

ASS2_BIT302_E1700882_E170 0873_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

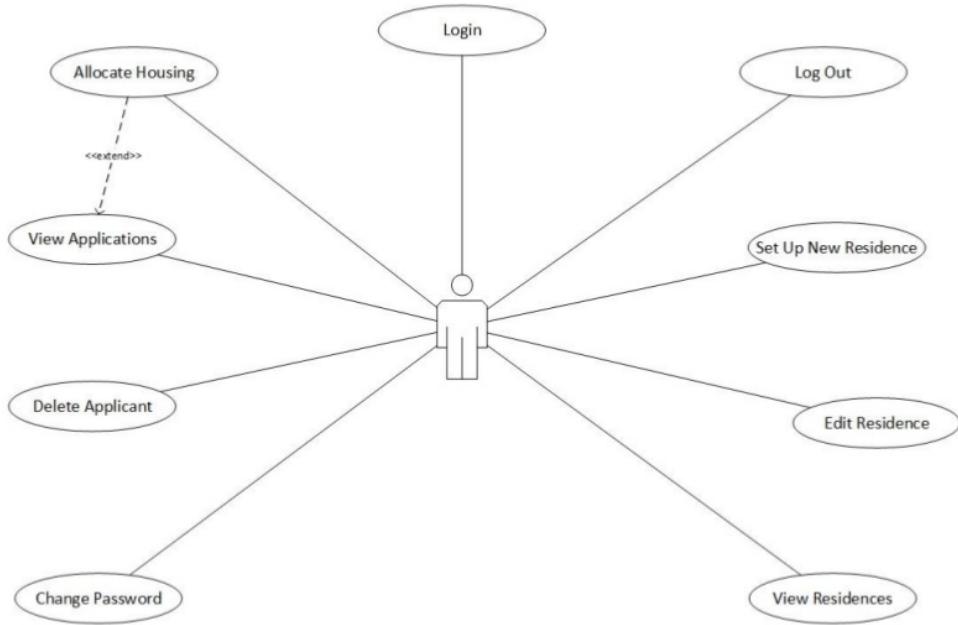
| | |
|---|----|
| Introduction | 1 |
| Use Case Diagram | 2 |
| Architectural Design | 3 |
| Class Diagram Design | 4 |
| Entity Relationship Diagram (ERD) | 5 |
| Sitemap | 6 |
| Wireframe (General UI Design)..... | 7 |
| System Sequence Diagram + Contracts | 16 |
| Database Design | 30 |
| Iteration 1 | 31 |
| Use Case..... | 31 |
| Test Objectives..... | 32 |
| Test Plan | 32 |
| Unit Testing | 33 |
| Integration Testing | 51 |
| System Testing | 52 |
| Test Analysis Report | 67 |
| Updated Gantt Chart, Trello Board, GitHub | 68 |

Introduction

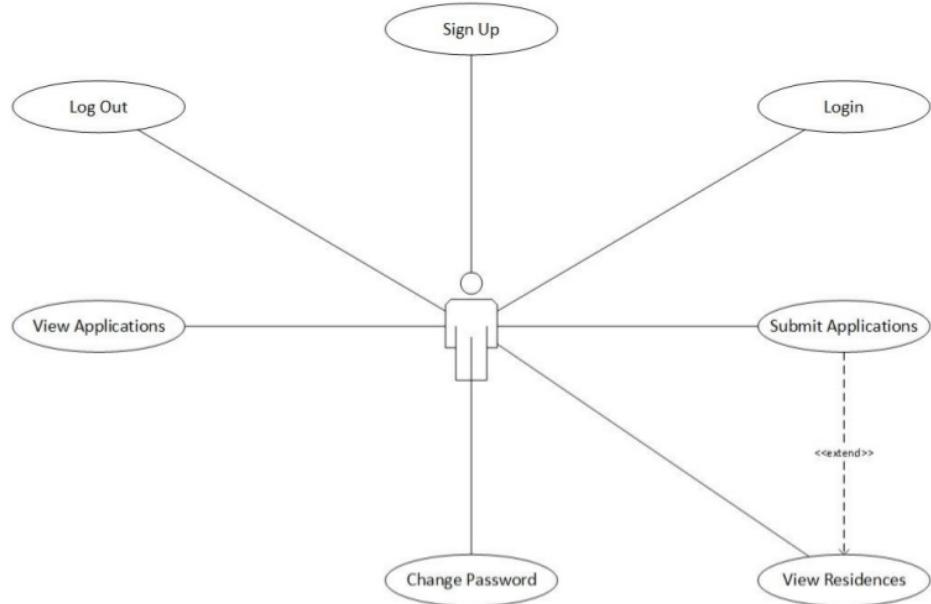
Every software development must have a clear and understandable requirement or specification to ensure developers can keep developing software within the boundary of requirement, while also, developer can adjust to what users need with the software, and developers can optimize the software according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

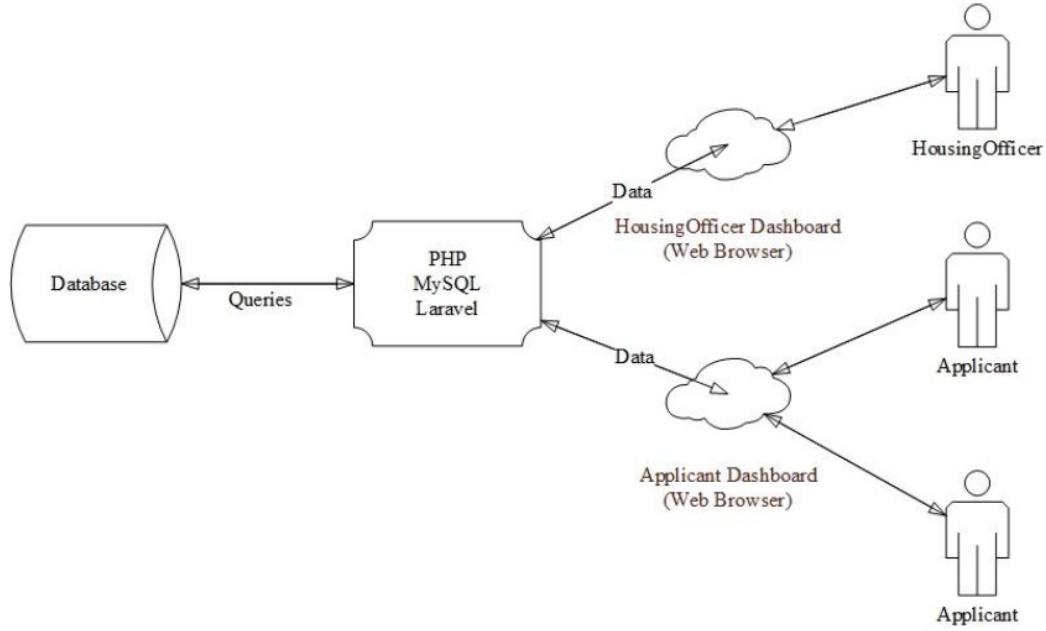


Use Case Diagram with Actor HousingOfficer

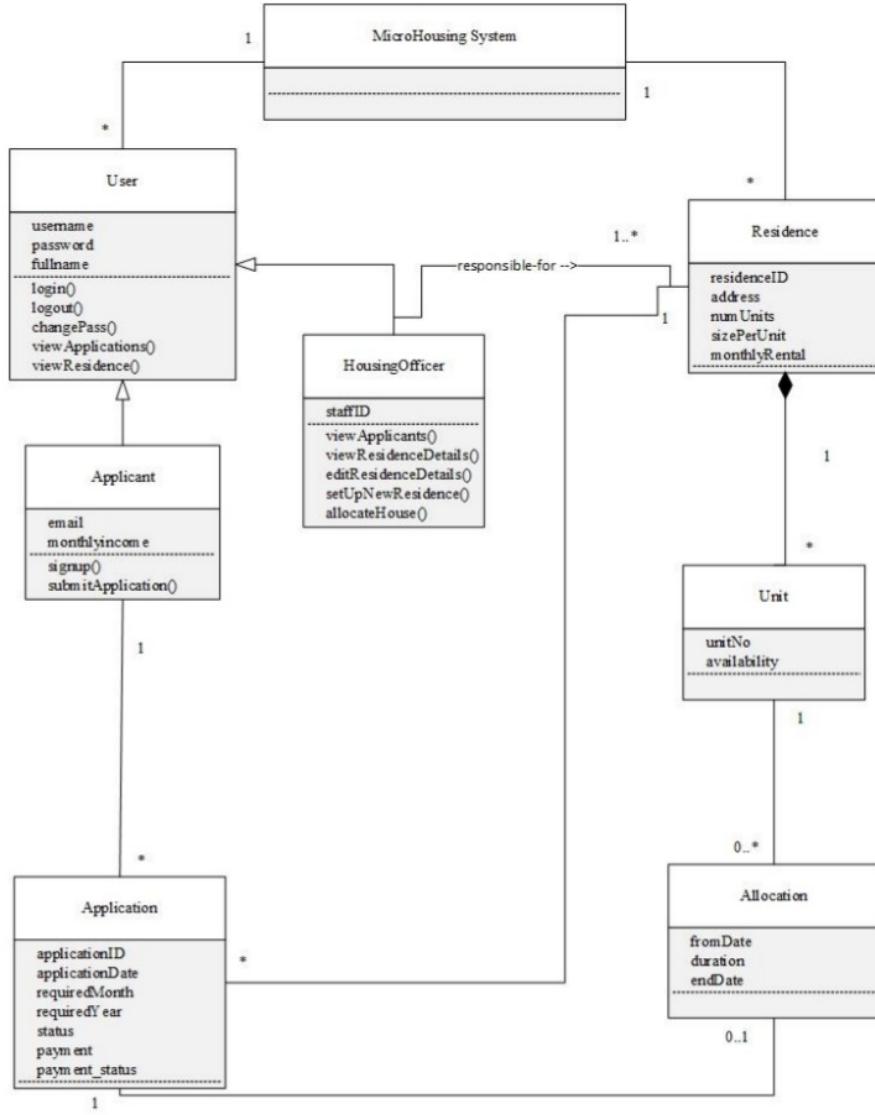


Use Case Diagram with Actor Applicant

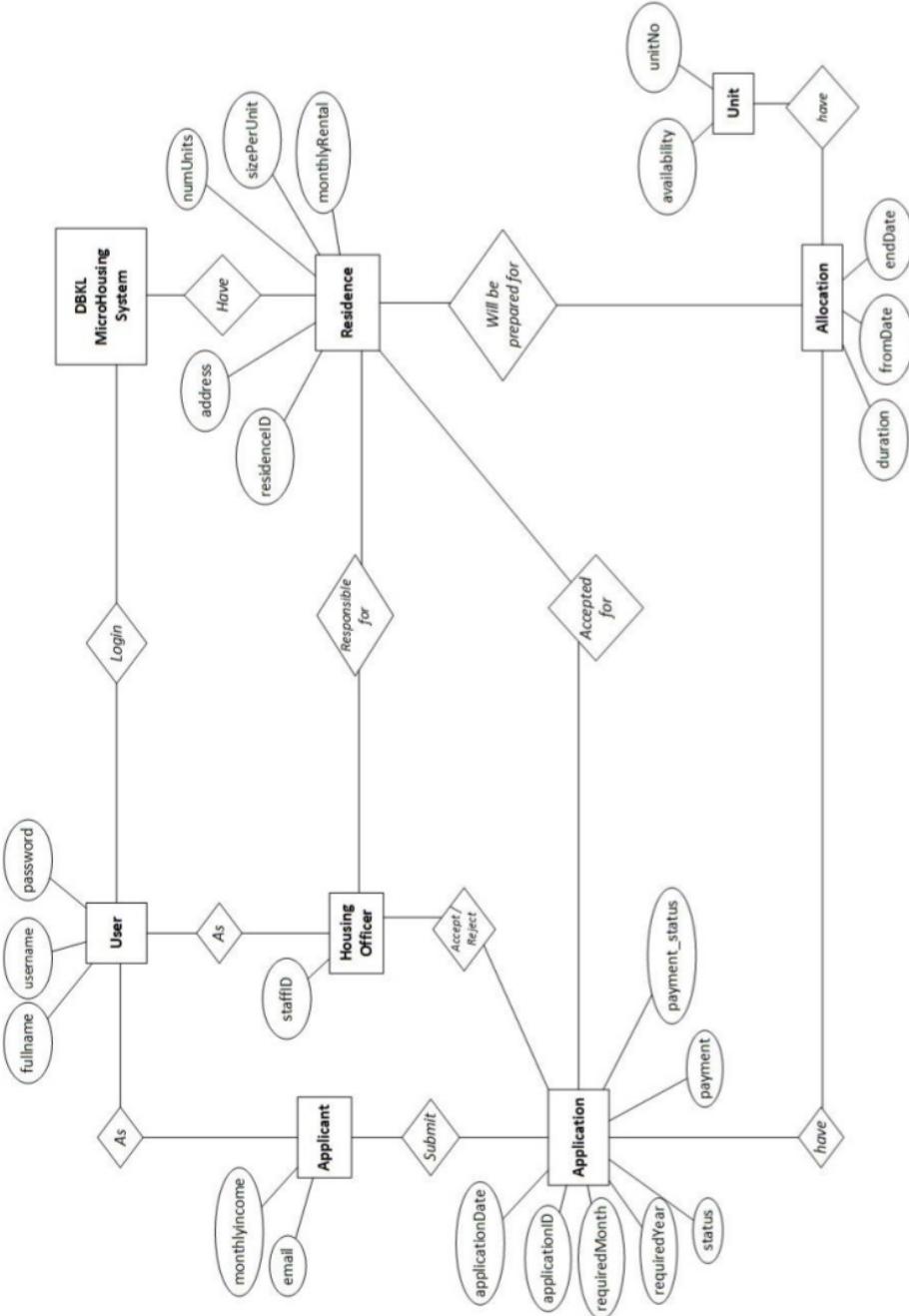
Architectural Design



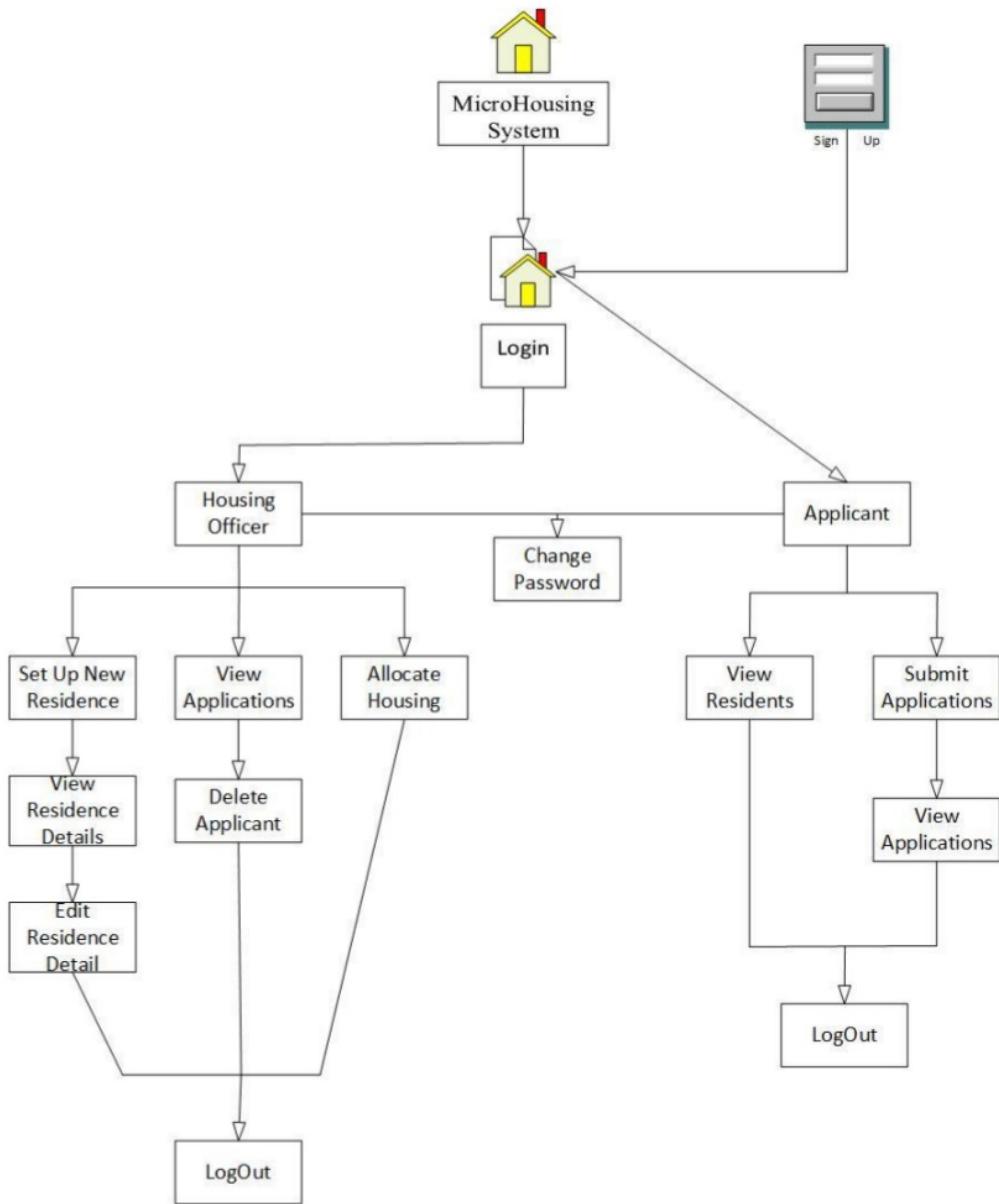
Class Diagram Design



Entity Relationship Diagram (ERD)

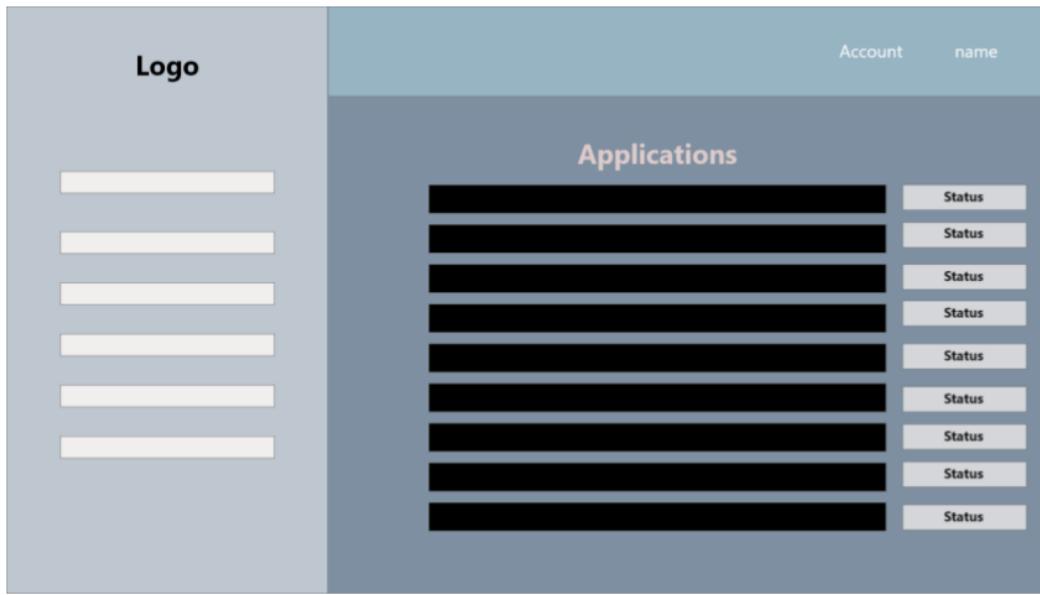


Sitemap

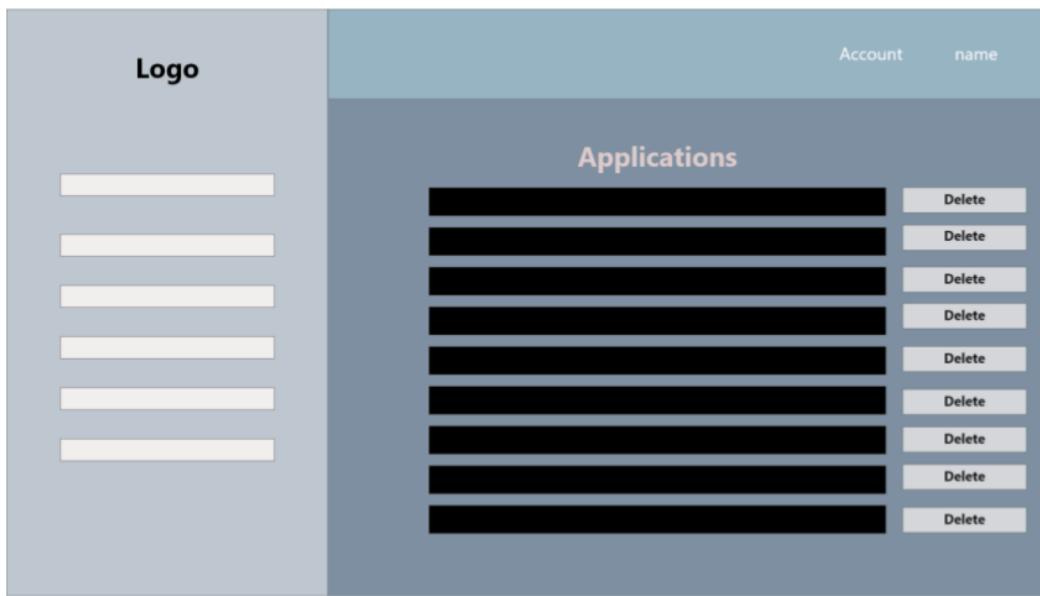


Wireframe

Housing Officer



Allocate Housing



Delete Application

Logo

Account name

Residence ID

Address

Number of Units

Size of Unit

Monthly Rental

Update

This screenshot shows a user interface for editing residence information. On the left is a sidebar with a logo and several empty input fields. The main area has a header with 'Account' and 'name'. Below the header are five input fields: 'Residence ID', 'Address', 'Number of Units', 'Size of Unit', and 'Monthly Rental'. A red oval highlights the 'Update' button at the bottom right.

Edit Residence

Logo

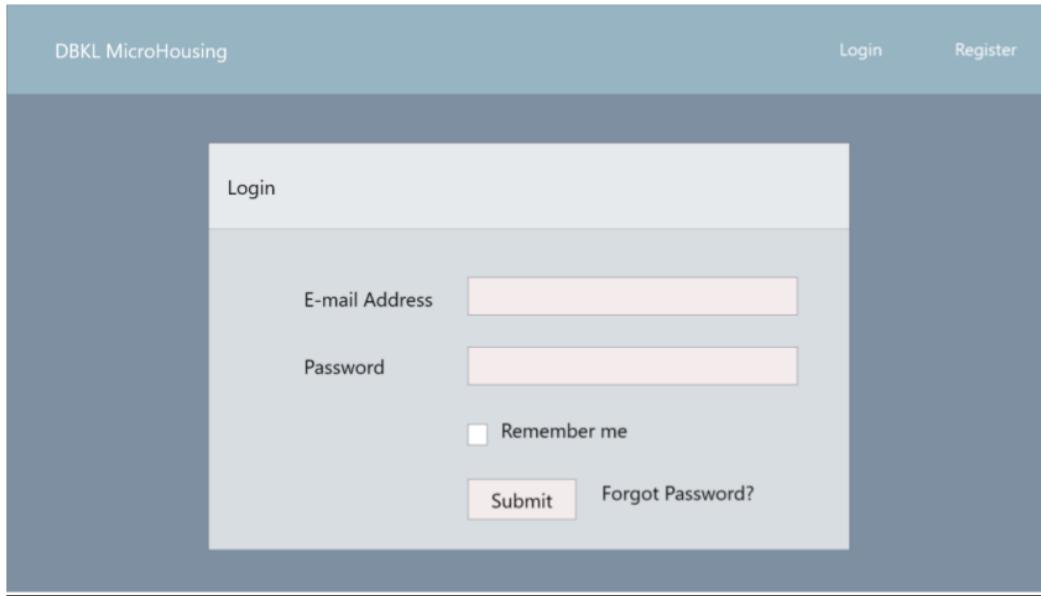
Account name

Logout

User Profile

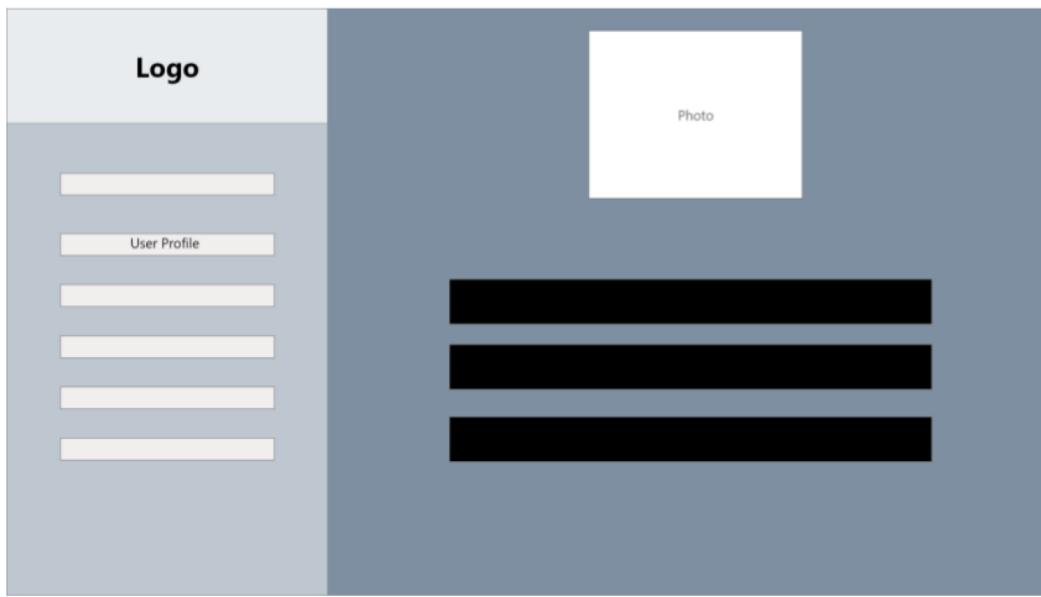
This screenshot shows a user interface for logging out. On the left is a sidebar with a logo and a 'User Profile' link. The main area has a header with 'Account' and 'name'. Below the header are two buttons: 'Logout' and another unlabeled button.

Logout



The image shows the login page of the DBKL MicroHousing website. The header features the "DBKL MicroHousing" logo on the left and "Login" and "Register" links on the right. The main content area is titled "Login". It contains fields for "E-mail Address" and "Password", a "Remember me" checkbox, and buttons for "Submit" and "Forgot Password?".

Login



The image shows the "My Profile" page. On the left, there is a sidebar labeled "Logo" containing several empty input fields and a "User Profile" button. The main content area features a placeholder "Photo" box with a blacked-out profile picture below it. To the right of the photo are three blacked-out horizontal bars.

My Profile

A user interface for setting up a new residence. On the left, there is a vertical sidebar with the word "Logo" at the top and six horizontal white bars below it. The main area has a light blue header bar with the text "Account" and "name". Below this is a dark blue section containing five input fields: "Residence ID", "Address", "Number of Units", "Size of Unit", and "Monthly Rental". A pink oval button labeled "Submit" is located at the bottom right of the dark blue section.

Residence ID

Address

Number of Units

Size of Unit

Monthly Rental

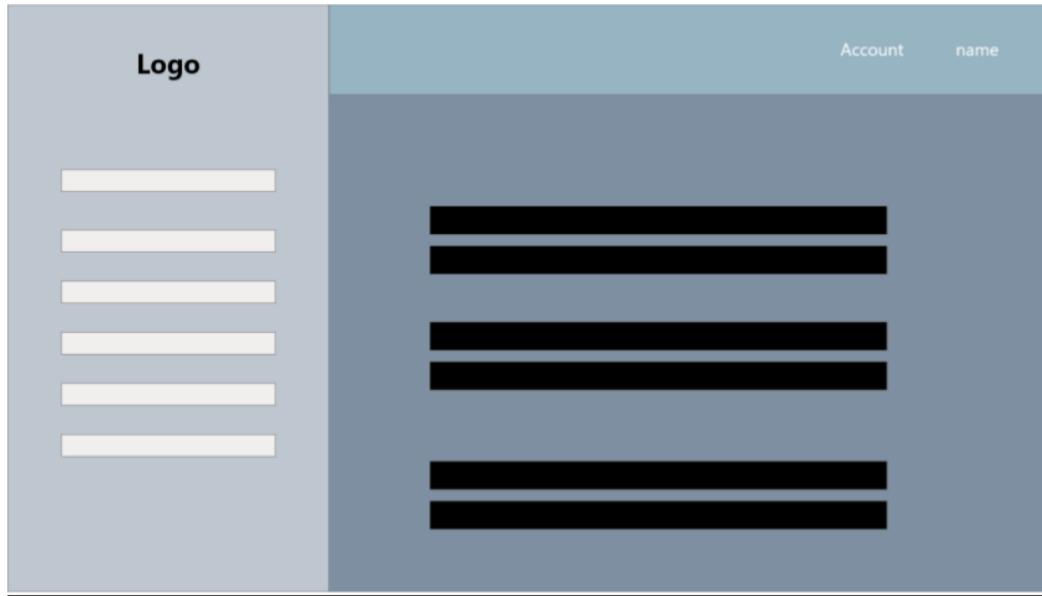
Submit

Set Up New Residence

A user interface for viewing applications. On the left, there is a vertical sidebar with the word "Logo" at the top and six horizontal white bars below it. The main area has a light blue header bar with the text "Account" and "name". Below this is a dark blue section with the heading "Applications" in bold. There are eight thick black horizontal bars representing application entries.

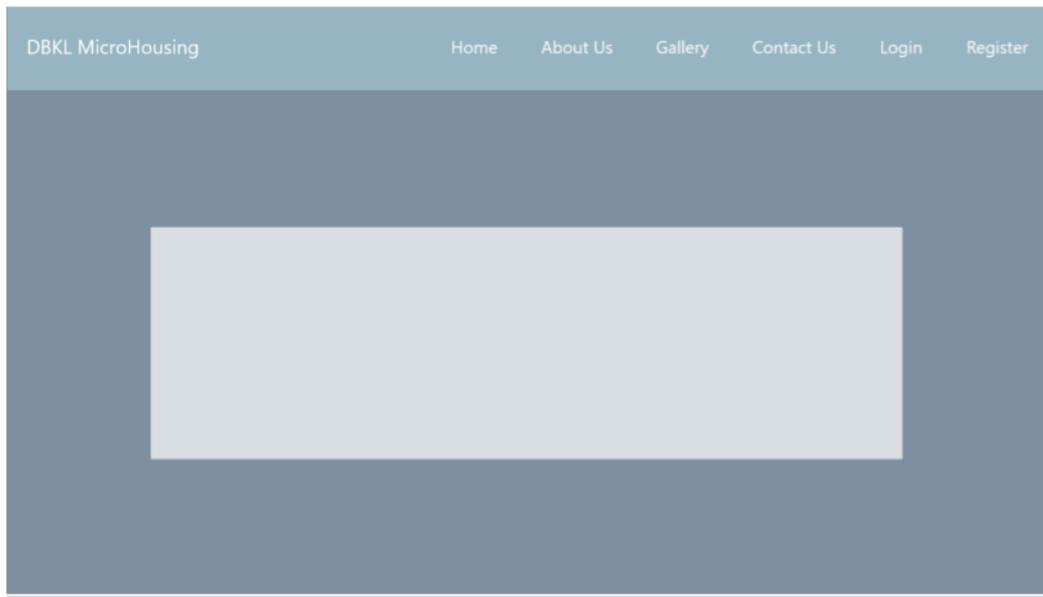
Applications

View Applications



View Residence

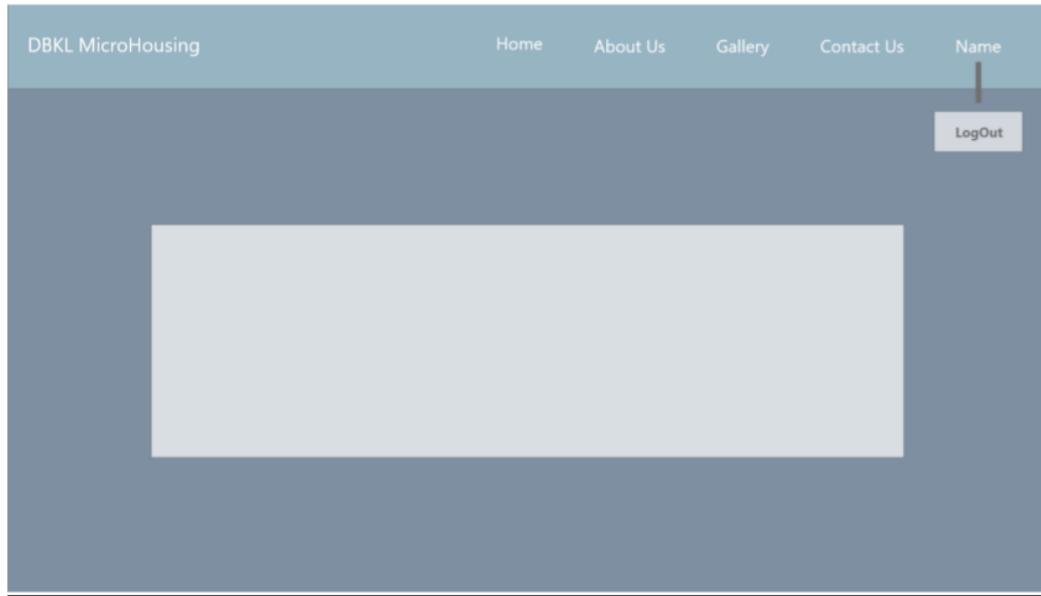
Applicant



Homepage

A screenshot of a login form. At the top left is the "DBKL MicroHousing" logo. To the right are two links: "Login" and "Register". The main form area has a light grey background and a white input box. The word "Login" is centered above the input box. Inside the input box, there are three rows of text fields: "E-mail Address", "Password", and "Remember me". Below the input box are two buttons: "Submit" on the left and "Forgot Password?" on the right.

Login



Logout



Register

DBKL MicroHousing

Home About Us Gallery Contact Us Login Register

Submit Application

applicationID

applicationDate

requiredMonth

requiredYear

status

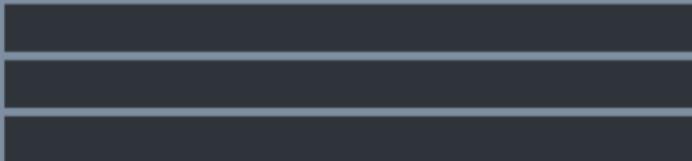
Submit

Submit Applications

DBKL MicroHousing

Home About Us Gallery Contact Us Login Register

View Applications

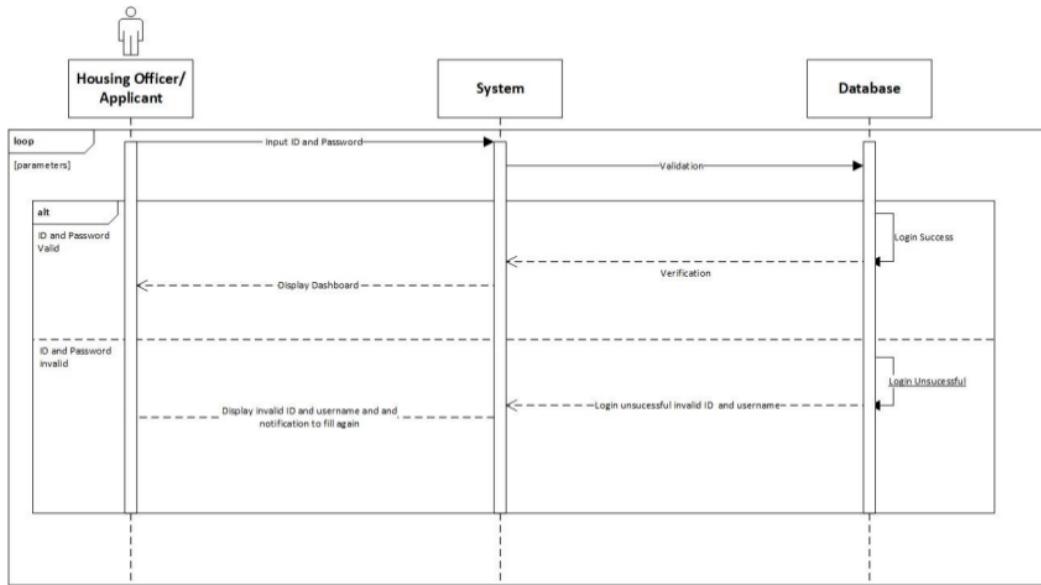


Submit Application
Submit Application
Submit Application

View Residence

System Sequence Diagram + Contracts

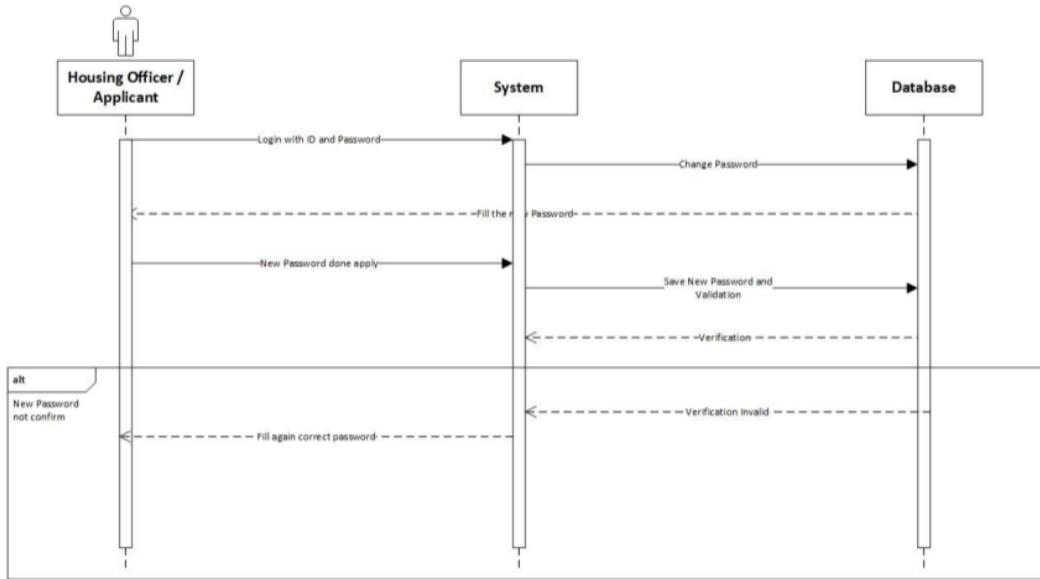
1. Login



Prepared by: Luh Wulandari Maharani

| Cross References | Login |
|------------------|---|
| Operation | Login with ID and username |
| Responsible | To access the main page |
| Pre-conditions | <ul style="list-style-type: none">• Username must be available• Password must be available |
| Post-conditions | <ul style="list-style-type: none">• Username is matched• Password must be match based on user's password input• Display dashboard |

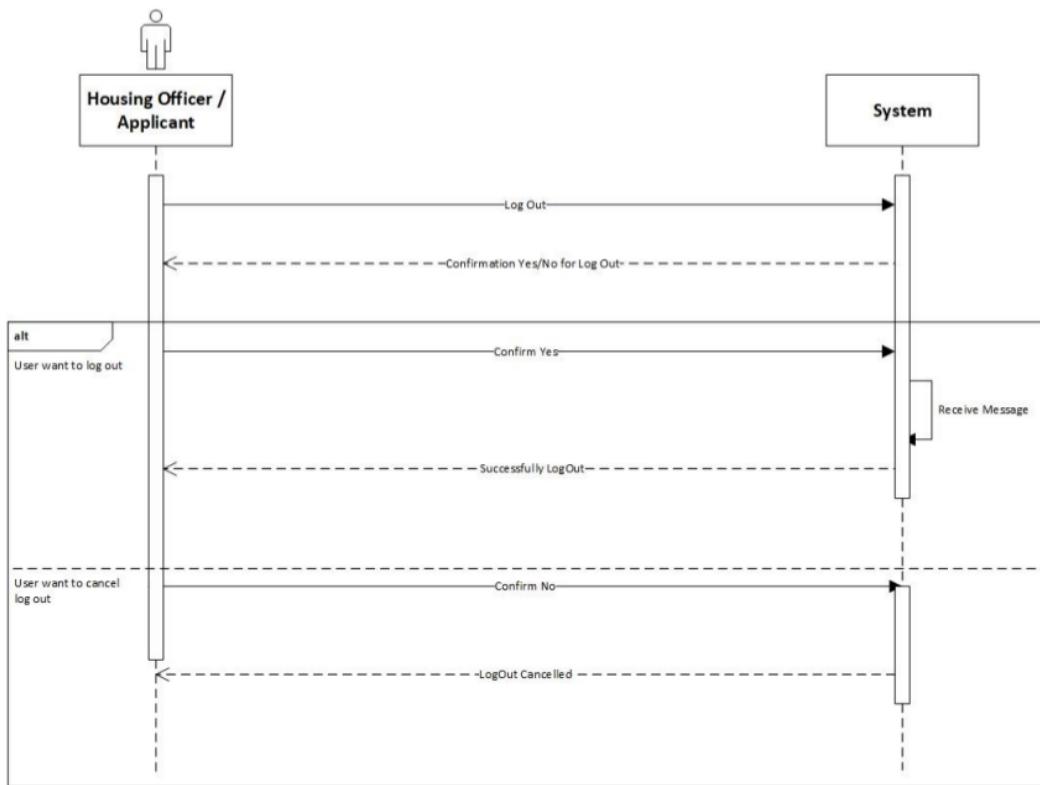
2. Change Password



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 |

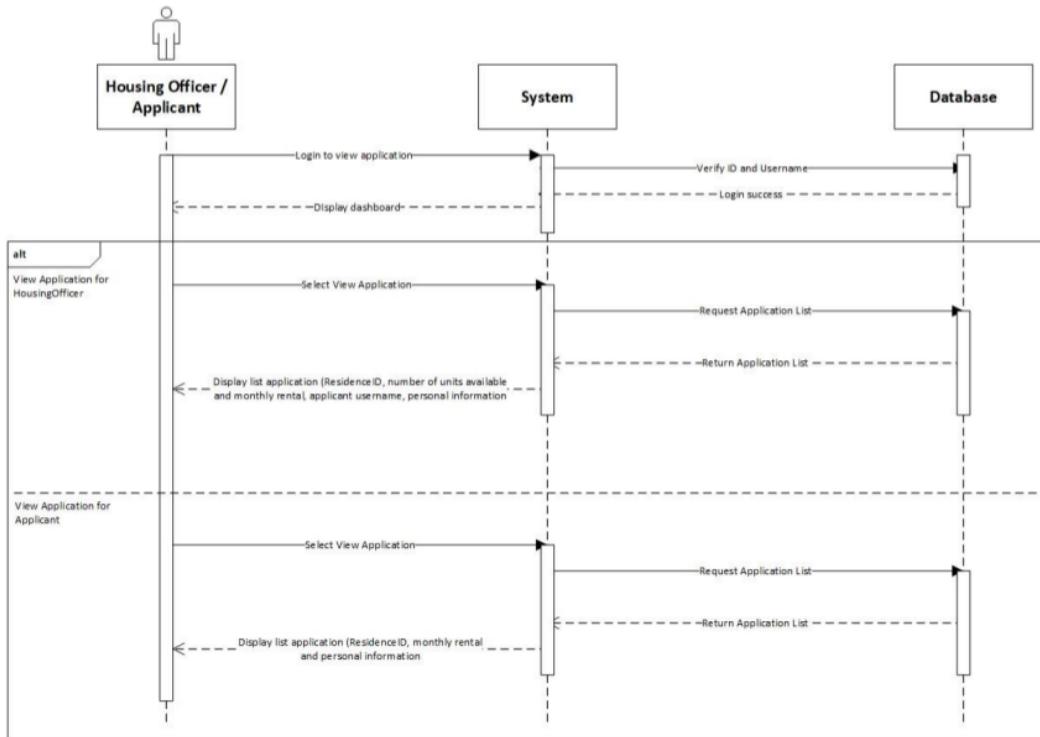
3. Log Out



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

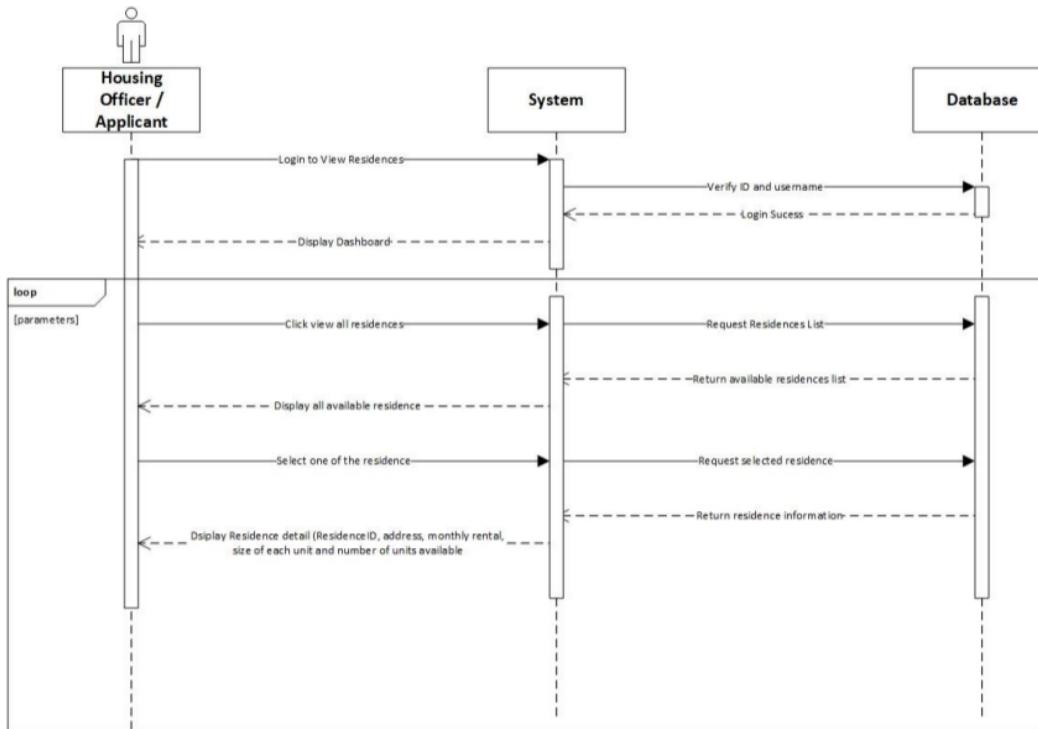


Prepared by: Luh Wulandari Maharani

| Cross References | <u>View Applications</u> |
|------------------|---|
| Operation | Login with username and password |
| Responsible | To get access to view the applications |
| Pre-conditions | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • Username is matched • Password must be match based on user's password input • Display dashboard |
| Cross References | <u>View Applications</u> |
| 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

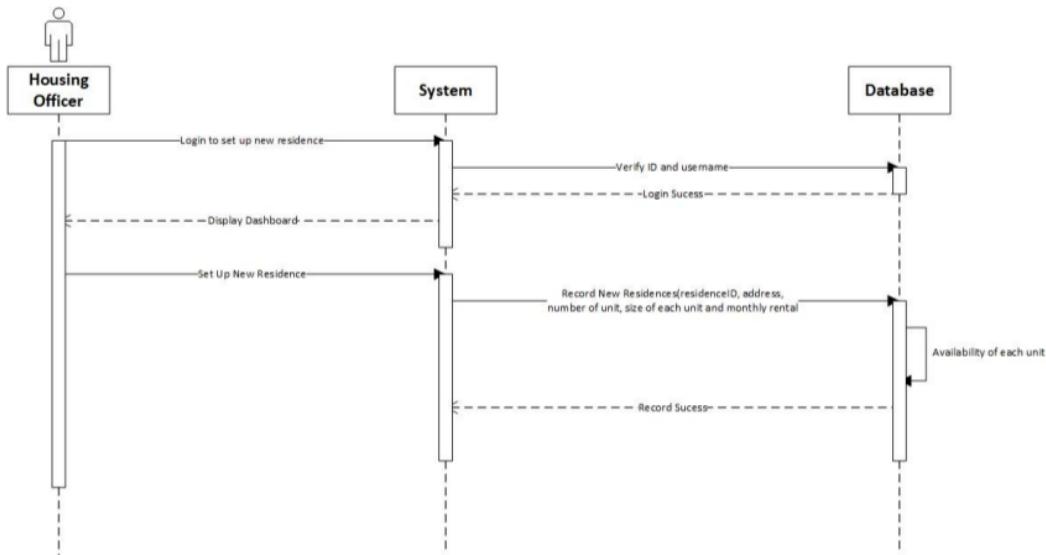


Prepared by: Luh Wulandari Maharani

| Cross References | View Residences |
|------------------|---|
| Operation | Login with ID and username |
| Responsible | To access the main page |
| Pre-conditions | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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

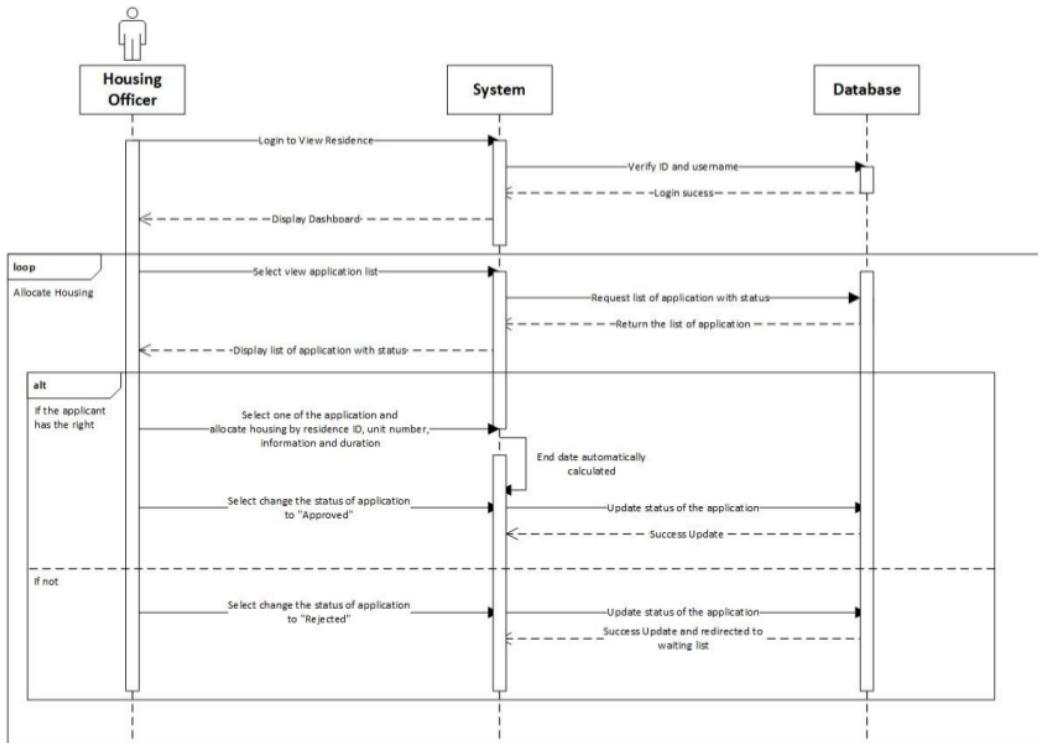


Prepared by: Luh Wulandari Maharani

| Cross References | Set Up New Residences |
|------------------|---|
| Operation | Login with ID and username |
| Responsible | To access the main page |
| Pre-conditions | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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 | 2) 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

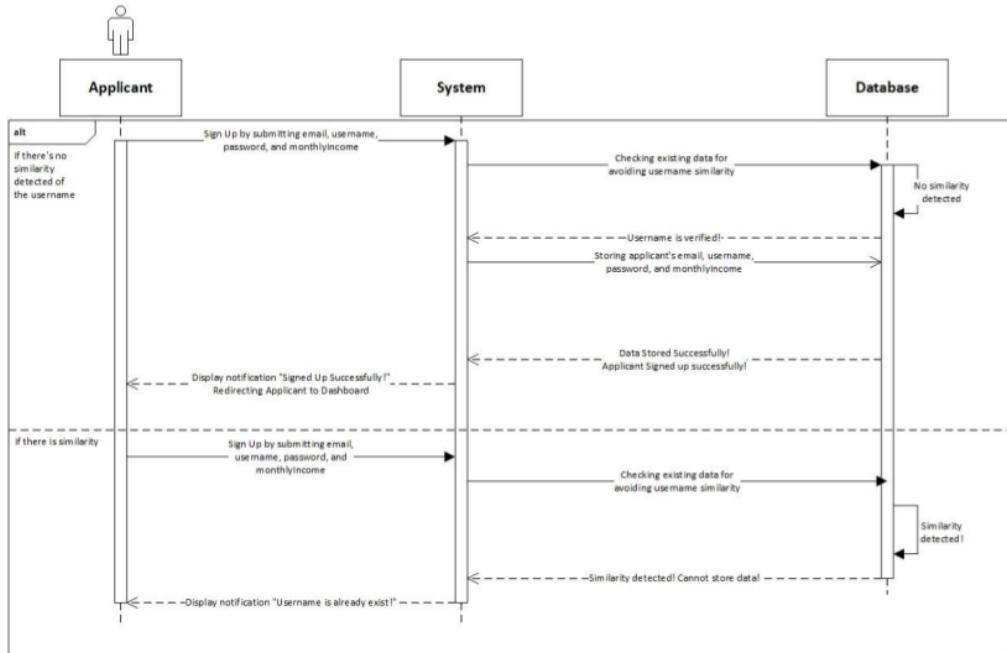


Prepared by: Luh Wulandari Maharani

| Cross References | Allocate Housing |
|------------------|---|
| Operation | Login with ID and username |
| Responsible | To access the main page |
| Pre-conditions | <ul style="list-style-type: none"> Username must be available Password must be available |
| Post-conditions | <ul style="list-style-type: none"> 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 |

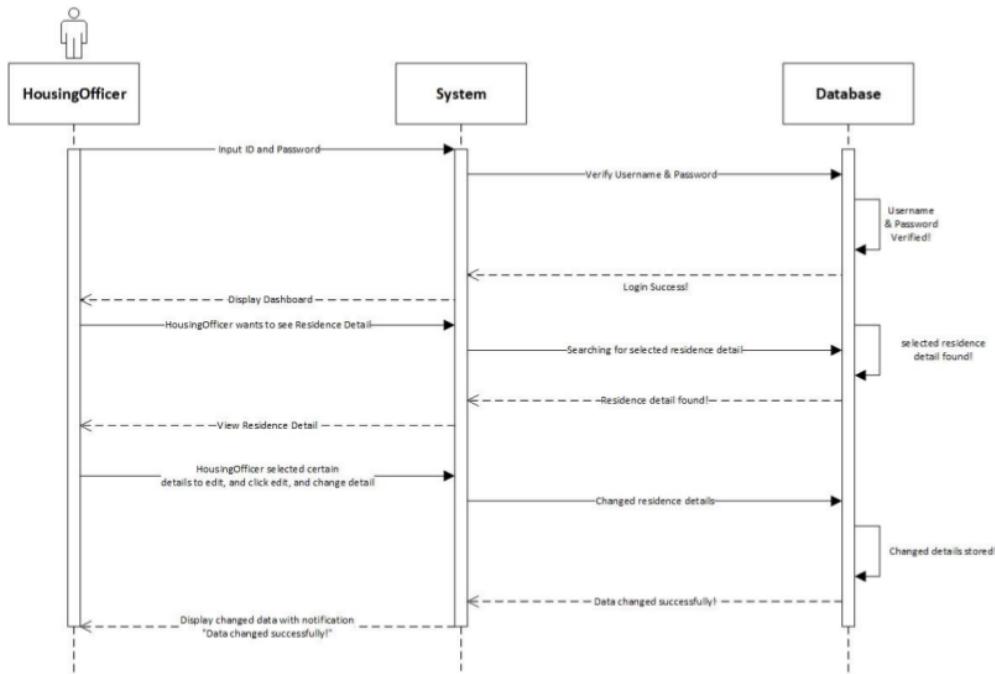
8. Sign Up Applicant



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 | <ul style="list-style-type: none"> Email must be available Username must be available Password must be available MonthlyIncome must be available |
| Post-conditions | <ul style="list-style-type: none"> 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 | <ul style="list-style-type: none"> Email must be available Username must be available Password must be available MonthlyIncome must be available |
| Post-conditions | <ul style="list-style-type: none"> Username is not verified (already existed in system) Applicant must re-enter new username. |

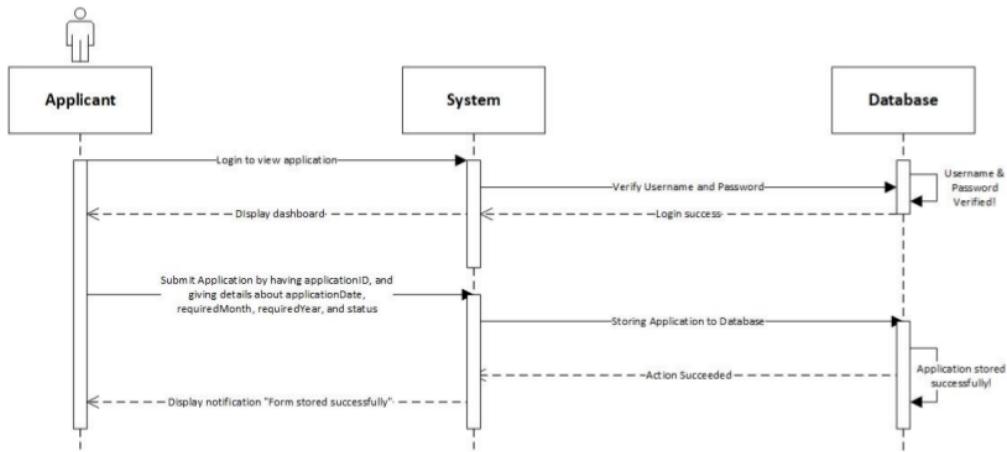
9. Edit Residence



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 | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • Email must be available • Username must be available • Password must be available • MonthlyIncome must be available |
| Post-conditions | Residence Detail changed successfully |

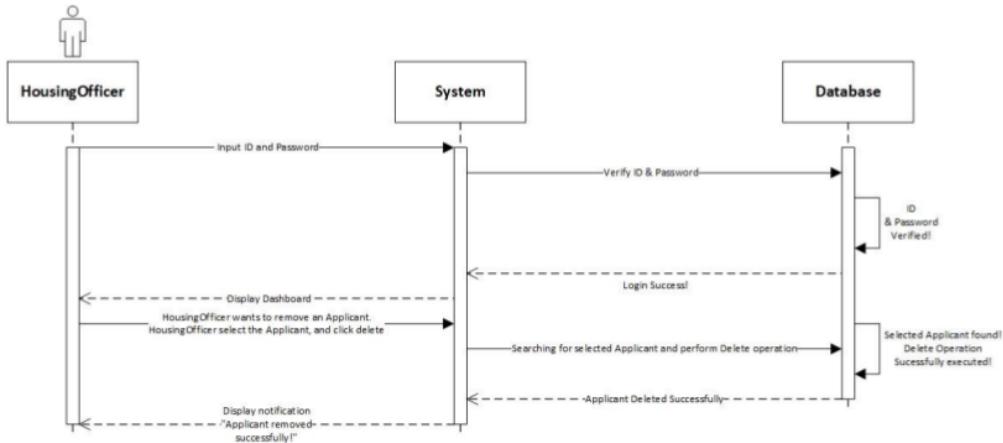
10. Submit Application



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 | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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 | <ul style="list-style-type: none"> • 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 | <p>Applicant has already selected the residence. We will need all of the details below:</p> <ul style="list-style-type: none"> • 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. |

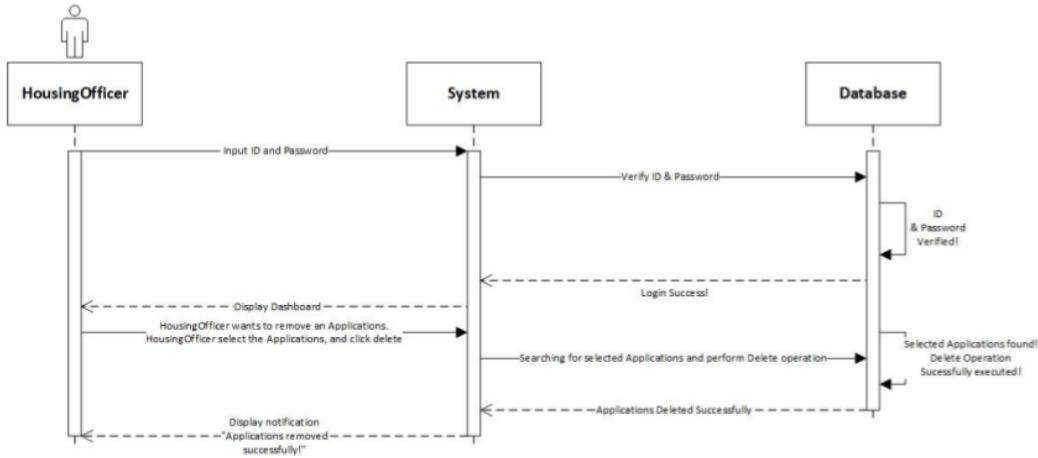
11. Delete Applicant



Prepared by: Rivaldo Bagus Soepardhy

| Cross References | Delete Applicant |
|------------------|---|
| Operation | Login with ID and username |
| Responsible | To access dashboard, and delete applicant |
| Pre-conditions | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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 |

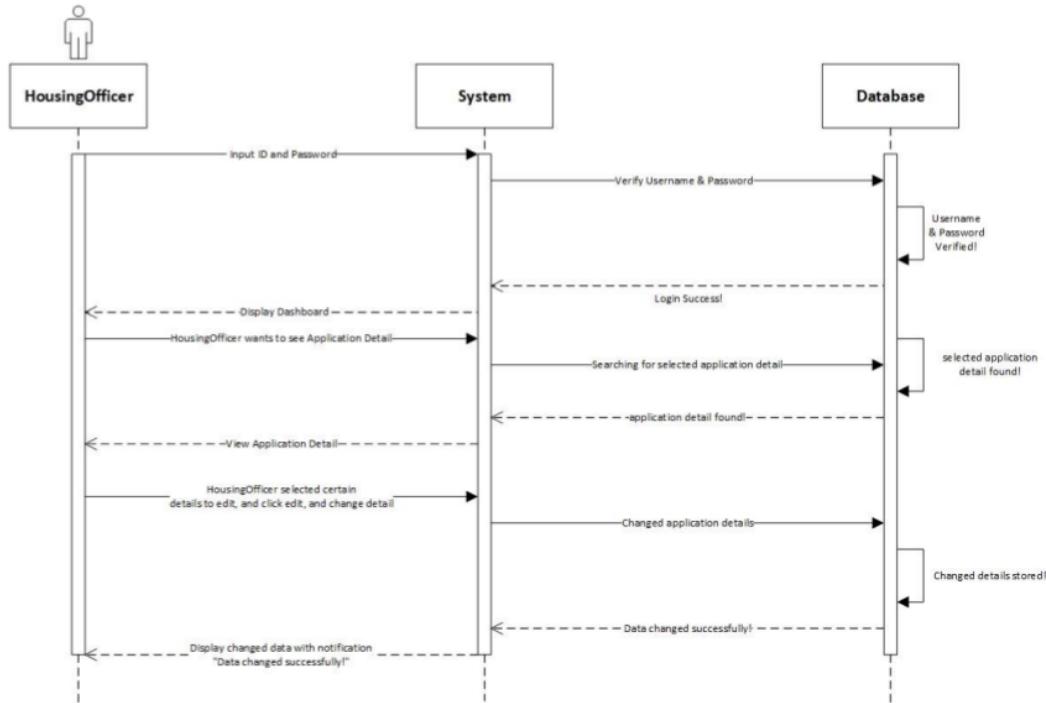
12. Delete Applications



Prepared by: Luh Wulandari Maharani

| Cross References | Delete Applications |
|------------------|---|
| Operation | Login with ID and username |
| Responsible | To access dashboard, and delete applicant |
| Pre-conditions | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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 |

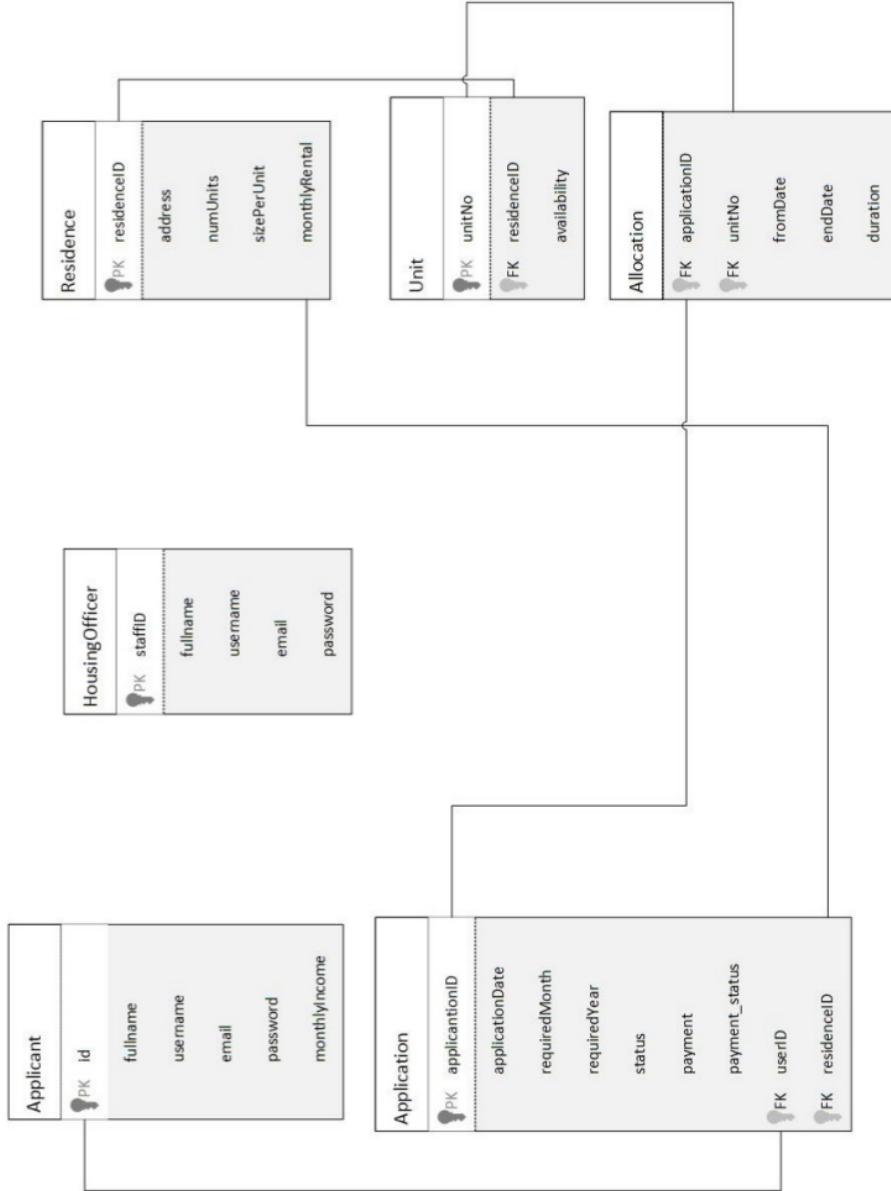
13. Edit Applications



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 | <ul style="list-style-type: none"> • Username must be available • Password must be available |
| Post-conditions | <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 counter 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.

```
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 registerController 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.

```
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']),
    ]);
}
```

```

@extends('layouts.app')

@section('content')


{{ __('Register') }}



@csrf



{{ __('Name') }}

@error('name')
{{ $message }}@enderror



{{ __('Username') }}

@error('username')
{{ $message }}@enderror


```

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

```

<?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.

The screenshot shows a registration form titled "Register". The fields are: Name (empty), Username (empty), E-Mail Address (empty), Monthly Income (empty), Password (empty), and Confirm Password (empty). The "Username" field has a red border and a validation message: "Please fill out this field." Below the form is a "Register" button.

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

The screenshot shows a registration form titled "Register". The fields are: Name (empty), Username ("wulan"), E-Mail Address ("Admin"), Monthly Income (empty), Password (empty), and Confirm Password (empty). The "Username" field has a red border and a validation message: "The username has already been taken." The "E-Mail Address" field has a red border and a validation message: "Please include an '@' in the email address. 'Admin' is missing an '@'." Below the form is a "Register" button.

- 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.**

The screenshot shows a registration form titled "Register". The fields are: Name ("wulan"), Username ("wulan"), E-Mail Address ("wulan"), Monthly Income (empty), Password ("**"), and Confirm Password ("**"). The "E-Mail Address" field has a red border and a validation message: "Please include an '@' in the email address. 'wulan' is missing an '@'." The "Password" field has a red border and a validation message: "Please include an '@' in the email address. 'wulan' is missing an '@'." Below the form is a "Register" button.

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.L1YWIBu.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')


{{ __('Login') }}



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

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

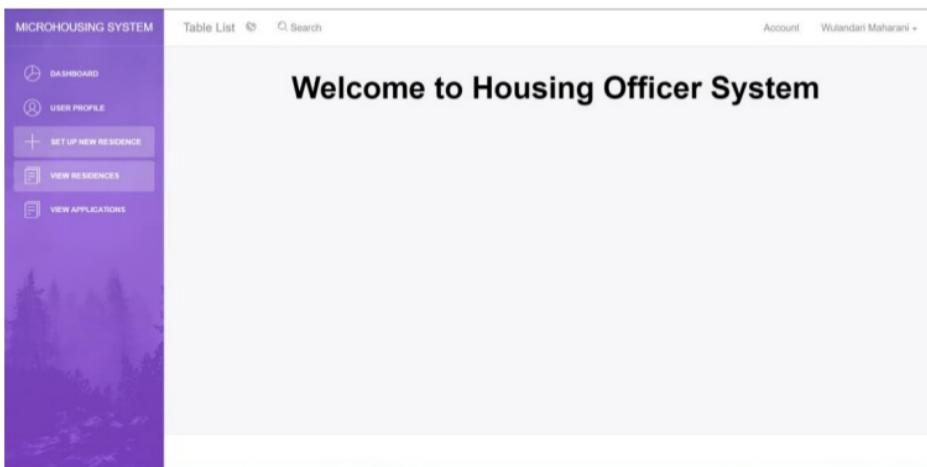

```

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



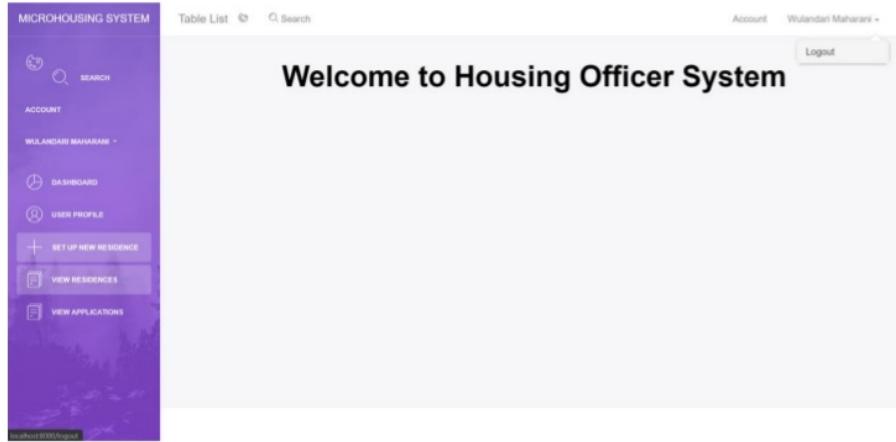
Housing Officer Homepage



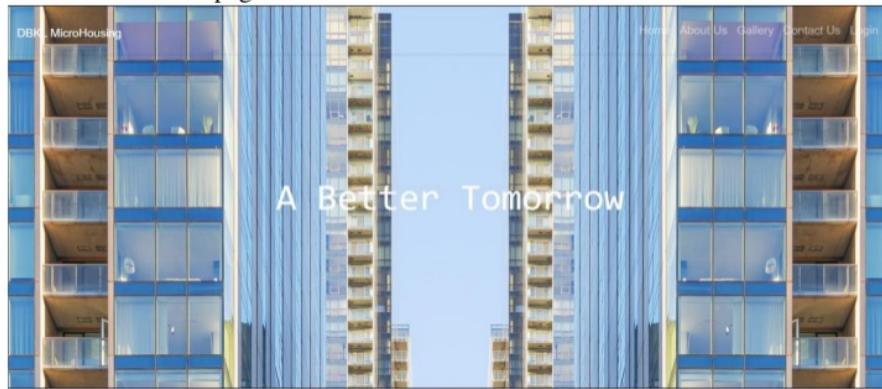
3. LogOut

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.

```
<a 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>
```



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 addressence.blade.php which is form to input the data

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

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

@section('content')



# 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> </form>


```

The screenshot shows the 'Microhousing System' dashboard on the left with three main buttons: 'DASHBOARD', 'SET UP NEW RESIDENCE' (which is highlighted in purple), and 'VIEW RESIDENCES'. The central area is titled 'Set Up New Residence' and contains four input fields with placeholder text: 'Address', 'Num of Units', 'Size of Units', and 'Monthly Rental'. Below these fields is a blue 'Save Data' button.

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')


#### View Residences



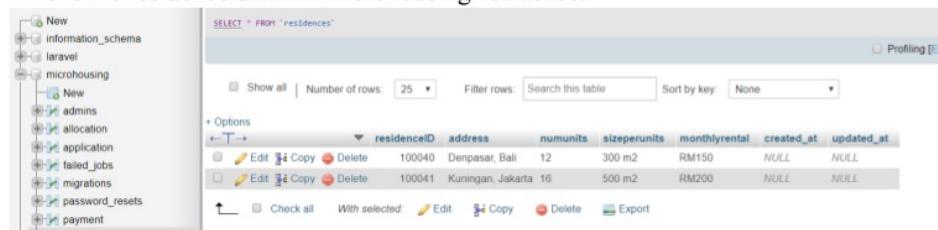
| Residence ID                 | Address                  | Number of Units           | Size per Unit                 | Monthly Rental                 | EDIT                                                                                       |
|------------------------------|--------------------------|---------------------------|-------------------------------|--------------------------------|--------------------------------------------------------------------------------------------|
| {{\$residence->residenceID}} | {{\$residence->address}} | {{\$residence->numunits}} | {{\$residence->sizeperunits}} | {{\$residence->monthlyrental}} | <a class="btn btn-sucess" href="/editresidences({{\$residence-&gt;residenceID}})">EDIT</a> |


```

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

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

This is the residence data in 'microhousing' database.



The screenshot shows a MySQL database browser interface. On the left, there is a tree view of databases: 'information_schema', 'laravel', 'microhousing', and several other tables like 'admins', 'allocation', 'application', 'failed_jobs', 'migrations', 'password_resets', and 'payment'. The 'residences' table is selected. In the main area, the SQL query 'SELECT * FROM "residences"' is displayed above a table. The table has columns: residenceID, address, numunits, sizeperunits, monthlyrental, created_at, and updated_at. There are two rows of data:

| residenceID | address | numunits | sizeperunits | monthlyrental | created_at | updated_at |
|-------------|-------------------|----------|--------------|---------------|------------|------------|
| 100040 | Denpasar, Bali | 12 | 300 m2 | RM150 | NULL | NULL |
| 100041 | Kuningan, Jakarta | 16 | 500 m2 | RM200 | NULL | NULL |

The view residence in the housing officer page

| RESIDENCE ID | ADDRESS | NUMBER OF UNITS | SIZE PER UNIT | MONTHLY RENTAL | EDIT |
|--------------|-------------------|-----------------|---------------|----------------|-----------------------|
| 100040 | Denpasar, Bali | 12 | 300 m2 | RM150 | <button>EDIT</button> |
| 100041 | Kuningan, Jakarta | 16 | 500 m2 | RM200 | <button>EDIT</button> |

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

```
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.

```
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>
            </form>
        </div>
    </div>
```

This this display form edit residence

The screenshot shows the MicroHousing System dashboard. On the left, there is a sidebar with three buttons: 'DASHBOARD' (selected), 'SET UP NEW RESIDENCE', and 'VIEW RESIDENCES'. The main content area has a title 'Edit Residence'. Below the title are four input fields: 'Address' (Denpasar, Bali), 'Num of Units' (12), 'Size of Units' (300 m2), and 'Monthly Rental' (RM150). At the bottom right of the form is a blue 'Update Data' button.

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="row">
        <div class="col-md-12">
            <div class="card striped-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>Application 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

The screenshot shows a web-based application interface for 'MICROHOUSING SYSTEM'. On the left, there is a sidebar with navigation links: 'DASHBOARD', 'SET UP NEW RESIDENCE', 'VIEW RESIDENCES', and 'VIEW APPLICATIONS'. The main content area is titled 'View Applications' and displays a table with one row of data. The table columns are: APPLICANT ID, APPLICATION ID, RESIDENCE ID, APPLICATION DATE, REQUIRED MONTH, REQUIRED YEAR, STATUS, DELETE, and ALLOCATE. The data in the table is as follows:

| APPLICANT ID | APPLICATION ID | RESIDENCE ID | APPLICATION DATE | REQUIRED MONTH | REQUIRED YEAR | STATUS | DELETE | ALLOCATE |
|--------------|----------------|--------------|---------------------|----------------|---------------|------------|-------------------------|---------------------------|
| 1750100 | 2020100 | 100040 | 2020-04-04 19:56:48 | June | 2020 | Processing | <button>DELETE</button> | <button>ALLOCATE</button> |

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('logout-form').submit();">Logout</a>
                            <form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">@csrf</form>
                        </div>
                    </li>
                </ul>
            </div>
        </nav>
    </body>

```

This is display submit application page

The screenshot shows a web page titled "Submit Application". The page has a dark header with the "DBKL MicroHousing" logo and a navigation bar. The main content area features a large image of a modern house with a blue roof and white trim. Overlaid on the image is the word "Submit Application" in a large, light-colored font. Below the image, there are four input fields in a dark box:

| | |
|----------------|-------------------|
| Applicant ID | 2020100 |
| Residence ID | 100040 |
| Required Month | E.g. May, June... |
| Required Year | E.g. 2020 |

At the bottom of the form is a blue "Submit Application" button.

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('logout-form').submit();">Logout</a>
                        </div>
                    </li>
                </ul>
            </div>
        </nav>

```

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

| Residence ID | Address | Number of Units | Size Per Unit | Monthly Rental | Action |
|--------------|-------------------|-----------------|--------------------|----------------|------------------------------------|
| 100040 | Denpasar, Bali | 12 | 300 m ² | RM150 | Submit Application |
| 100041 | Kuningan, Jakarta | 16 | 500 m ² | RM200 | Submit Application |

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
                </table>
            </div>
        </div>
    </div>
</div>
```

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

| Application ID | Residence ID | Application Date | Required Month | Required Year | Status | Payment | Payment Status |
|----------------|--------------|---------------------|----------------|---------------|------------|-------------|----------------|
| 1750100 | 100040 | 2020-04-04 19:56:48 | June | 2020 | Processing | Unavailable | Unavailable |

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()
```

| Log | Reference |
|---|-----------|
| 3. click on linkText=Login | OK |
| 4. click on id=username | OK |
| 5. type on id=username with value wulan | OK |
| 6. click on id=password | OK |
| 7. type on id=password with value 12345678 | OK |
| 8. click on css=.btn-primary | OK |
| 'LOGIN APPLICANT' completed successfully | |

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()
```

| Log | Reference |
|---|-----------|
| 11. click on id=password OK | |
| 12. type on id=password with value 12345678 OK | |
| 13. type on id=password-confirm with value 12345678 OK | |
| 14. click on css=.btn OK | |
| 15. click on id=navbarDropdown OK | |
| 16. click on linkText=Logout OK | |
| 'REGISTER APPLICANT' completed successfully | |

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 |
|---|---------------------|---------------|--------------|------------|-------------|---------|
| Click the drop-down arrow to toggle column's visibility. | | | | | | |
| 1750101 | 2020-04-04 15:15:15 | June | 2020 | Processing | 100040 | 2020101 |
| 1750102 | 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 |



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()
```

The screenshot shows a software interface with a dark header bar. Below it, there are two tabs: 'Log' (underlined in blue) and 'Reference'. The main area contains a list of numbered steps, each followed by the word 'OK' in green, indicating successful completion. At the bottom of the list, a green message reads "'Edit Residence' completed successfully".

| | |
|---|----|
| 19. type on name=sizeperunits with value 500 m2 | OK |
| 20. click on name=monthlyrental | OK |
| 21. type on name=monthlyrental with value RM300 | OK |
| 22. click on name=submit | OK |
| 23. click on id=navbarDropdown | OK |
| 24. click on linkText=Logout | OK |

'Edit Residence' completed successfully

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

| Name | <input type="text" value="I"/> |
|---|--------------------------------|
| Username | <input type="text"/> |
| E-Mail Address | <input type="text"/> |
| Monthly Income | <input type="text"/> |
| Password | <input type="password"/> |
| Confirm Password | <input type="password"/> |
| <input type="button" value="Register"/> | |

a. Applicant input blank information

The screenshot shows the 'Register' page of the DBKL Microhousing website. The page has a header with 'DBKL Microhousing' and navigation links 'Login' and 'Register'. The main content area is titled 'Register' and contains six input fields: 'Name', 'Username', 'E-Mail Address', 'Monthly Income', 'Password', and 'Confirm Password'. Below these fields is a blue 'Register' button.

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

The screenshot shows the 'Register' page with validation errors. The 'Username' field contains 'wulan' and has a red border with the error message 'The username has already been taken.' The 'E-Mail Address' field contains 'Admin' and has a red border with the error message 'Please include an '@' in the email address. Admin' is missing an '@'.' The other fields ('Name', 'Password', 'Confirm Password') are empty and have no validation errors.

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

The screenshot shows the 'Register' page with multiple validation errors. The 'Username' field contains 'wulan' and has a red border with the error message 'The username has already been taken.' The 'E-Mail Address' field contains 'Admin' and has a red border with the error message 'Please include an '@' in the email address. Admin' is missing an '@'.' The 'Password' field has a red border with the error message 'The password is too short. It must be at least 8 characters long.' The 'Confirm Password' field has a red border with the error message 'The password does not match the confirmation.' The 'Name' field is empty and has no validation errors.

DBKL Microhousing

Login Register

Register

| | |
|---|--|
| Name | <input type="text" value="iwulan"/> |
| Username | <input type="text" value="wulandari"/> |
| E-Mail Address | <input type="text" value="wulandari@gmail.com"/> |
| Monthly Income | <input type="text" value="10000"/> |
| Password | <input type="password"/> |
| The password must be at least 8 characters. | |
| Confirm Password | <input type="password"/> |
| <input type="button" value="Register"/> | |

2. Login

Login page same location in the dashboard

DBKL Microhousing

Login Register

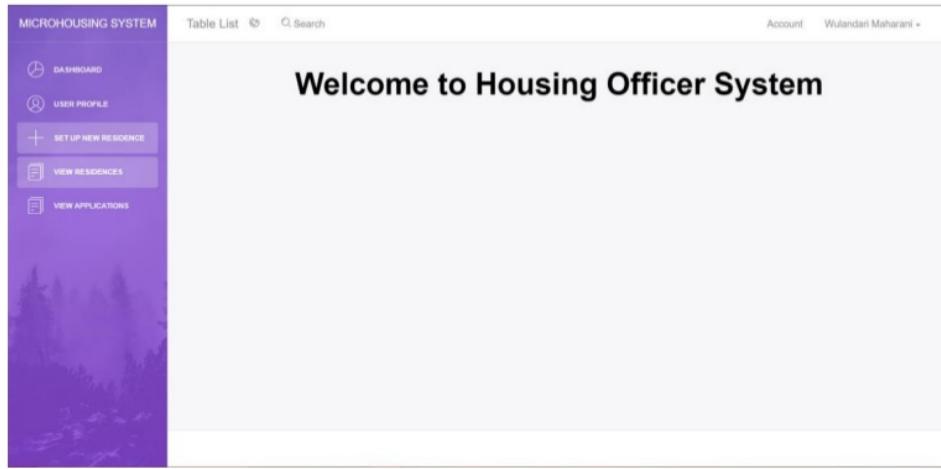
Login

| | |
|--------------------------------------|--------------------------|
| Username/Email | <input type="text"/> |
| Password | <input type="password"/> |
| <input type="checkbox"/> Remember Me | |
| <input type="button" value="Login"/> | |

Applicant Homepage

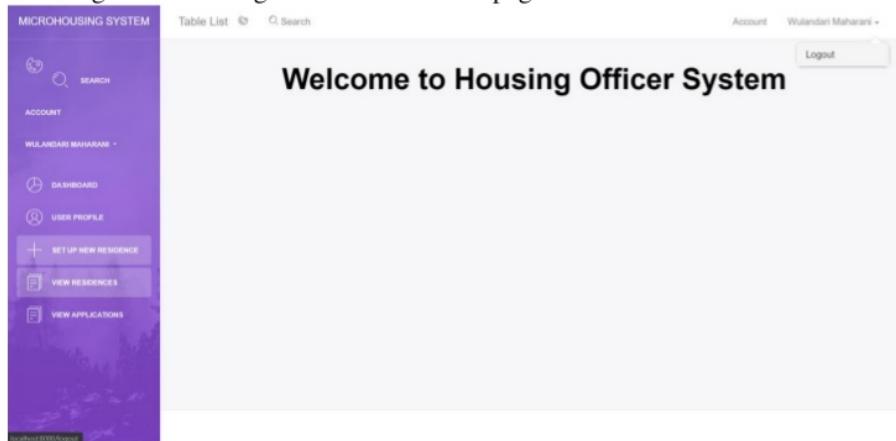


Housing Officer Homepage

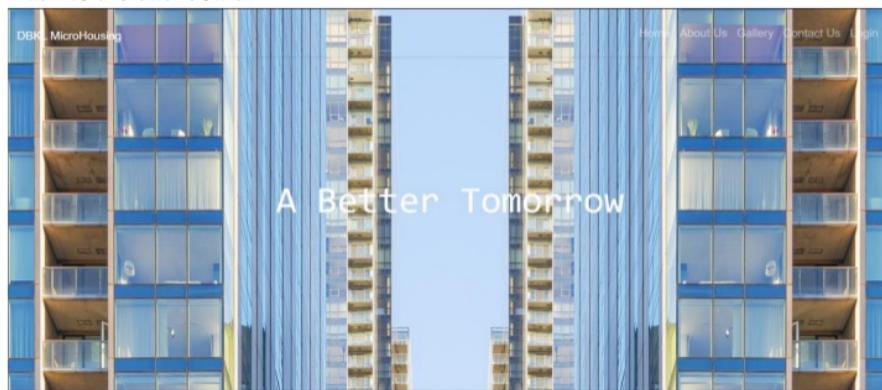


3. LogOut

After login there are logout button above the page

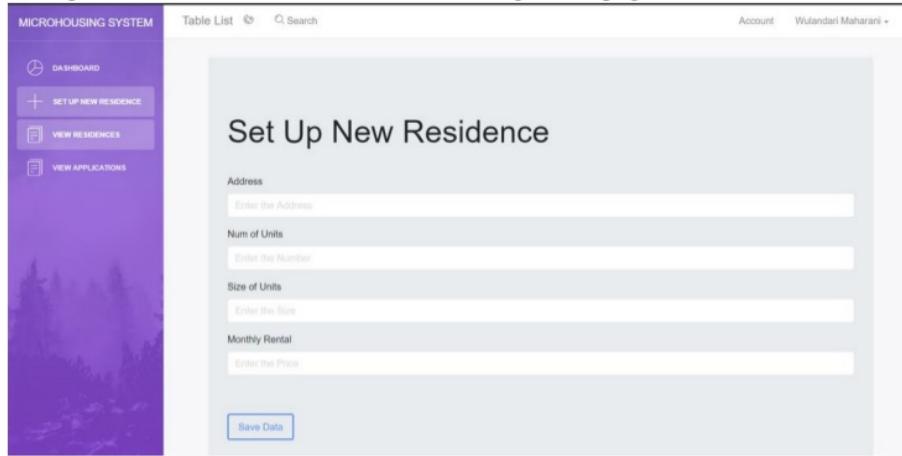


Back to the dashboard



4. Set Up New Residence

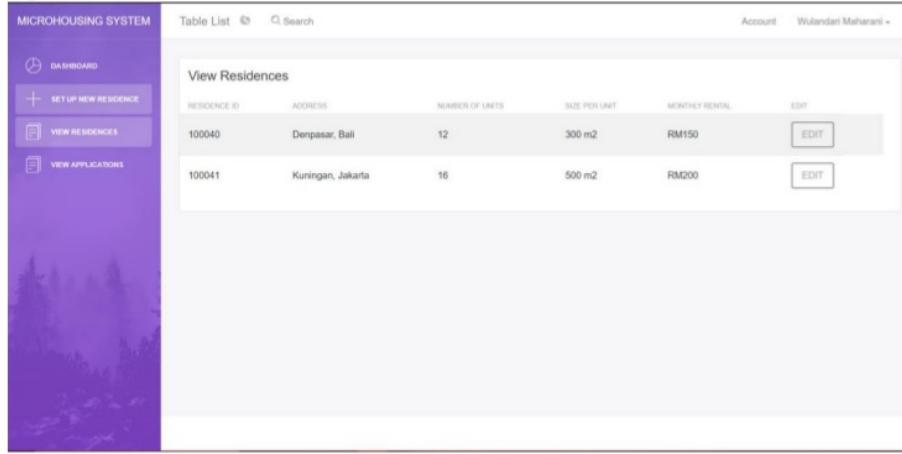
Set Up New Residence menu located in housingofficer page



The screenshot shows the 'Set Up New Residence' form. It has four input fields: 'Address' (placeholder: Enter the Address), 'Num of Units' (placeholder: Enter the Number), 'Size of Units' (placeholder: Enter the Size), and 'Monthly Rental' (placeholder: Enter the Price). Below the fields is a blue 'Save Data' button.

5. View Residence Detail

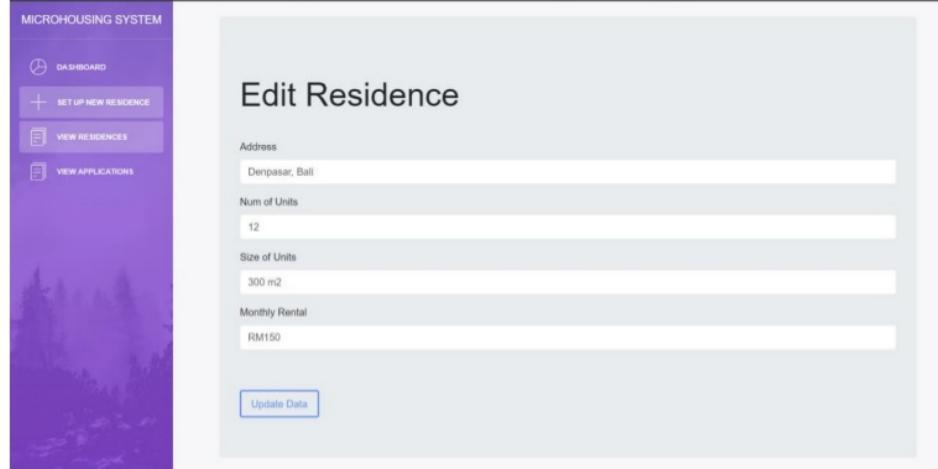
To view residence, which does the housing officer in the housing officer page already input



| RESIDENCE ID | ADDRESS | NUMBER OF UNITS | SIZE PER UNIT | MONTHLY RENTAL | EDIT |
|--------------|-------------------|-----------------|---------------|----------------|-----------------------|
| 100040 | Denpasar, Bali | 12 | 300 m2 | RM150 | <button>EDIT</button> |
| 100041 | Kuningan, Jakarta | 16 | 500 m2 | RM200 | <button>EDIT</button> |

6. Edit Residence Detail

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



MICROHOUSING SYSTEM

DASHBOARD

+ SET UP NEW RESIDENCE

VIEW RESIDENCES

VIEW APPLICATIONS

Edit Residence

Address

Denpasar, Bali

Num of Units

12

Size of Units

300 m²

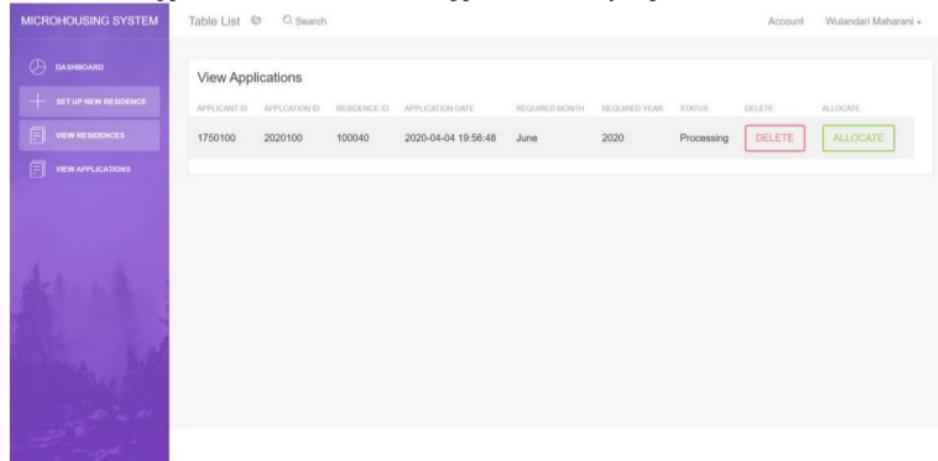
Monthly Rental

RM150

Update Data

7. View Applications

To view the application which does the applicant already input.



MICROHOUSING SYSTEM

Table List ⏪ ⏴ Search Account Wulandari Maharani ▾

View Applications

| APPLICANT ID | APPLICATION ID | RESIDENCE ID | APPLICATION DATE | REQUIRED MONTH | REQUIRED YEAR | STATUS | DELETE | ALLOCATE |
|--------------|----------------|--------------|---------------------|----------------|---------------|------------|---------------------|-----------------------|
| 1750100 | 2020100 | 100040 | 2020-04-04 19:56:48 | June | 2020 | Processing | DELETE | ALLOCATE |

8. Submit Application

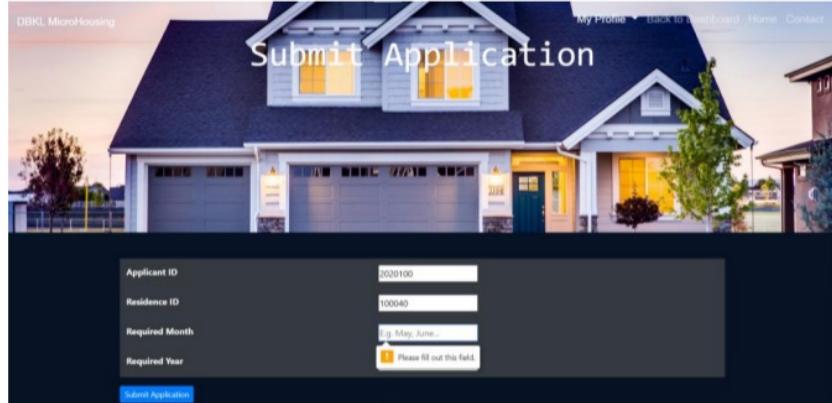
To submit the application



| | |
|----------------|-------------------|
| Applicant ID | 2020100 |
| Residence ID | 100040 |
| Required Month | E.g. May, June... |
| Required Year | E.g. 2020 |

Submit Application

- Applicant did not input the required month or required year



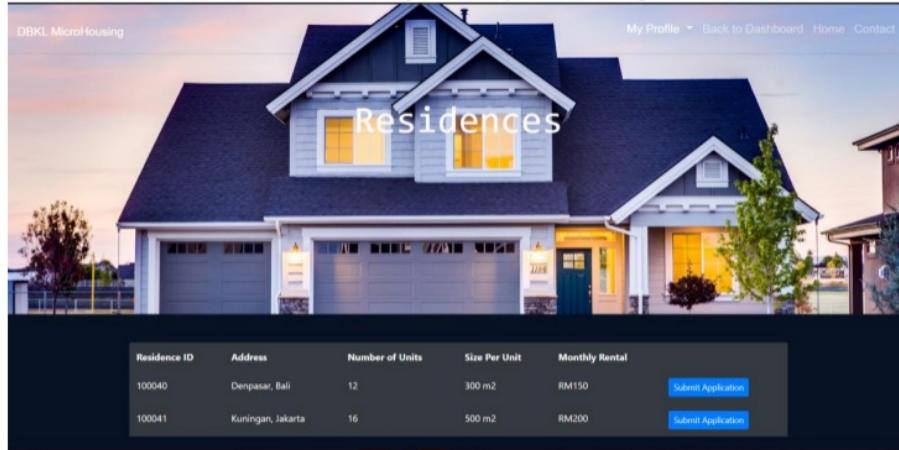
| | |
|----------------|-------------------|
| Applicant ID | 2020100 |
| Residence ID | 100040 |
| Required Month | E.g. May, June... |
| Required Year | E.g. 2020 |

Please fill out this field.

Submit Application

9. View Residence (Applicant)

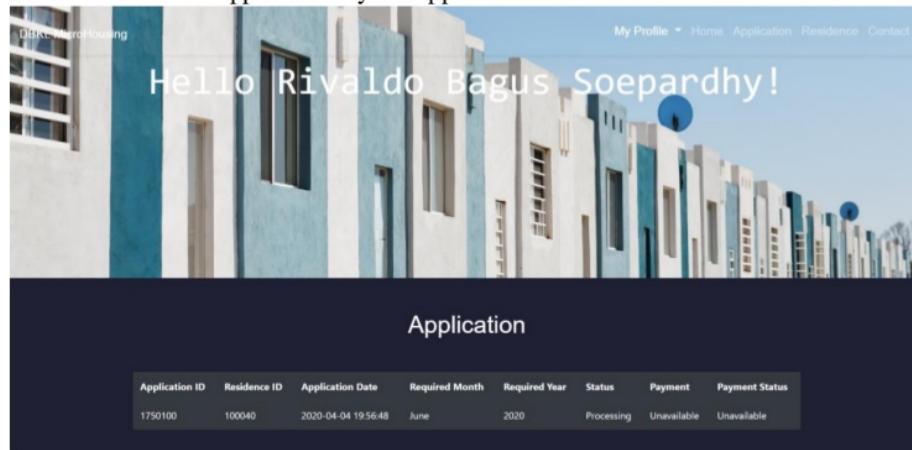
To view all residence which is already make by the housing officer



| Residence ID | Address | Number of Units | Size Per Unit | Monthly Rental | |
|--------------|-------------------|-----------------|--------------------|----------------|---------------------------|
| 100040 | Denpasar, Bali | 12 | 300 m ² | RM150 | Submit Application |
| 100041 | Kuningan, Jakarta | 16 | 500 m ² | RM200 | Submit Application |

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.

A screenshot of the DBKL Microhousing login page. The page has a light gray header with the text 'DBKL Microhousing' on the left and 'Login Register' on the right. Below the header is a 'Login' form. The form includes fields for 'Username/Email' and 'Password', both represented by input boxes. There is also a 'Remember Me' checkbox and a blue 'Login' button at the bottom of the form.

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

The image shows two screenshots of a web-based housing application system.

Screenshot 1: Dashboard

This screenshot shows the main dashboard of the "MicroHousing System". On the left is a sidebar with a purple header containing the system name and a search bar. Below the header, the sidebar has sections for "ACCOUNT" and "WULANDARI MAHARANI". Under "WULANDARI MAHARANI", there are four items: "DASHBOARD", "USER PROFILE", "SET UP NEW RESIDENCE", "VIEW RESIDENCES", and "VIEW APPLICATIONS". The main content area features a large heading "Welcome to Housing Officer System" and a "Logout" button in the top right corner.

Screenshot 2: Application Details

This screenshot shows a specific application detail page. At the top, it displays a welcome message: "Hello Rivaldo Bagus Soepardhy!". Below this is a section titled "Application". A table provides detailed information about the application:

| Application ID | Residence ID | Application Date | Required Month | Required Year | Status | Payment | Payment Status |
|----------------|--------------|---------------------|----------------|---------------|------------|-------------|----------------|
| 1750100 | 100040 | 2020-04-04 19:56:48 | June | 2020 | Processing | Unavailable | Unavailable |

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 |
|---|---------------------|---------------|--------------|------------|-------------|---------|
| Click the drop-down arrow to toggle column's visibility. | | | | | | |
| 1750101 | 2020-04-04 15:15:15 | June | 2020 | Processing | 100040 | 2020101 |
| 1750102 | 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 |



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

The screenshot shows the 'Set Up New Residence' form. On the left is a sidebar with a purple background featuring a forest scene. The main area has a light gray background. At the top right, it says 'Account: Wulandari Maharani'. The title 'Set Up New Residence' is centered at the top. Below it are four input fields: 'Address' (placeholder 'Enter the Address'), 'Num of Units' (placeholder 'Enter the Number'), 'Size of Units' (placeholder 'Enter the Size'), and 'Monthly Rental' (placeholder 'Enter the Price'). A blue 'Save Data' button is at the bottom.

The screenshot shows the 'View Residences' table. The sidebar on the left is identical to the previous screenshot. The main area has a light gray background. At the top right, it says 'Account: Wulandari Maharani'. The title 'View Residences' is centered at the top. Below it is a table with the following data:

| RESIDENCE ID | ADDRESS | NUMBER OF UNITS | SIZE PER UNIT | MONTHLY RENTAL | EDIT |
|--------------|-------------------|-----------------|---------------|----------------|-----------------------|
| 100040 | Denpasar, Bali | 12 | 300 m2 | RM150 | <button>EDIT</button> |
| 100041 | Kuningan, Jakarta | 16 | 500 m2 | RM200 | <button>EDIT</button> |

Edit Residence

Address
Denpasar, Bali

Num of Units
12

Size of Units
300 m2

Monthly Rental
RM150

Update Data

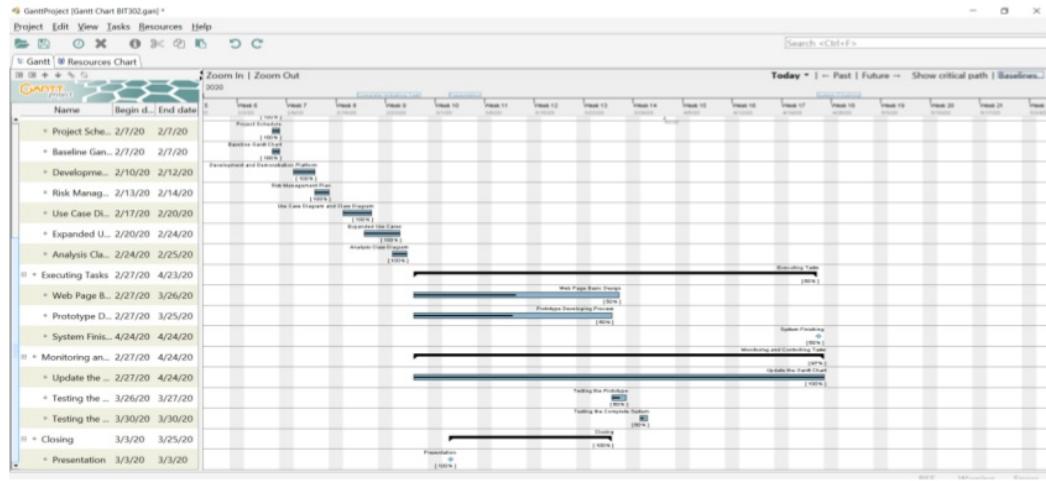
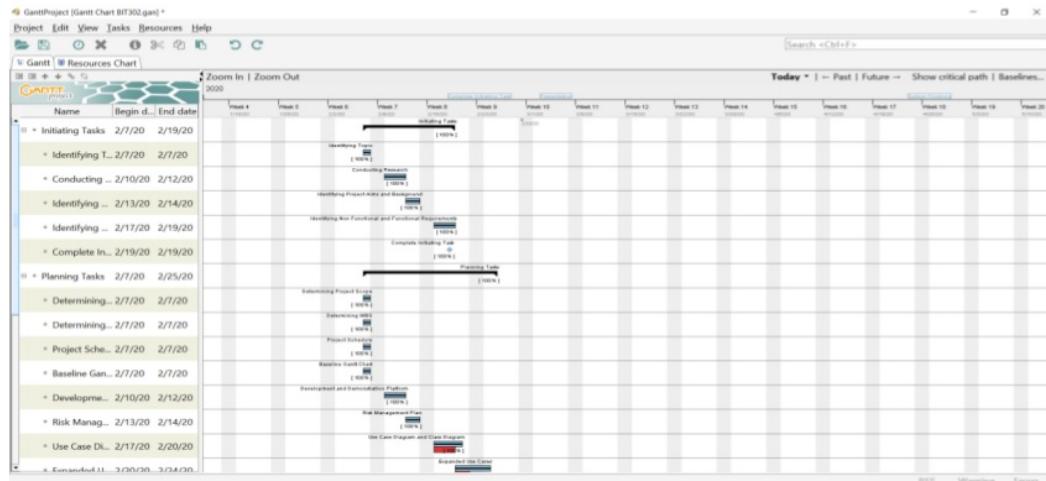
View Applications

| APPLICANT ID | APPLICATION ID | RESIDENCE ID | APPLICATION DATE | REQUIRED MONTH | REQUIRED YEAR | STATUS | DELETE | ALLOCATE |
|--------------|----------------|--------------|---------------------|----------------|---------------|------------|---------------|-----------------|
| 1750100 | 2020100 | 100040 | 2020-04-04 19:56:48 | June | 2020 | Processing | DELETE | ALLOCATE |

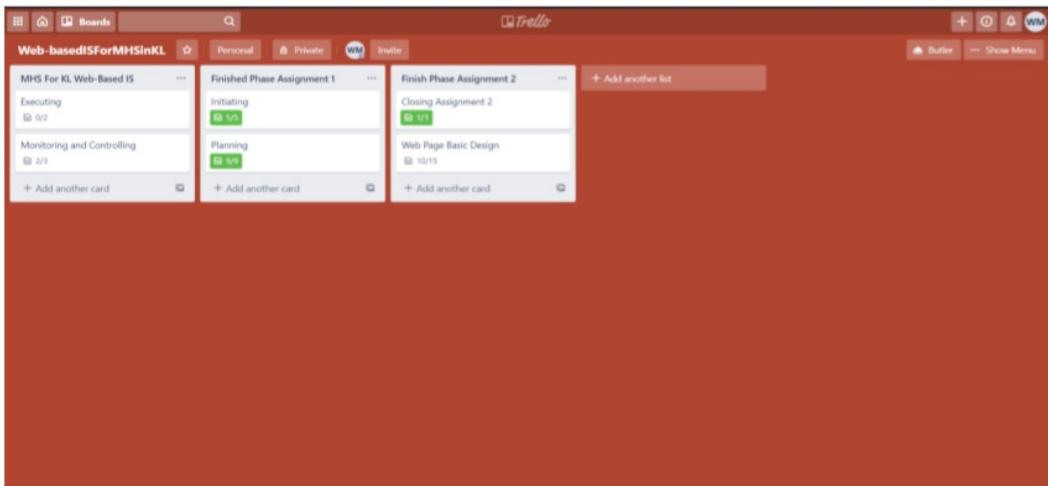
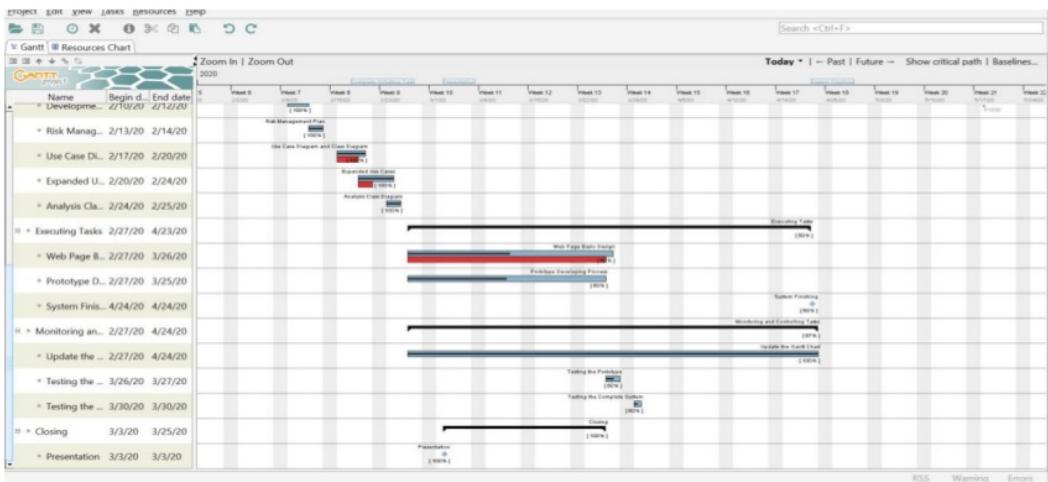
Test Analysis Report

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](#)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Web-Based Information System for MicroHousing System in Kuala Lumpur

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 | 5216e0e | 🔗 |
| aldobagus725 committed 38 seconds ago | | |
| edit word | 1aa8f07 | 🔗 |
| LuhWulandari committed 1 minute ago | | |
| Update Gantt Chart BIT302.gan | a825451 | 🔗 |
| LuhWulandari committed 4 minutes ago | | |
| Delete ~\$Edit Applications.-vsdx | 468ce68 | 🔗 |
| LuhWulandari committed 5 minutes ago | | |
| edit applications done | 7022ef1 | 🔗 |
| LuhWulandari committed 7 minutes ago | | |
| Merge branch 'master' of https://github.com/aldobagus725/WebBased_InfoSys_MHSinKL | ca28699 | 🔗 |
| LuhWulandari committed 27 minutes ago | | |
| edit the unit testing | 7ab7309 | 🔗 |
| LuhWulandari committed 27 minutes ago | | |
| PPT FIX | e77b27e | 🔗 |
| aldobagus725 committed 28 minutes ago | | |
| Merge branch 'master' of https://github.com/aldobagus725/WebBased_InfoSys_MHSinKL | 4481948 | 🔗 |
| aldobagus725 committed 2 hours ago | | |
| PPT | 107087c | 🔗 |
| aldobagus725 committed 2 hours ago | | |
| Merge branch 'master' of https://github.com/aldobagus725/WebBased_InfoSys_MHSinKL | 5b7521c | 🔗 |
| LuhWulandari committed 2 hours ago | | |

ASS2_BIT302_E1700882_E1700873_GroupAssignment

ORIGINALITY REPORT



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