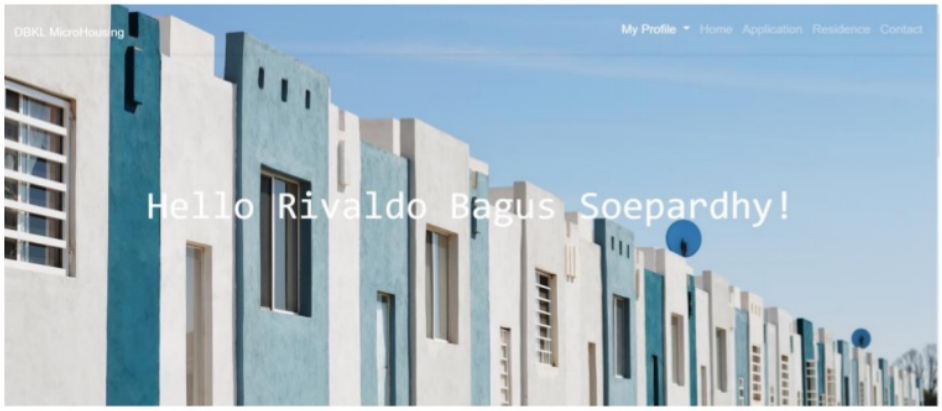
```
public function login(Request $request)
{
$this->validate($request, [
    'username' => 'required|string',
    'password' => 'required|string|min:6',
]);


$loginType = filter_var($request->username, FILTER_VALIDATE_EMAIL) ? 'email' : 'username';


$login = [
    $loginType => $request->username,
    'password' => $request->password
];


if (auth()->attempt($login)) {
    if(Auth::user()->usertype=='Housingofficer')
    {
        return redirect()->intended('/dashboard');
    }
    else
    {
        return redirect()->intended('/dashboard_user');
    }

}

return redirect()->route('login')->with(['error' => 'Email/Password salah!']);
}
```

This function to make a form for login and the input will be verified whether it is in the database.

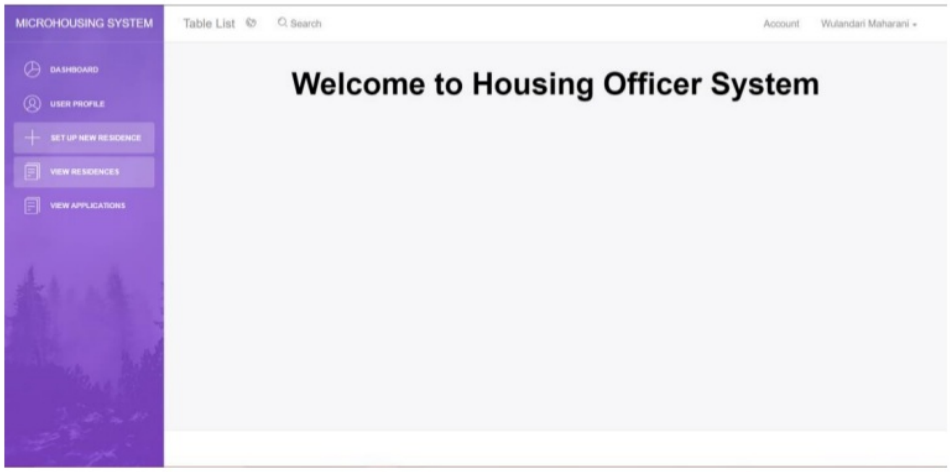
```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Login') }}</div>
                <div class="card-body">
                    <form method="POST" action="{{ route('login') }}">
                        @csrf
                        <div class="form-group row">
                            <label for="username" class="col-md-4 col-form-label text-md-right">Username/Email</label>
                            <div class="col-md-6">
                                <input id="username" type="text" class="form-control @error('username') is-invalid @enderror" name="username" value=

                                @error('username')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="form-group row">
                            <label for="password" class="col-md-4 col-form-label text-md-right">{{ __('Password') }}</label>

                            <div class="col-md-6">
                                <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name="password" re

                                @error('password')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>
```

User input invalid username and password, the system will not response and will stay in the login page. User input valid username and password, the system will checking into the database and each user will be direct to their dashboard according to their role.

Applicant Homepage

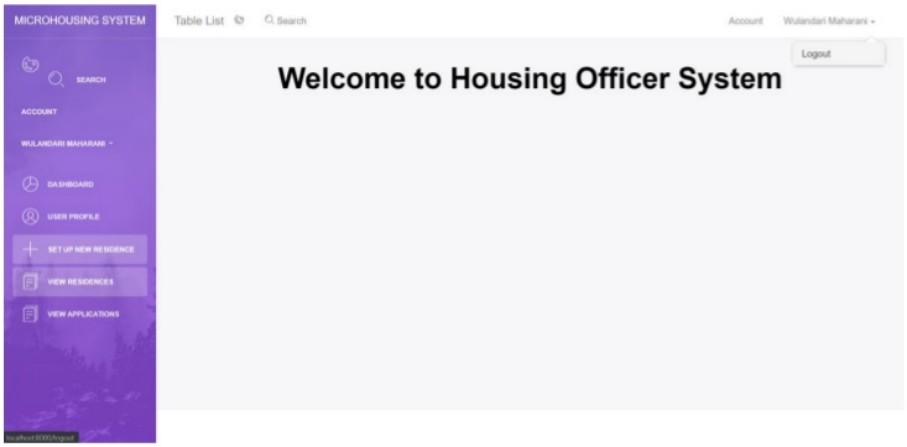


Housing Officer Homepage



3. **LogOut**
   This nav bar will delete the user session which is allocate in master.blade.php. so the website cannot access user information again. If this function is remove, so the user will return to the dashboard login page.

```
<li class="nav-item dropdown">
    <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-pre>
        {{ Auth::user()->name }} <span class="caret"></span>
    </a>

    <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="{{ route('logout') }}"
           onclick="event.preventDefault();
                         document.getElementById('logout-form').submit();">
            {{ __('Logout') }}
        </a>

        <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">
            @csrf
        </form>
    </div>
</li>
```



Back to dashboard page

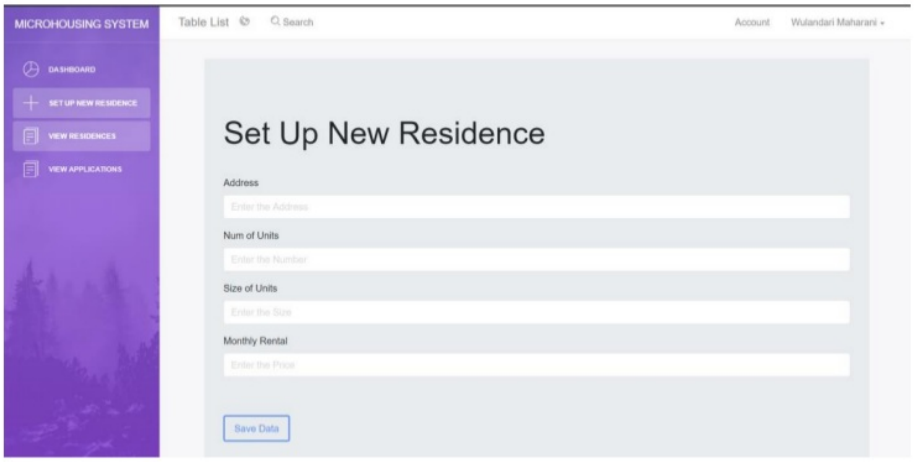
4. Set Up New Residence

In this test, the set up new residence will be able to add residence and each variable will input to 'microhousing' database in residence table.

```
public function addresidences()
{
    return view('Housingofficer.addresidences');
}
public function store(Request $request)
{
    $residences = new Residences();
    $residences->address = $request->input('address');
    $residences->numunits = $request->input('numunits');
    $residences->sizeperunits = $request->input('sizeperunits');
    $residences->monthlyrental = $request->input('monthlyrental');

    $residences->save();
    return view('Housingofficer.addresidences')->with('Housingofficer.addresidences',$residences);
}
```

This addresidence.blade.php which is form to input the data

```
@extends ('layouts.master')

@section('title')
    Set Up New Residence
@endsection


@section('content')

    <div class="container">
        <div class="jumbotron">
        <h1> Set Up New Residence</h1><br>
            <form action="{{ route ('addedData')}}" method="POST" enctype="multipart/form-data">
            {{csrf_field()}}
                <div class="form-group">
                    <label> Address </label>
                    <input type="text" name="address" class="form-control" placeholder="Enter the Address">
                </div>

                <div class="form-group">
                    <label> Num of Units </label>
                    <input type="text" name="numunits" class="form-control" placeholder="Enter the Number">
                </div>

                <div class="form-group">
                    <label> Size of Units </label>
                    <input type="text" name="sizeperunits" class="form-control" placeholder="Enter the Size">
                </div>

                <div class="form-group">
                    <label> Monthly Rental </label>
                    <input type="text" name="monthlyrental" class="form-control" placeholder="Enter the Price">
                </div>
                <br><br>
                <button type="submit" name="submit" class="btn btn-primary"> Save Data </button>

</body>
</html>
```



5. **View Residence Detail**

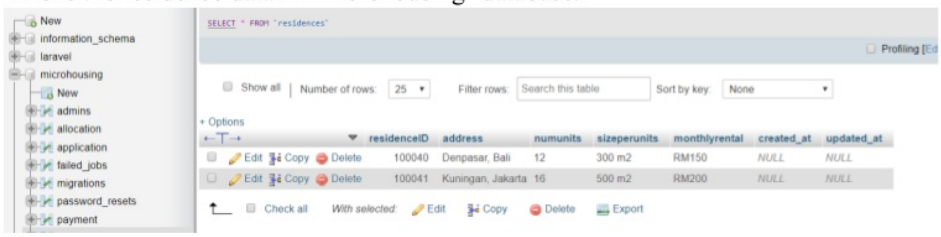
This test, viewresidence.blade.php function runs properly, displaying all the residence in the database. Residence data in the database will be stored in row that is will be send to the housingofficer's page.

```
@extends ('layouts.master')

@section('title')
    View Residences
@endsection


@section('content')
<div class="container-fluid">
                <div class="row">
                    <div class="col-md-12">
                        <div class="card strpied-tabled-with-hover">
                            <div class="card-header ">
                                <h4 class="card-title">View Residences</h4>
                            </div>
                            <div class="card-body table-full-width table-responsive">
                                <table class="table table-hover table-striped">
                                    <thead>
                                        <th>Residence ID</th>
                                        <th>Address</th>
                                        <th>Number of Units</th>
                                        <th>Size per Unit </th>
                                        <th>Monthly Rental</th>
                                        <th>EDIT</th>

                                    </thead>
                                    <tbody>
                                        @foreach ($residencess as $residence)
                                        <tr>
                                            <td>{{$residence->residenceID}}</td>
                                            <td>{{$residence->address}}</td>
                                            <td>{{$residence->numunits}}</td>
                                            <td>{{$residence->sizeperunits}}</td>
                                            <td>{{$residence->monthlyrental}}</td>

                                            <td>
                                                <a href="/editresidences{{$residence->residenceID}}" class="btn btn-sucess">EDIT</a>
                                            </td>
```

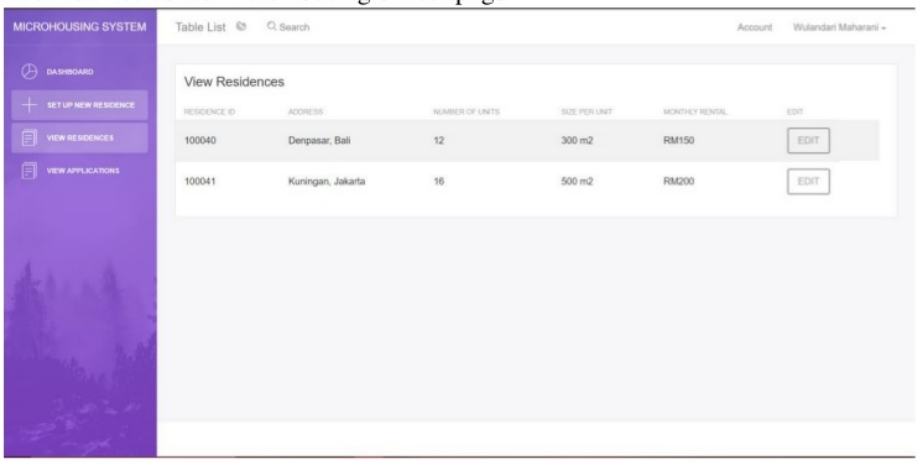
This function to call all the residence which has been inputted and store in the database.

```
public function viewresidences()
{
    $residencess = Residences::all();
    return view('Housingofficer.viewresidences')->with('residencess',$residencess);
}
```

This is the residence data in 'microhousing' database.



The view residence in the housing officer page



6. Edit Residence Detail

In this test will editresidence.blade.php function runs properly. This function useful for editing the residence, which is already input. For editing we need 'Model' that will receive data input from the residence model and then update the data into the database. This is the Residence.model

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Residences extends Model
{
    protected $table='residences';
    protected $primaryKey = 'residenceID';
    protected $fillable=['address','numunits','sizeperunits','monthlyrental'];
}
```

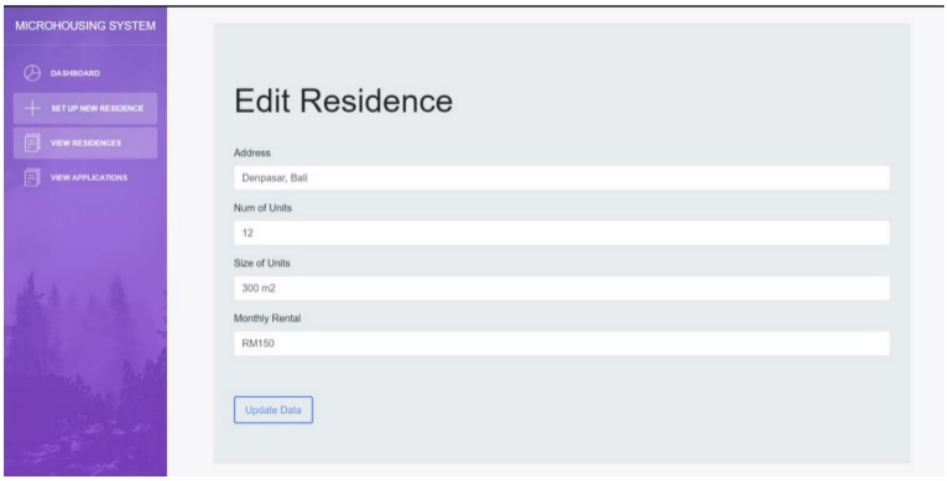
This function is to find the residenceID because when we want to edit every residence has edit button so we will find the residenceID.

```php
public function editresidences($residenceID)
{
    $residencess = Residences::find($residenceID);
    return view('Housingofficer.editresidences')->with('residencess',$residencess);
}
```

This is the editresidence.blade.php is form for the edit residence.

```
@extends ('layouts.master')

@section('title')
    Edit Residence
@endsection


@section('content')

    <div class="container">
        <div class="jumbotron">
        <h1> Edit Residence</h1><br>
            <form action="/updateresidences{{$residencess->residenceID}}" method="POST" enctype="multipart/form-data">
            {{csrf_field()}}
            {{method_field('PUT')}}

                <input type="hidden" name="residenceID" residenceID="residenceID" value="{{$residencess->residenceID}}">

                <div class="form-group">
                    <label> Address </label>
                    <input type="text" name="address" class="form-control" value="{{$residencess->address}}" placeholder="Enter the Address">
                </div>

                <div class="form-group">
                    <label> Num of Units </label>
                    <input type="text" name="numunits" class="form-control" value="{{$residencess->numunits}}"placeholder="Enter the Number">
                </div>

                <div class="form-group">
                    <label> Size of Units </label>
                    <input type="text" name="sizeperunits" class="form-control" value="{{$residencess->sizeperunits}}"placeholder="Enter the Size">
                </div>

                <div class="form-group">
                    <label> Monthly Rental </label>
                    <input type="text" name="monthlyrental" class="form-control" value="{{$residencess->monthlyrental}}"placeholder="Enter the Price
                </div>
```

This this display form edit residence



7. **View Applications**

This test, viewapplication.blade.php function runs properly, the applications are displayed in the view application menu for Housing Officer menu, are applications that related to the residence.

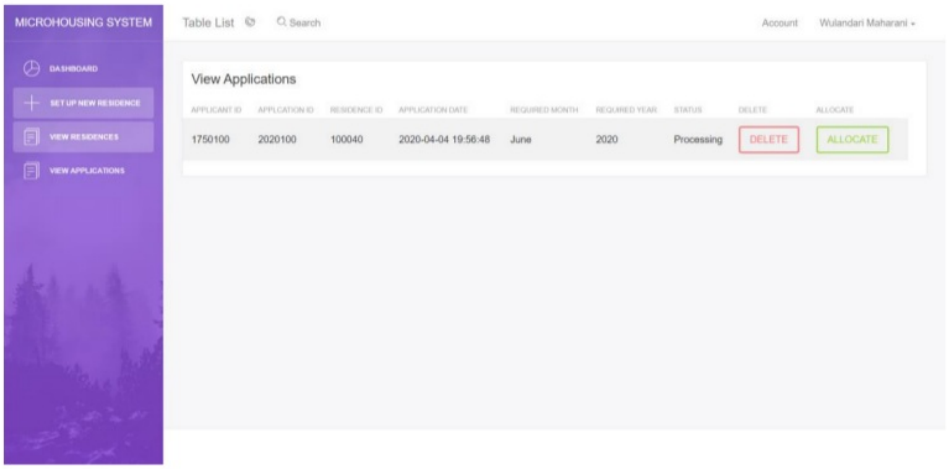
This is the form for the view application and the button for delete and allocate but not functioning yet

```
@extends ('layouts.master')

@section('title')
    View Applications
@endsection


@section('content')
<div class="container-fluid">
                    <div class="row">
                        <div class="col-md-12">
                            <div class="card strpied-tabled-with-hover">
                                <div class="card-header ">
                                    <h4 class="card-title">View Applications</h4>
                                </div>
                                <div class="card-body table-full-width table-responsive">
                                    <table class="table table-hover table-striped">
                                        <thead>
                                            <th>Applicant ID</th>
                                            <th>Applcation ID</th>
                                            <th>Residence ID</th>
                                            <th>Application Date</th>
                                            <th>Required Month</th>
                                            <th>Required Year </th>
                                            <th>Status</th>
                                            <th>DELETE</th>
                                            <th>ALLOCATE</th>

                                        </thead>
                                        <tbody>
                                            @foreach($application as $app)
                                            <tr>
                                                <td>{{ $app->applicationID}}</td>
                                                <td>{{ $app->userID}}</td>
                                                <td>{{ $app->residenceID}}</td>
                                                <td>{{ $app->applicationDate}}</td>
                                                <td>{{ $app->requiredMonth}}</td>
                                                <td>{{ $app->requiredYear}}</td>
```

This function is to call the table application in the database, which is input by the applicant

```
public function viewapplication()
{
    $application = DB::table('application')->get();
    return view('Housingofficer.viewapplication',['application' => $application]);
}
```

This display for view applications from input by applicant



8. Submit Application

This test applicationcontroller , will be able to submit application and each variable will input to 'microhousing' database in application table.

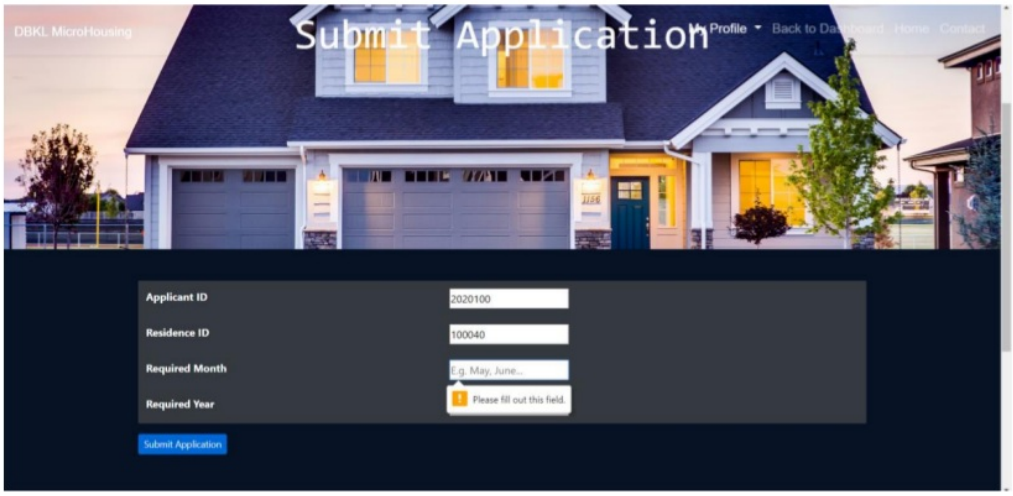
```
public function SubmitApplication($id)
{
    $residence = DB::table('residences')->where('residenceID',$id)->get();
    return view('Applicant.SubmitApplication',['residences' => $residence]);
}
public function store(Request $request)
{
    DB::table('application')->insert([
        'requiredMonth' => $request->requiredMonth,
        'requiredYear' => $request->requiredYear,
        'status' => $request->status,
        'residenceID' => $request->residenceID,
        'userID' => $request->userID,
        'payment' => $request->payment,
        'payment_status' => $request->payment_status,
    ]);
    return redirect('/dashboard_user');
}
```

This is form for the submit application that is submitapplication.blade.php

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Submit</title>
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/bootstrap.css') }}">
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/bootstrap.min.css') }}">
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/dbkl.css') }}">
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/animate.css') }}">
    </head>
    <body>
       <nav class="navbar navbar-expand-sm navbar-dark fixed-top shadow-sm" role="navigation">
            <h2><a class="navbar-brand" href="/dashboard_user">
                DBKL MicroHousing
            </a></h2>
                <div class="navbar-header page-scroll">
                    <button type="button" class="navbar-toggler" data-toggle="collapse" data-target=".navbar-main-collapse">
                        <span class="navbar-toggler-icon"></span>
                    </button>
                </div>
                <div class="collapse navbar-collapse navbar-main-collapse" id="NavNavbar">
                    <ul class="navbar-nav ml-auto">
                        <li class="nav-link nav-item dropdown">
                            <a class="dropdown-toggle" href="#" aria-haspopup="true" aria-expanded="false" id="navbardrop" data-toggle="dropdown">
                                <span class="caret">
                                My Profile
                                </span>
                            </a>
                              <div class="dropdown-menu animate slideIn">
                                  <a class="dropdown-item" href="#" data-toggle="modal" data-target="#myModal">Account Details</a>
                                  <a class="dropdown-item" href="#">Change Password</a>
                                  <a class="dropdown-item" href="{{ route('logout') }}"onclick="event.preventDefault();document.getElementById('logo
                                      {{ __('Logout') }}</a>
                                  <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">@csrf</form>
                              </div>
                        </li>
```

This is display submit application page

If the applicant input invalid information that is required month and required year (Not fill), the system will display an error.



9. **View Residence (Applicant)**
   In this test, viewresidence.blade.php which is the folder function runs properly, displaying all the residence which is input by the housing officer and all the residence data in the database.

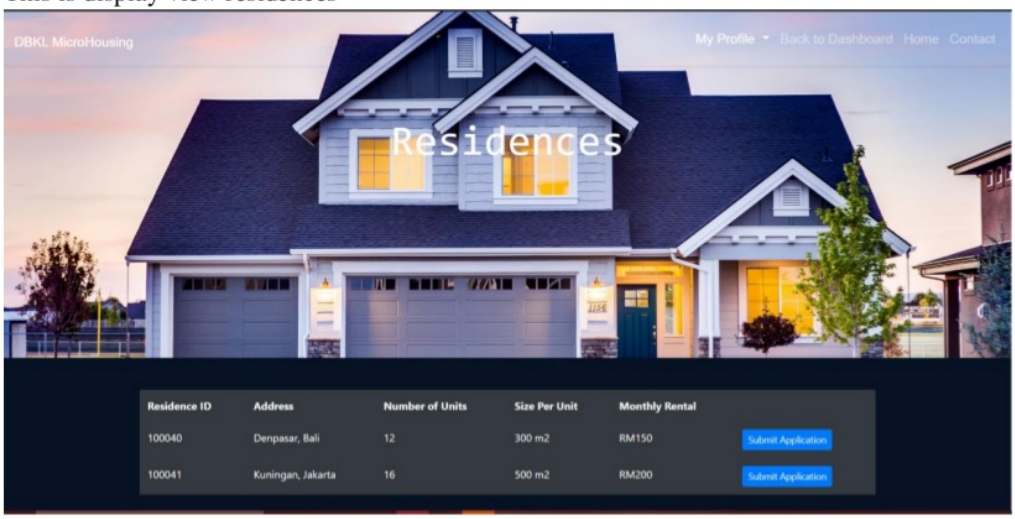
   This is form for the view residence in the applicant role

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Residences</title>
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/bootstrap.css') }}">
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/bootstrap.min.css') }}">
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/dbkl.css') }}">
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/animate.css') }}">
    </head>
    <body>
        <nav class="navbar navbar-expand-sm navbar-dark fixed-top shadow-sm" role="navigation">
            <h2><a class="navbar-brand" href="/dashboard_user">
                DBKL MicroHousing
            </a></h2>
                <div class="navbar-header page-scroll">
                    <button type="button" class="navbar-toggler" data-toggle="collapse" data-target=".navbar-main-collapse">
                        <span class="navbar-toggler-icon"></span>
                    </button>
                </div>
                <div class="collapse navbar-collapse navbar-main-collapse animate slideIn" id="NavNavbar">
                    <ul class="navbar-nav ml-auto">
                        <li class="nav-link nav-item dropdown">
                            <a class="dropdown-toggle" href="#" aria-haspopup="true" aria-expanded="false" id="navbardrop" data-toggle="dropdown">
                                <span class="caret">
                                My Profile
                                </span>
                            </a>
                            <div class="dropdown-menu animate slideIn">
                                <a class="dropdown-item" href="#" data-toggle="modal" data-target="#myModal">Account Details</a>
                                <a class="dropdown-item" href="#">Change Password</a>
                                <a class="dropdown-item" href="{{ route('logout') }}"onclick="event.preventDefault();document.getElementById('logo
                                    {{ __('Logout') }}</a>
                                <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">@csrf</form>
                            </div>
                        </li>
```

This function is to call the table application record in the database

```
public function ViewApplication()
{
    $application = DB::table('application')->get();
    return view('Applicant.dashboard_user',['application' => $application]);
}
```

This is display view residences



10. View Application

This test ensure the applications that are display the viewapplication which is the function in the applicant folder. The applicantcontroller functioning properly. In this code the applicantID will be login first, the the applicant id will look for, and the application will display the application that has the same applicantID thas has login

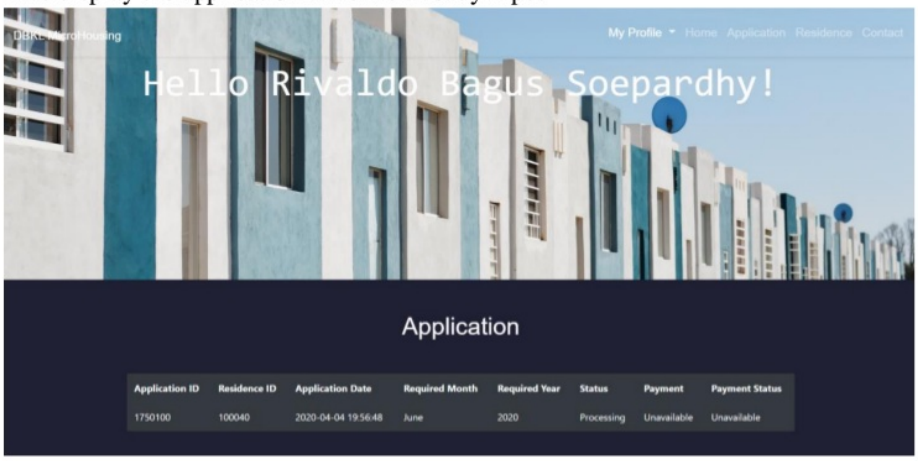
This is the form table will be display in the applicant dashboard

```
<div id="application" class="row p-5">
    <div class="container">
        <div class="row">
            <div class="title-aboutus">
                Application
            </div>
        </div>
        <br>
        <br>
        <div class="row">
            <div class="col">
                <table class="table table-borderless table-hover table-dark">
                    <tr>
                        <th>Application ID</th>
                        <th>Residence ID</th>
                        <th>Application Date</th>
                        <th>Required Month</th>
                        <th>Required Year</th>
                        <th>Status</th>
                        <th>Payment</th>
                        <th>Payment Status</th>
                    </tr>
                    @foreach($application as $app)
                        @if($app->userID==Auth::user()->id)
                    <tr>
                        <td>{{ $app->applicationID}}</td>
                        <td>{{ $app->residenceID}}</td>
                        <td>{{ $app->applicationDate}}</td>
                        <td>{{ $app->requiredMonth}}</td>
                        <td>{{ $app->requiredYear}}</td>
                        <td>{{ $app->status}}</td>
                        <td>{{ $app->payment}}</td>
                        <td>{{ $app->payment_status}}</td>
                    </tr>
                        @endif
                    @endforeach
```

This function is to call the applications which is in the database and already input by the applicant.

```
public function ViewApplication()
{
    $application = DB::table('application')->get();
    return view('Applicant.dashboard_user',['application' => $application]);
}
```

This display the application which is already input



For all the function above, we are using route it is to handle our request in the Url and then direct the application to call a specific page / resource which is route to housing officer page and applicant page

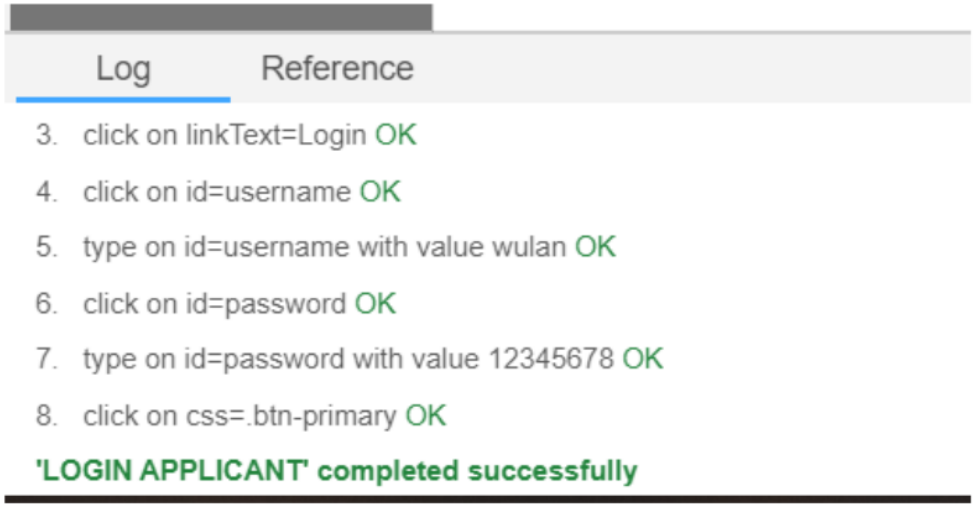
```
Route::get('/', function () {
    return view('homepage');
});

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

Route::group(['middleware'=>['auth','Housingofficer']],function(){
    Route::get('/dashboard', function () {
        return view('Housingofficer.dashboard');
    });
    Route::get('/viewresidences','Housingofficer\DashboardController@viewresidences');
    Route::get('/editresidences{residenceID}','Housingofficer\DashboardController@editresidences');
    Route::put('/updateresidences{residenceID}','Housingofficer\DashboardController@updateresidences');
    Route::get('/viewapplications','Housingofficer\DashboardController@viewapplication');
    Route::get('/addresidences','Housingofficer\DashboardController@addresidences');
    Route::post('/addresidences','Housingofficer\DashboardController@store')->name('addedData');
});

Route::group(['middleware'=>['auth','Applicant']],function(){
    Route::get('/dashboard_user', function () {
        return view('Applicant.dashboard_user');
    });
    Route::get('/dashboard_user','ApplicantController@ViewApplication');
    Route::get('/ViewResidence','ApplicantController@ViewResidence');
    Route::get('/SubmitApplication/{residenceID}','ApplicantController@SubmitApplication');
    Route::post('/store','ApplicantController@store');
});
```

Integration Testing

We use Selenium-Web Driver application. The system will run based on the functions existed in each use case. The result of the testing will be exported in a Python .py file.

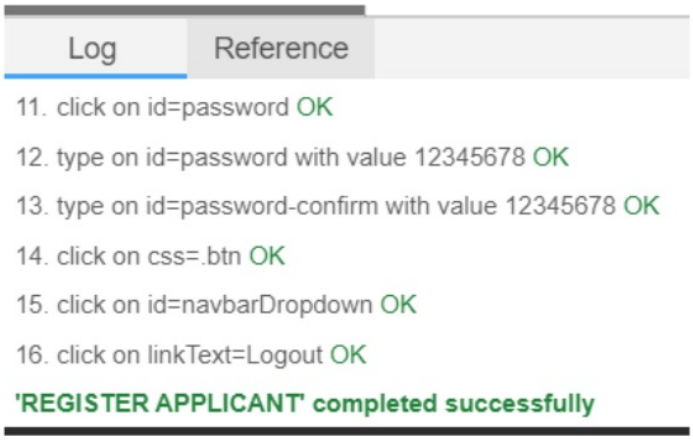
1. Login Applicant

```
def test_lOGINAPPLICANT(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(783, 824)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.ID, "username").click()
  self.driver.find_element(By.ID, "username").send_keys("wulan")
  self.driver.find_element(By.ID, "password").click()
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
```



2. Register Applicant

```
def test_rEGISTERAPPLICANT(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(1552, 840)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.LINK_TEXT, "Register").click()
  self.driver.find_element(By.ID, "name").click()
  self.driver.find_element(By.ID, "name").send_keys("foo")
  self.driver.find_element(By.ID, "username").send_keys("foo by foo")
  self.driver.find_element(By.ID, "email").send_keys("foo@gmail.com")
  self.driver.find_element(By.ID, "monthlyincome").click()
  self.driver.find_element(By.ID, "monthlyincome").send_keys("800")
  self.driver.find_element(By.ID, "password").click()
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.ID, "password-confirm").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn").click()
  self.driver.find_element(By.ID, "navbarDropdown").click()
  self.driver.find_element(By.LINK_TEXT, "Logout").click()
```



3. Submit Application – Applicant

```
def test_sUBMITAPP(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(1552, 840)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.ID, "username").click()
  self.driver.find_element(By.ID, "username").send_keys("wulan")
  self.driver.find_element(By.ID, "password").click()
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
  self.driver.find_element(By.LINK_TEXT, "Home").click()
  self.driver.find_element(By.ID, "navbardrop").click()
  self.driver.find_element(By.LINK_TEXT, "Account Details").click()
  self.driver.find_element(By.CSS_SELECTOR, ".close").click()
  self.driver.find_element(By.LINK_TEXT, "Application").click()
  self.driver.find_element(By.LINK_TEXT, "Residence").click()
  self.driver.find_element(By.LINK_TEXT, "Contact").click()
  self.driver.find_element(By.LINK_TEXT, "Residence").click()
  self.driver.find_element(By.CSS_SELECTOR, ".btn-success").click()
  self.driver.find_element(By.CSS_SELECTOR, ".ibg-bg").click()
  self.driver.find_element(By.LINK_TEXT, "Submit Application").click()
  self.driver.find_element(By.NAME, "requiredMonth").click()
  self.driver.find_element(By.NAME, "requiredMonth").send_keys("December")
  self.driver.find_element(By.NAME, "requiredYear").send_keys("2021")
  self.driver.find_element(By.CSS_SELECTOR, ".form-inline").click()
  self.driver.find_element(By.CSS_SELECTOR, ".btn-sm").click()
  self.driver.find_element(By.ID, "navbardrop").click()
  self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

Log Reference

21. type on name=requiredMonth with value December OK
22. type on name=requiredYear with value 2021 OK
23. click on css=.form-inline OK
24. click on css=.btn-sm OK
25. click on id=navbardrop OK
26. Trying to find linkText=Logout... OK

'SUBMIT APP' completed successfully

applicationID	applicationDate	requiredMonth	requiredYear	status	residenceID	userID
	15:15:15	June	2020	Processing	100040	2020101
	16:09:09	September	2021	Processing	100040	2020103
1750102	2020-04-04 18:32:41	August	2020	Processing	100041	2020104
1750105	2020-04-04 19:26:17	December	2021	Processing	100040	2020101

Click the drop-down arrow to toggle column's visibility.

4. Login & Logout Housing Officer

```
def test_lOGINLOGOUTADMIN(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(1030, 543)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.ID, "username").click()
  self.driver.find_element(By.ID, "username").send_keys("aldobagus")
  self.driver.find_element(By.CSS_SELECTOR, ".form-group:nth-child(3)").click()
  self.driver.find_element(By.ID, "password").click()
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
  self.driver.find_element(By.ID, "navbarDropdown").click()
  self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

6. click on css=.form-group:nth-child(3) OK
7. click on id=password OK
8. type on id=password with value 12345678 OK
9. click on css=.btn-primary OK
10. click on id=navbarDropdown OK
11. click on linkText=Logout OK

'LOGIN & LOGOUT ADMIN' completed successfully

5. Set Up New Residence – HousingOfficer

```
def test_setUpNewResidence(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(1030, 543)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.ID, "username").click()
  self.driver.find_element(By.ID, "username").send_keys("aldobagus")
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
  self.driver.find_element(By.CSS_SELECTOR, ".nav-item:nth-child(3) p").click()
  self.driver.find_element(By.NAME, "address").click()
  self.driver.find_element(By.NAME, "address").send_keys("Jakarta")
  self.driver.find_element(By.NAME, "numunits").send_keys("10")
  self.driver.find_element(By.NAME, "sizeperunits").send_keys("600 m2")
  self.driver.find_element(By.NAME, "monthlyrental").click()
  self.driver.find_element(By.NAME, "monthlyrental").send_keys("3000")
  self.driver.find_element(By.NAME, "submit").click()
  self.driver.find_element(By.ID, "navbarDropdown").click()
  self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

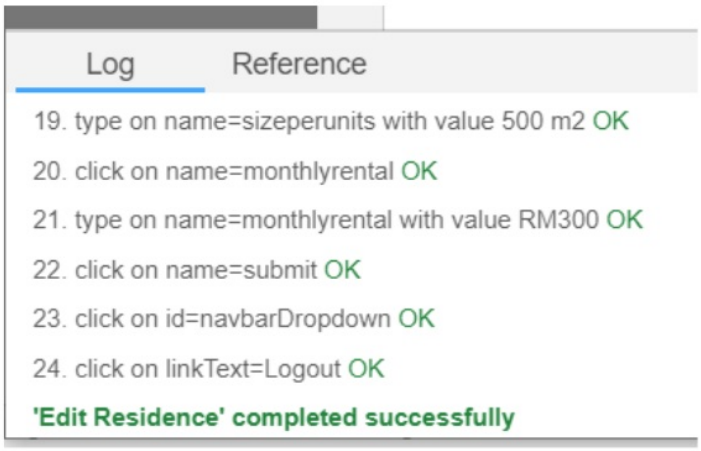
Log Reference

12. type on name=sizeperunits with value 600 m2 OK
13. click on name=monthlyrental OK
14. type on name=monthlyrental with value 3000 OK
15. click on name=submit OK
16. click on id=navbarDropdown OK
17. click on linkText=Logout OK

'SetUpNewResidence' completed successfully

6. Edit Residence – HousingOfficer

```
def test_editResidence(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(1030, 543)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.ID, "username").click()
  self.driver.find_element(By.ID, "username").send_keys("aldobagus")
  self.driver.find_element(By.ID, "password").click()
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
  self.driver.find_element(By.CSS_SELECTOR, ".nav-item:nth-child(4) > .nav-link").click()
  self.driver.find_element(By.CSS_SELECTOR, "tr:nth-child(2) .btn").click()
  self.driver.find_element(By.NAME, "address").click()
  self.driver.find_element(By.NAME, "address").click()
  self.driver.find_element(By.NAME, "address").click()
  self.driver.find_element(By.NAME, "numunits").click()
  self.driver.find_element(By.NAME, "numunits").send_keys("12")
  self.driver.find_element(By.NAME, "sizeperunits").click()
  self.driver.find_element(By.NAME, "sizeperunits").click()
  self.driver.find_element(By.NAME, "sizeperunits").click()
  self.driver.find_element(By.NAME, "sizeperunits").send_keys("500 m2")
  self.driver.find_element(By.NAME, "monthlyrental").click()
  self.driver.find_element(By.NAME, "monthlyrental").send_keys("RM300")
  self.driver.find_element(By.NAME, "submit").click()
  self.driver.find_element(By.ID, "navbarDropdown").click()
  self.driver.find_element(By.LINK_TEXT, "Logout").click()
```



System Testing

Functional Requirements

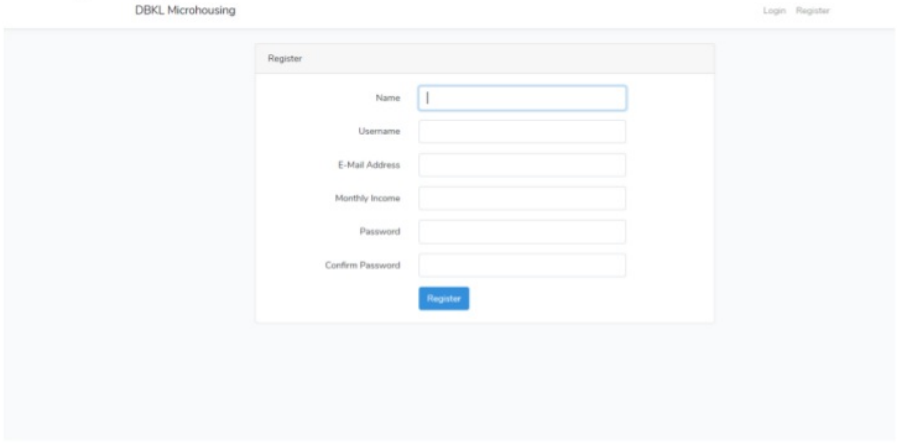
We will test the system according to the functional requirements.

1. **Register for Applicant**

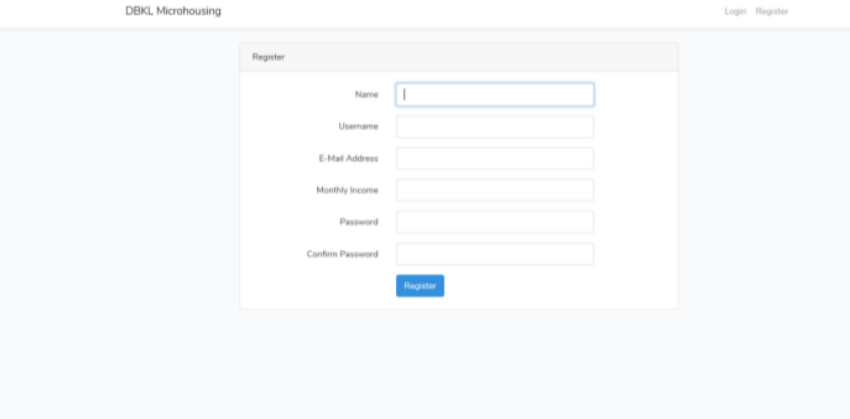
Testing start from the dashboard which is you enter the URL this is the dashboard page and register page.

The dashboard page

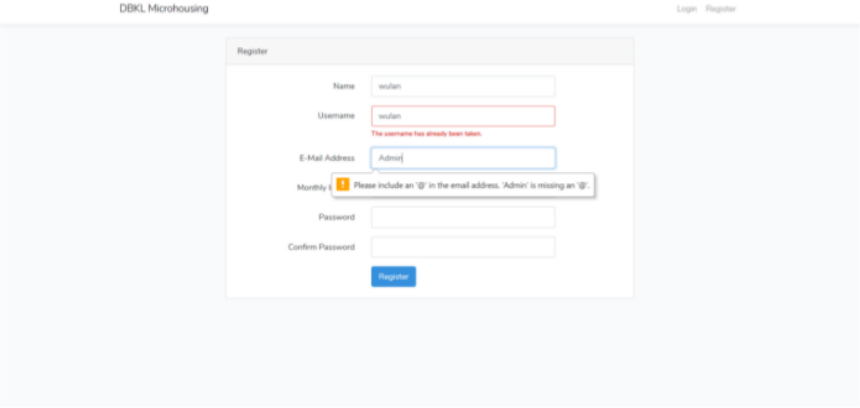
The register page



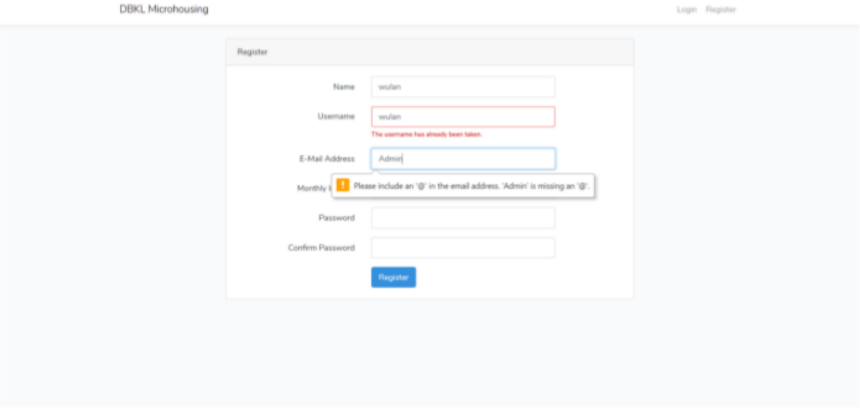
a. Applicant input blank information

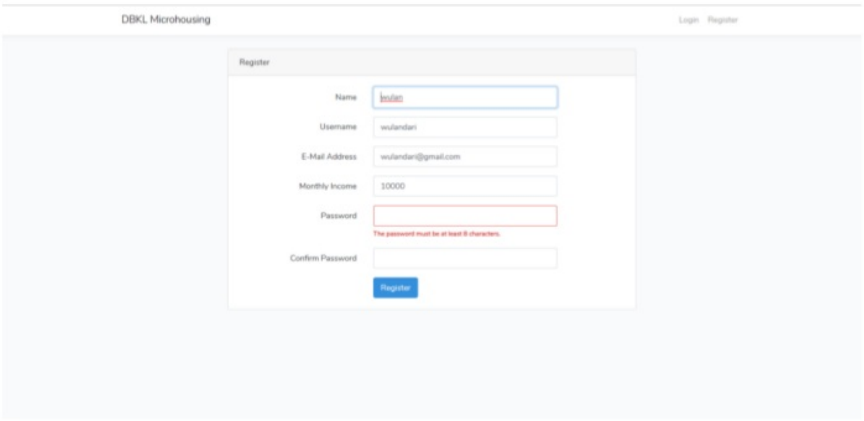


b. Applicant input wrong email or username which is already used



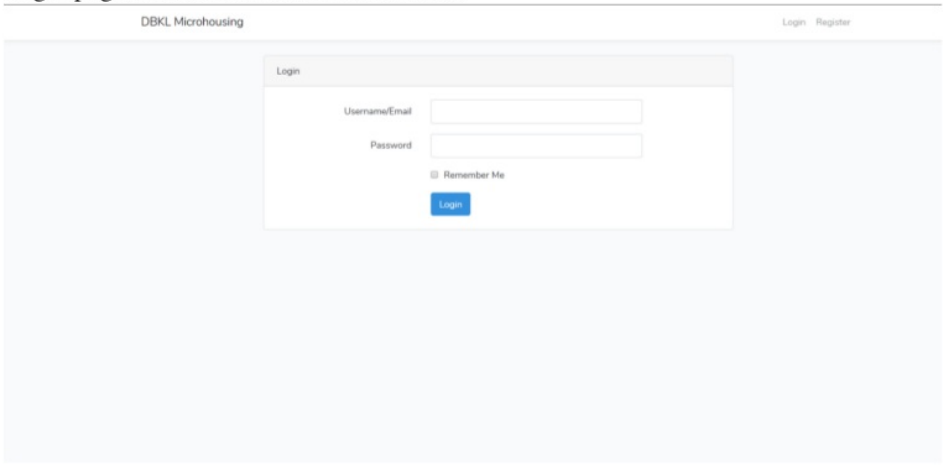
c. Applicant input invalid email and password (too short)



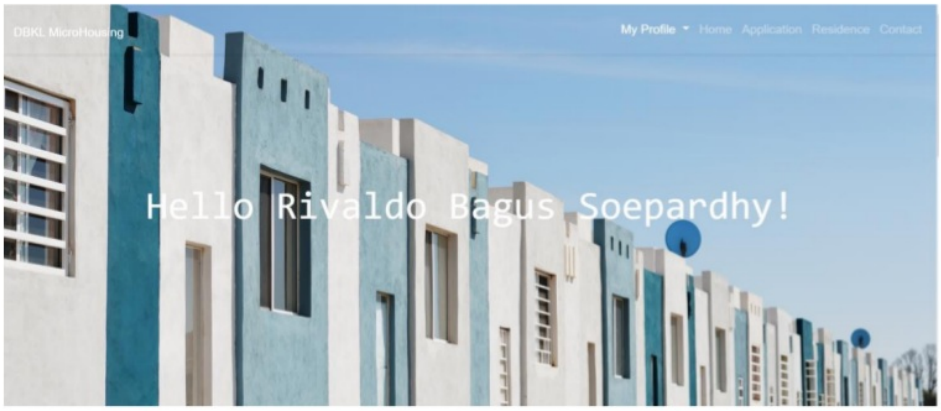


2. **Login**

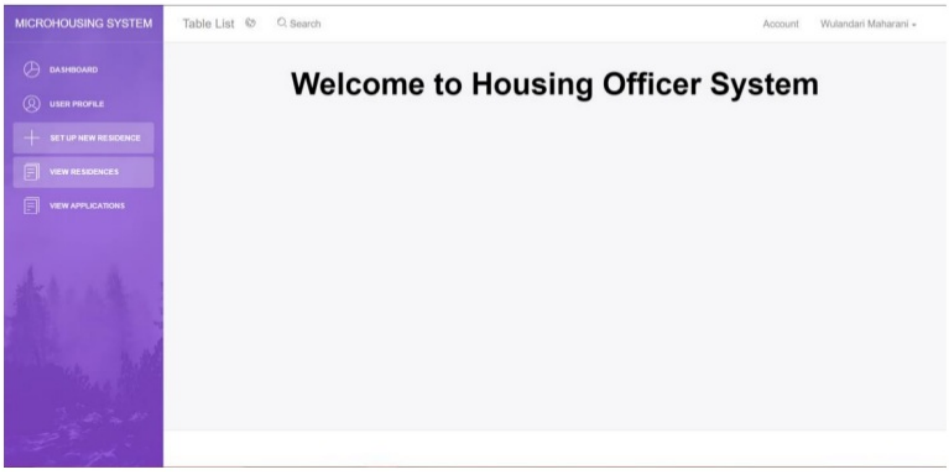
Login page same location in the dashboard



Applicant Homepage

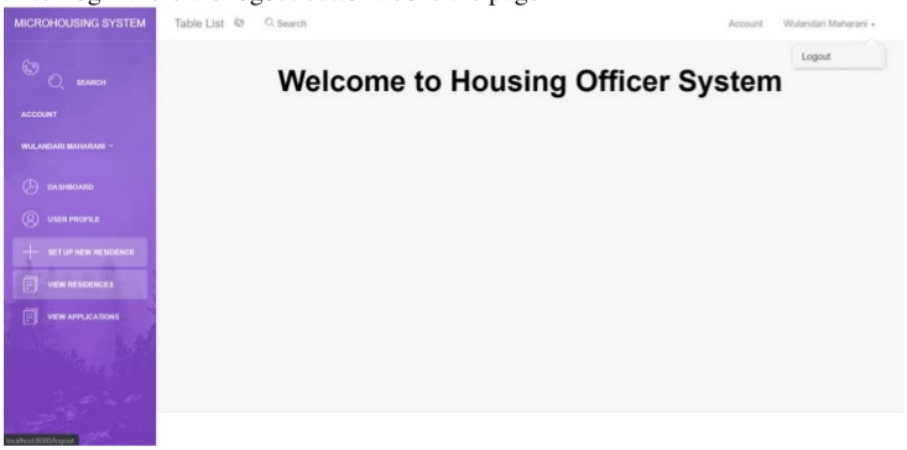


Housing Officer Homepage



3. LogOut

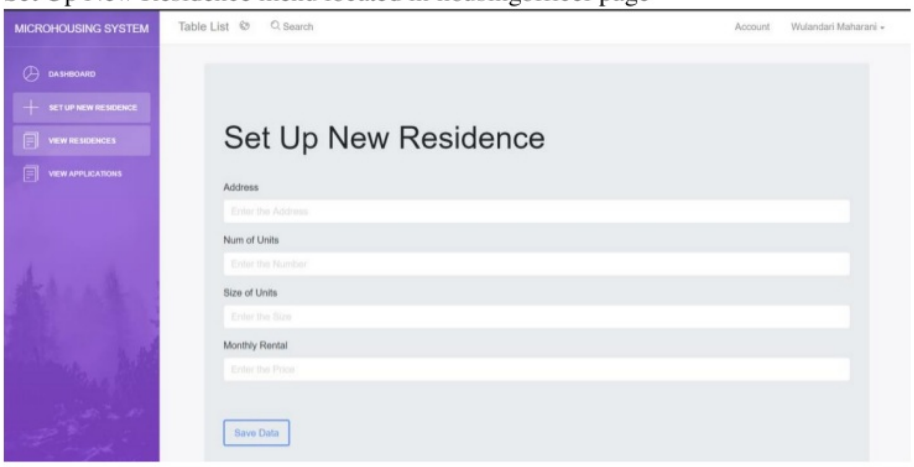
After login there are logout button above the page



Back to the dashboard

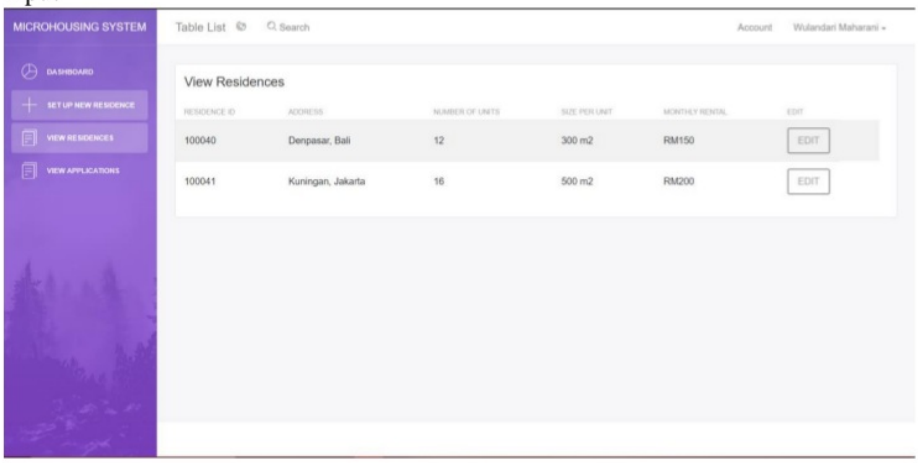
4. **Set Up New Residence**

Set Up New Residence menu located in housingofficer page



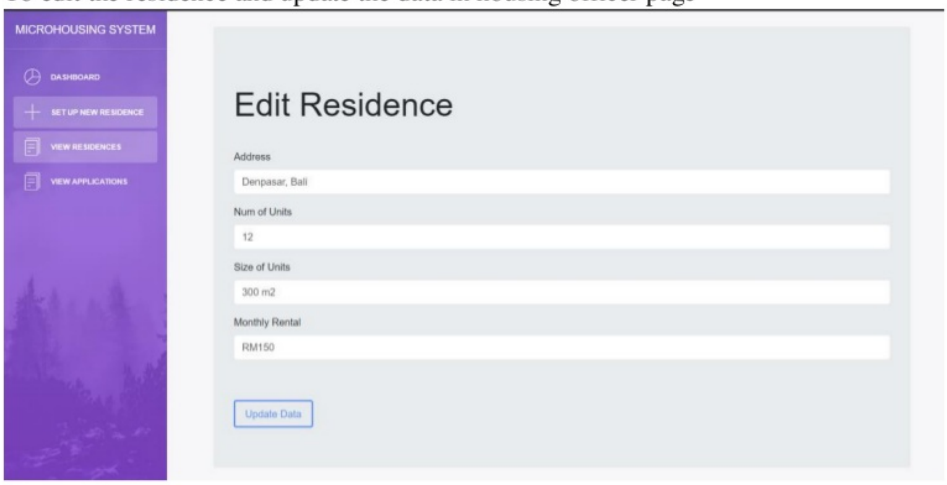
5. **View Residence Detail**

To view residence, which does the housing officer in the housing officer page already input



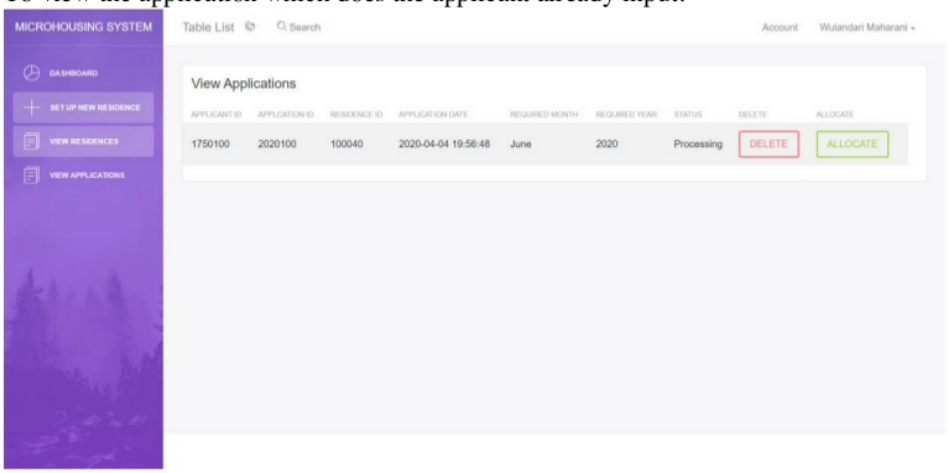
6. **Edit Residence Detail**

To edit the residence and update the data in housing officer page



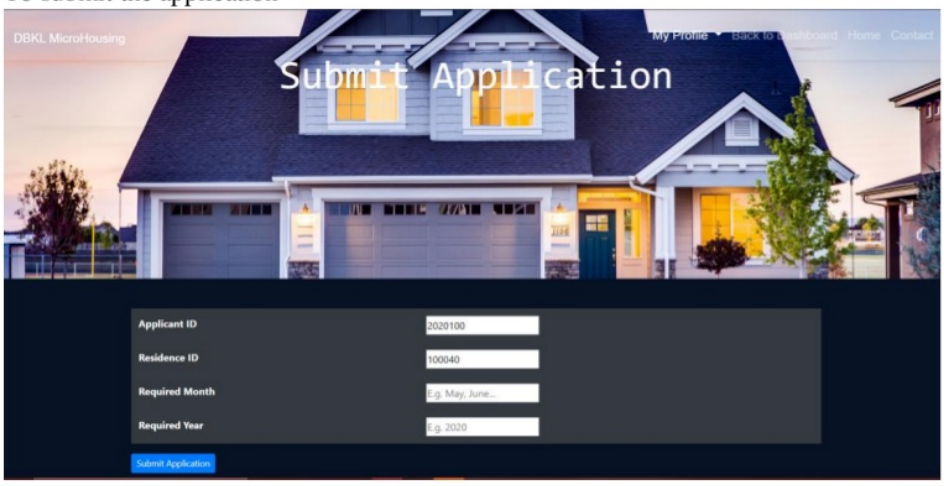
7. **View Applications**

To view the application which does the applicant already input.

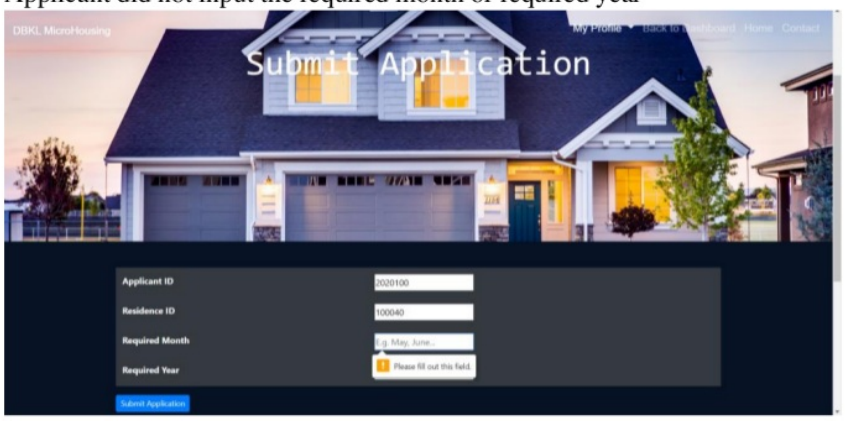


8. Submit Application

To submit the application

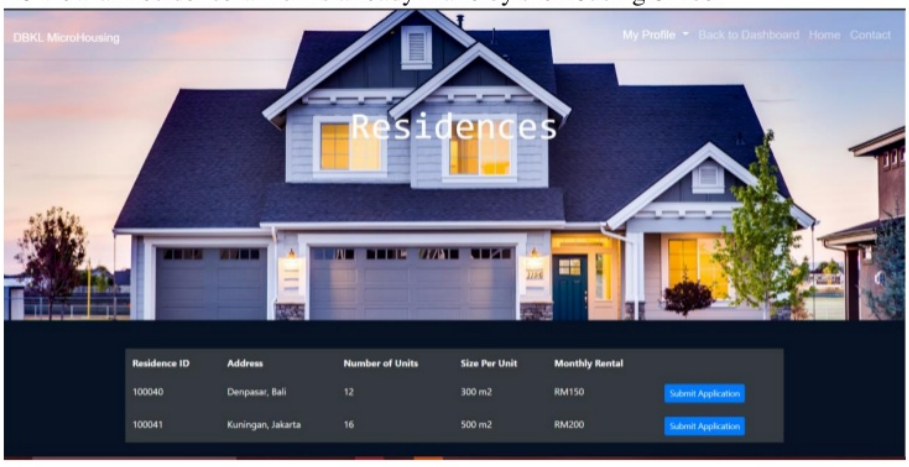


a. Applicant did not input the required month or required year



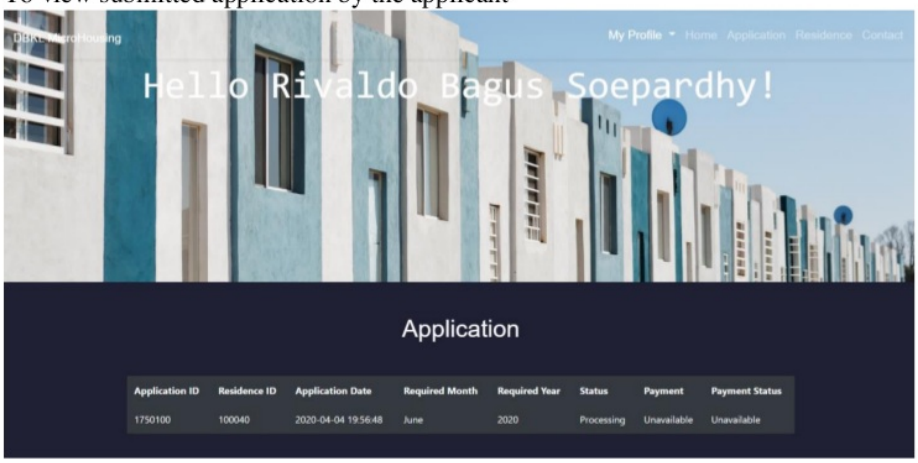
9. View Residence (Applicant)

To view all residence which is already make by the housing officer



10. View Applications

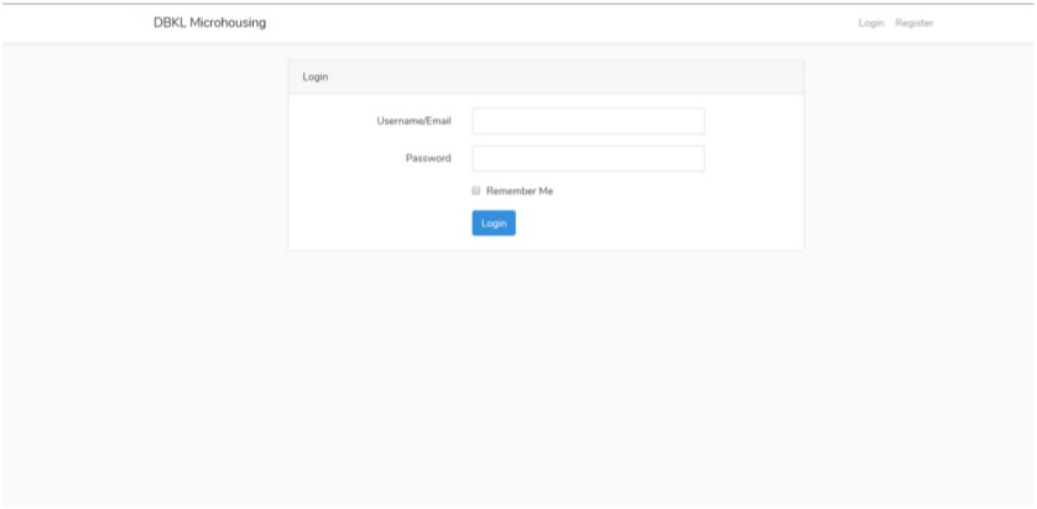
To view submitted application by the applicant



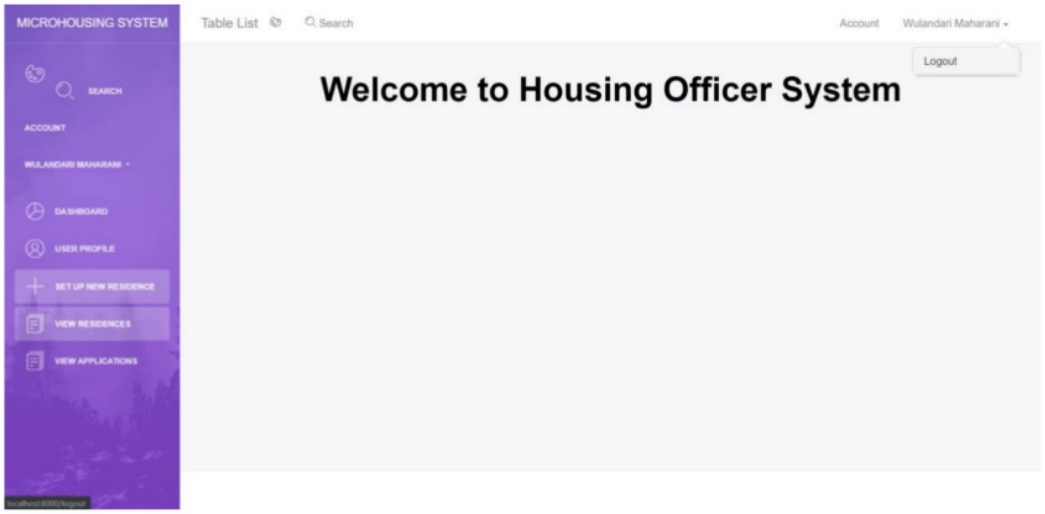
Non-functional Requirements

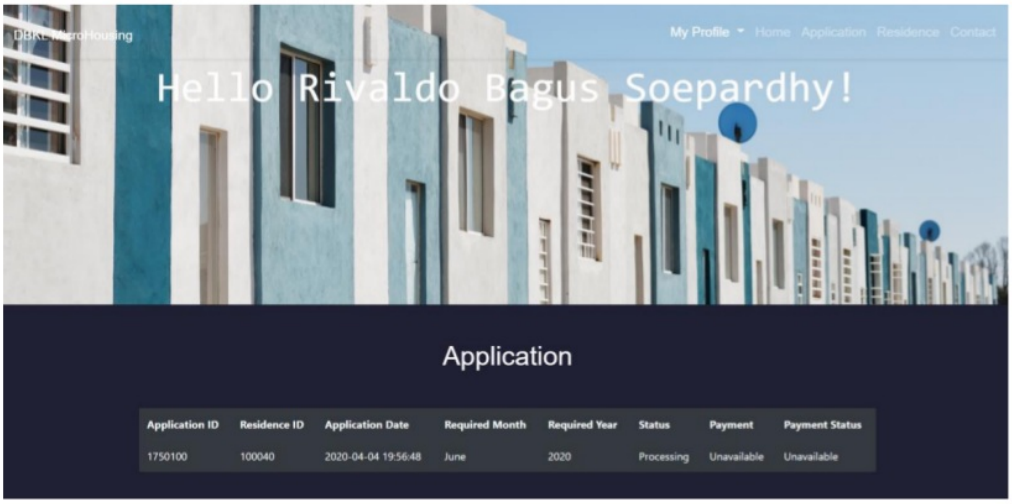
We will test the system according to the non-functional requirements that have been determined previously.

1. Security requirement: using login system for authorization to prevent unauthorized access of certain parties.



2. Usability: the system should be easy to access for Housing Officer and Applicant.





3. Accuracy and precision: Integrity which data has been inputted will still be the data entered. Data will be stored with it's real value.

```
def test_sUBMITAPP(self):
  self.driver.get("http://localhost:8000/")
  self.driver.set_window_size(1552, 840)
  self.driver.find_element(By.LINK_TEXT, "Login").click()
  self.driver.find_element(By.ID, "username").click()
  self.driver.find_element(By.ID, "username").send_keys("wulan")
  self.driver.find_element(By.ID, "password").click()
  self.driver.find_element(By.ID, "password").send_keys("12345678")
  self.driver.find_element(By.CSS_SELECTOR, ".btn-primary").click()
  self.driver.find_element(By.LINK_TEXT, "Home").click()
  self.driver.find_element(By.ID, "navbardrop").click()
  self.driver.find_element(By.LINK_TEXT, "Account Details").click()
  self.driver.find_element(By.CSS_SELECTOR, ".close").click()
  self.driver.find_element(By.LINK_TEXT, "Application").click()
  self.driver.find_element(By.LINK_TEXT, "Residence").click()
  self.driver.find_element(By.LINK_TEXT, "Contact").click()
  self.driver.find_element(By.LINK_TEXT, "Residence").click()
  self.driver.find_element(By.CSS_SELECTOR, ".btn-success").click()
  self.driver.find_element(By.CSS_SELECTOR, ".ibg-bg").click()
  self.driver.find_element(By.LINK_TEXT, "Submit Application").click()
  self.driver.find_element(By.NAME, "requiredMonth").click()
  self.driver.find_element(By.NAME, "requiredMonth").send_keys("December")
  self.driver.find_element(By.NAME, "requiredYear").send_keys("2021")
  self.driver.find_element(By.CSS_SELECTOR, ".form-inline").click()
  self.driver.find_element(By.CSS_SELECTOR, ".btn-sm").click()
  self.driver.find_element(By.ID, "navbardrop").click()
  self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

Log Reference

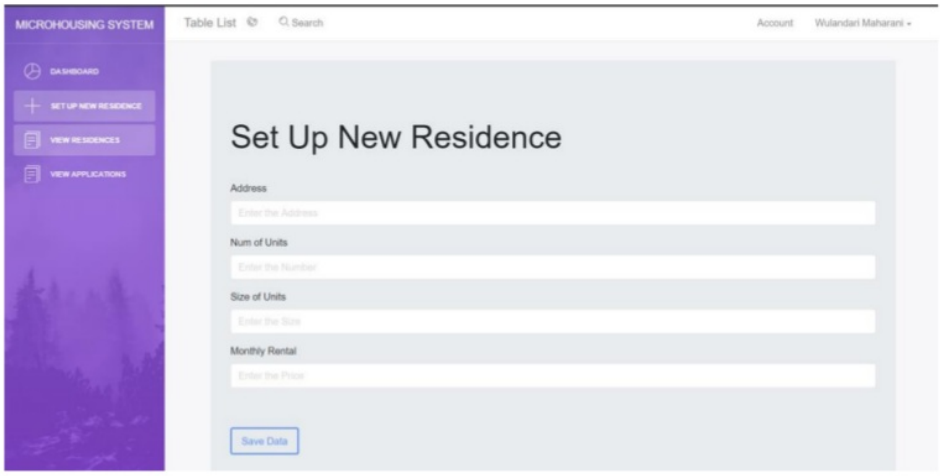
21. type on name=requiredMonth with value December OK
22. type on name=requiredYear with value 2021 OK
23. click on css=.form-inline OK
24. click on css=.btn-sm OK
25. click on id=navbardrop OK
26. Trying to find linkText=Logout... OK

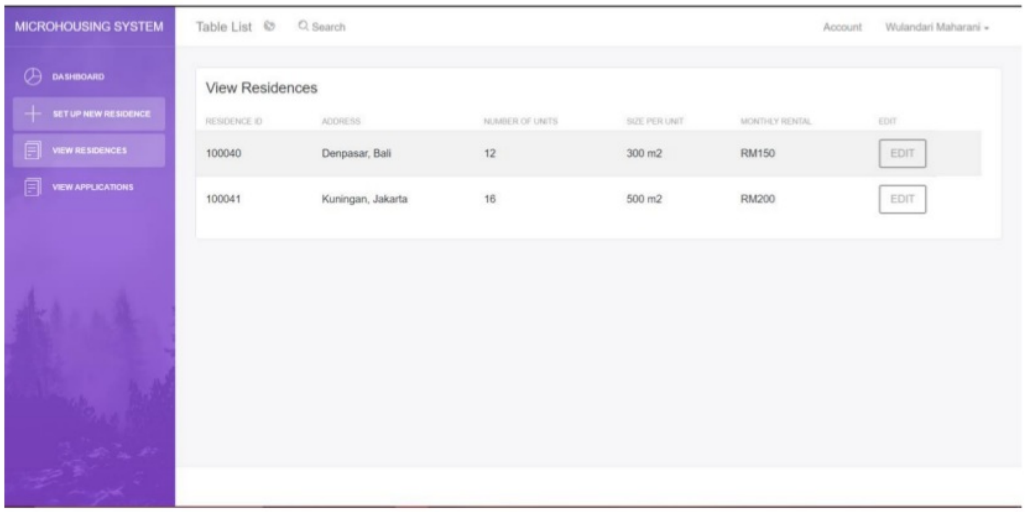
'SUBMIT APP' completed successfully

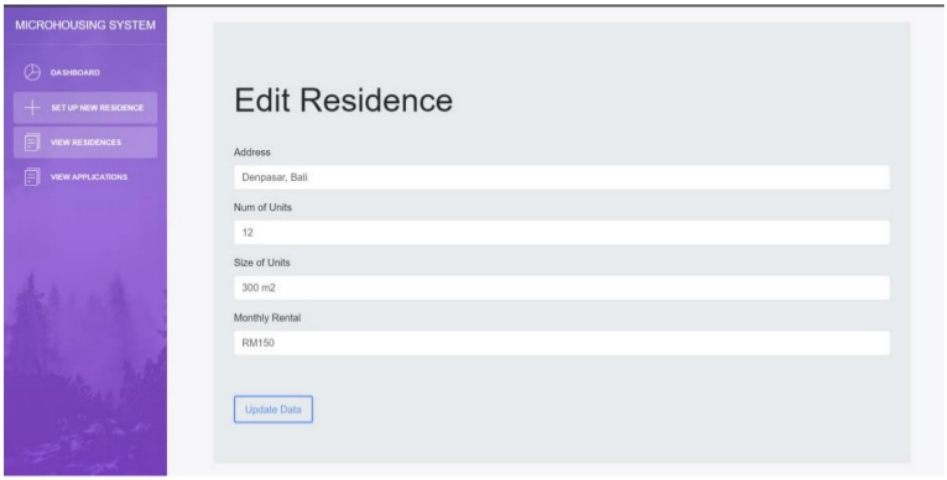
applicationID	applicationDate	requiredMonth	requiredYear	status	residenceID	userID
	15:15:15	June	2020	Processing	100040	2020101
1750101	2020-04-04 16:09:09	September	2021	Processing	100040	2020103
1750102	2020-04-04 18:32:41	August	2020	Processing	100041	2020104
1750105	2020-04-04 19:26:17	December	2021	Processing	100040	2020101

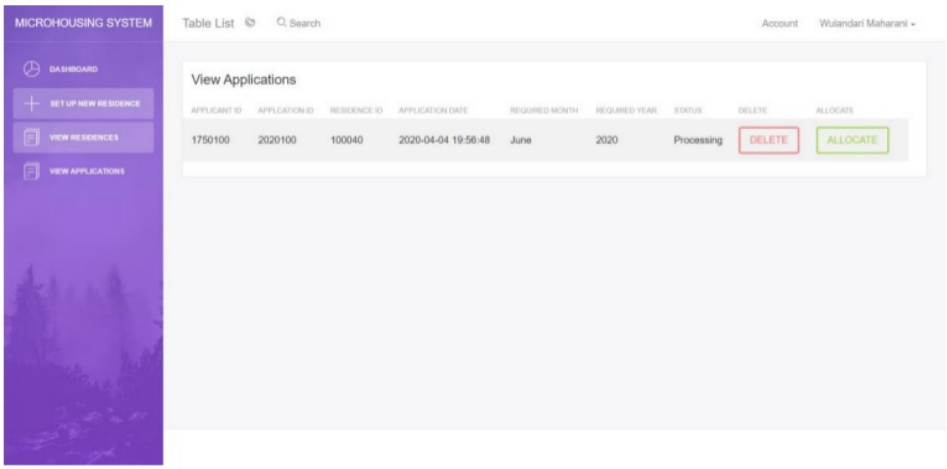
Click the drop-down arrow to toggle column's visibility.

4. Modifiability: data can only be change by authorized user (HousingOfficer).





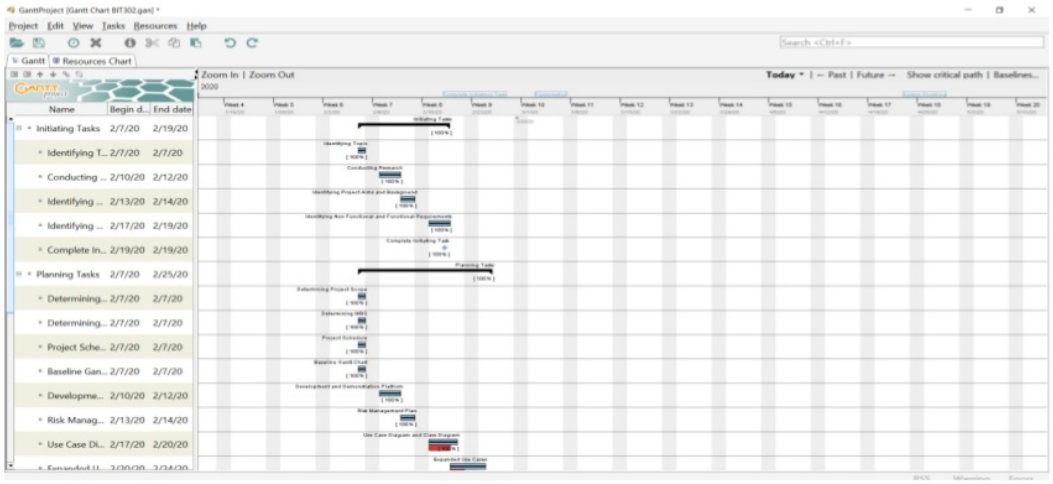


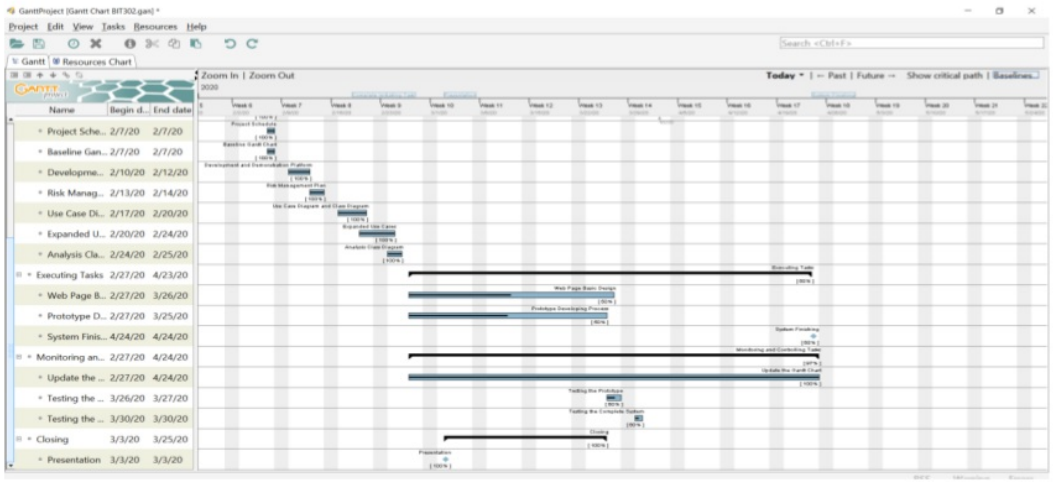


Test Analysis Report

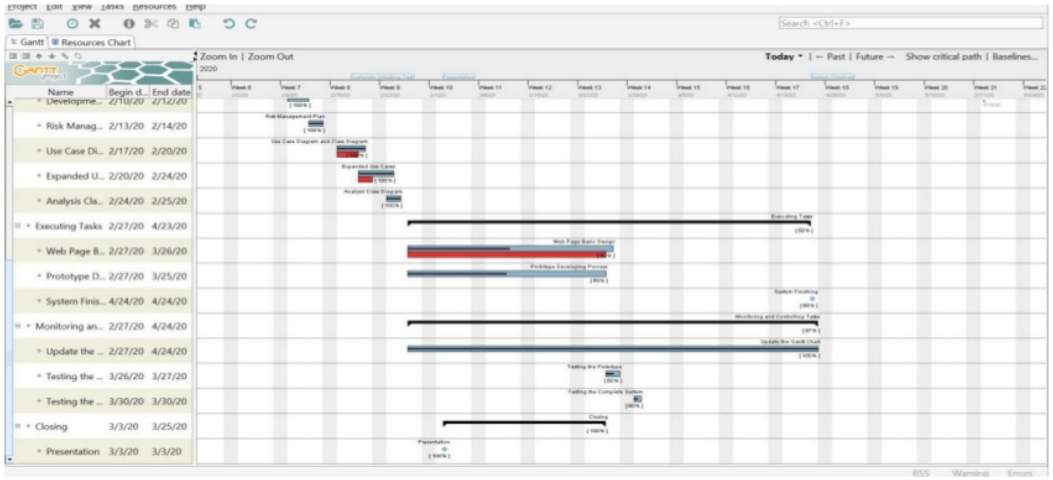
After conducting all of the required tests, we can say the system is working normally.From unit, integration and system testing, we can say that the system and codes are working fine and has passed the testing process. The scope of our testing is to make sure every functions of the system can work perfectly according what the tasks they are assigned. Overall, we have found bugs and errors and have corrected it.

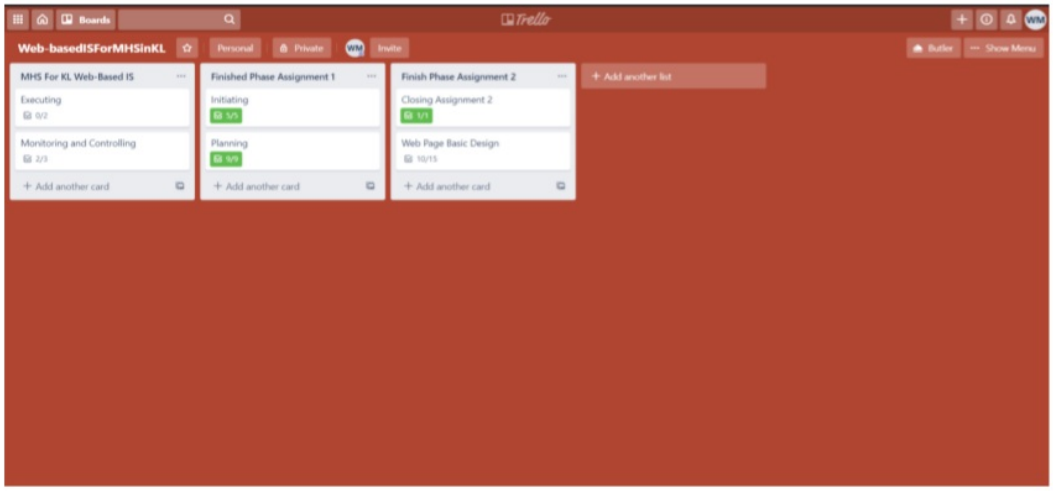
Updated Gantt Chart, Trello Board, GitHub

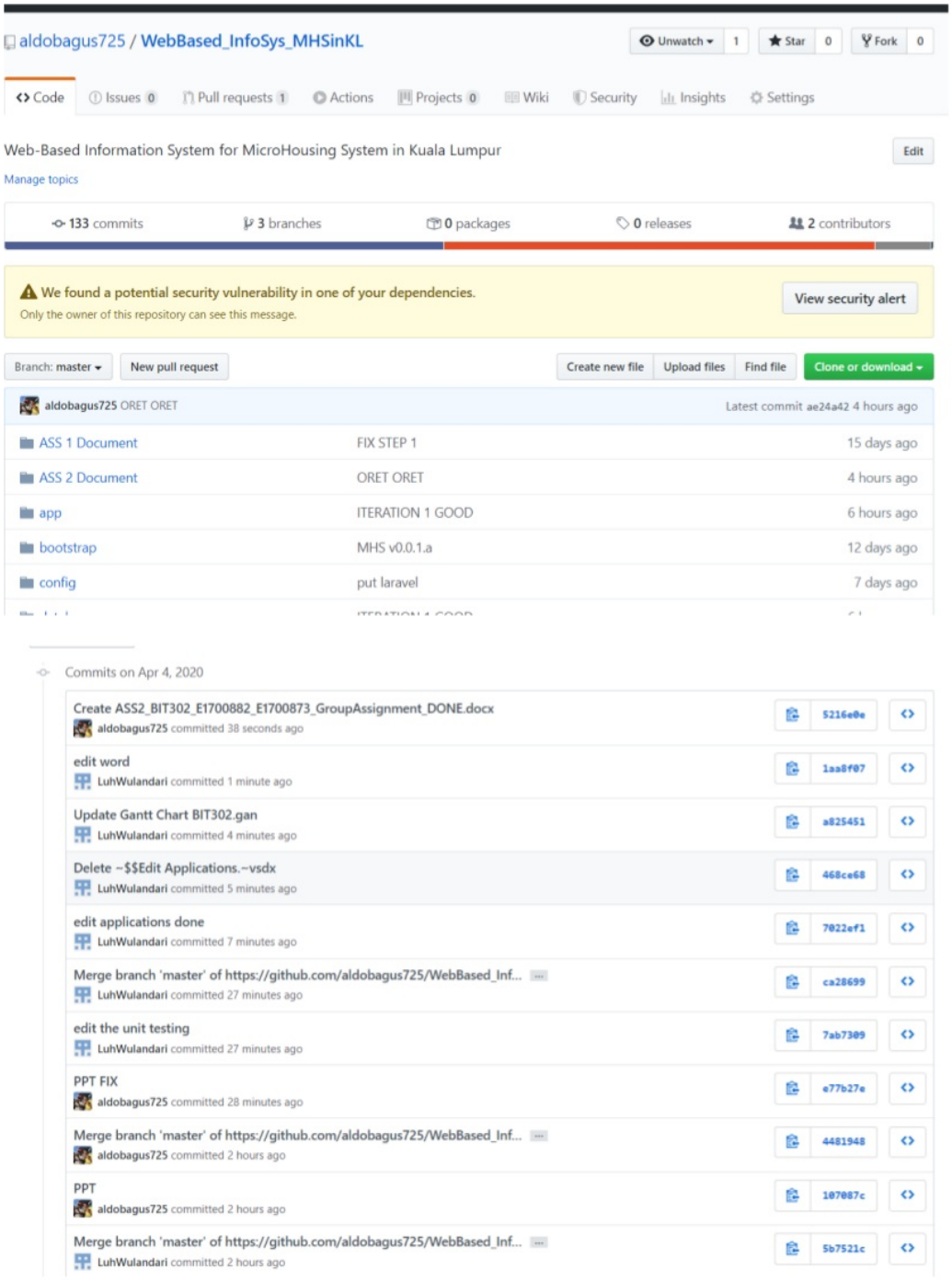




Baseline Gantt Chart






aldobagus725 / WebBased_InfoSys_MHSinKL
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 1
Actions
Projects 0
Wiki
Security
Insights
Settings
Web-Based Information System for MicroHousing System in Kuala Lumpur
Edit
Manage topics
133 commits
3 branches
0 packages
0 releases
2 contributors
We found a potential security vulnerability in one of your dependencies.
Only the owner of this repository can see this message.
View security alert
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
aldobagus725 ORET ORET
Latest commit ae24a42 4 hours ago
ASS 1 Document FIX STEP 1 15 days ago
ASS 2 Document ORET ORET 4 hours ago
app ITERATION 1 GOOD 6 hours ago
bootstrap MHS v0.0.1.a 12 days ago
config put laravel 7 days ago
Commits on Apr 4, 2020
Create ASS2_BIT302_E1700882_E1700873_GroupAssignment_DONE.docx
aldobagus725 committed 38 seconds ago
5216e0e
edit word
LuhWulandari committed 1 minute ago
1aa8f07
Update Gantt Chart BIT302.gan
LuhWulandari committed 4 minutes ago
a825451
Delete ~$$Edit Applications.~vsdx
LuhWulandari committed 5 minutes ago
468ce68
edit applications done
LuhWulandari committed 7 minutes ago
7022ef1
Merge branch 'master' of https://github.com/aldobagus725/WebBased_Inf...
LuhWulandari committed 27 minutes ago
ca28699
edit the unit testing
LuhWulandari committed 27 minutes ago
7ab7309
PPT FIX
aldobagus725 committed 28 minutes ago
e77b27e
Merge branch 'master' of https://github.com/aldobagus725/WebBased_Inf...
aldobagus725 committed 2 hours ago
4481948
PPT
aldobagus725 committed 2 hours ago
107087c
Merge branch 'master' of https://github.com/aldobagus725/WebBased_Inf...
LuhWulandari committed 2 hours ago
5b7521c

ASS2_BIT302_E1700882_E1700873_GroupAssignment

ORIGINALITY REPORT

1%	1%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	micheleguieu.com Internet Source	1%
2	Jaime P. Luque, Nuriddin Ikromov, William B. Noseworthy. "Chapter 7 Location, Location, Location", Springer Science and Business Media LLC, 2019 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		