

# ASS3\_BIT302\_E1700882\_E170 0873\_GroupAssignment - revised

*by* Rivaldo Soepardhy

---

**Submission date:** 04-May-2020 12:53AM (UTC+0800)

**Submission ID:** 1314645239

**File name:** ASS3\_BIT302\_E1700882\_E1700873\_GroupAssignment\_-\_revised.docx (46.64M)

**Word count:** 5480

**Character count:** 30826

# **BIT302**

# **Software Engineering**



## **ASSIGNMENT 3**

### **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
Changes & Updates .....	2
Use Case Diagram .....	3
Architectural Design .....	4
Class Diagram Design .....	5
Entity Relationship Diagram (ERD) .....	6
Sitemap.....	7
Wireframe.....	8
System Sequence Diagram + Contracts .....	21
Database Design.....	40
Iteration 2 .....	41
Use Case .....	41
Test Objectives .....	42
Test Plan .....	42
Unit Testing .....	43
Integration Testing .....	72
System Testing .....	88
Test Analysis Report .....	102
Conclusion.....	103
Updated Gantt Chart, Trello Board, GitHub .....	105

## **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.

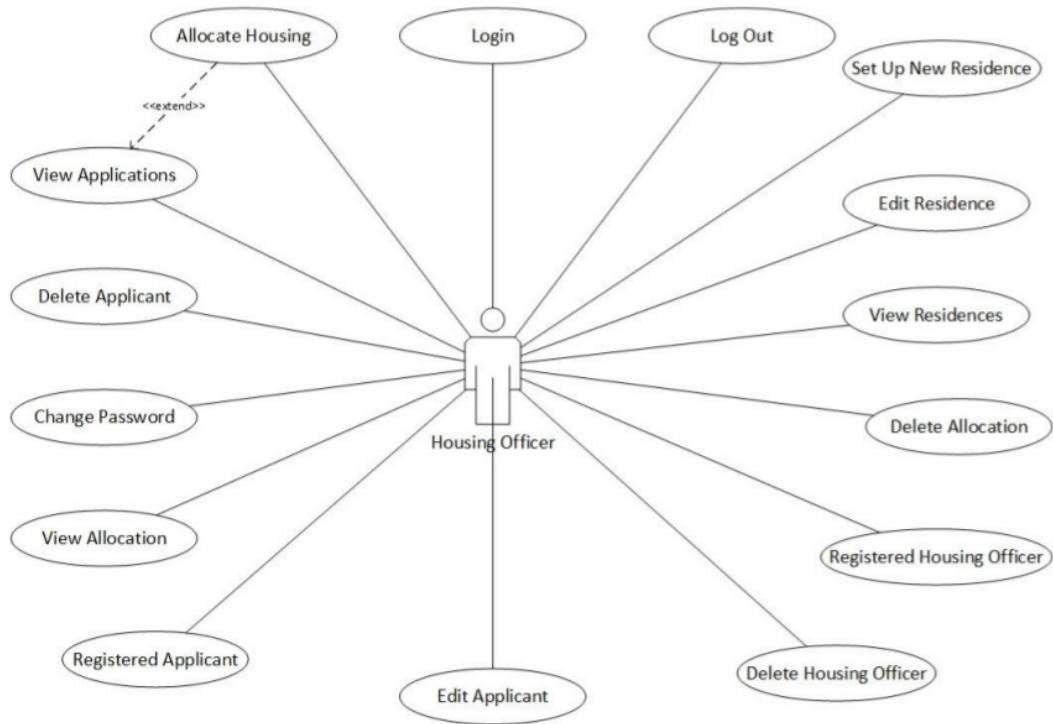
## **Changes & Updates**

We have been encountering many bugs and errors due to unexpected obstacles along the way. Because of that, we finally decided to change some of our diagrams & design to follow up with new ways that we use to accomplish the task.

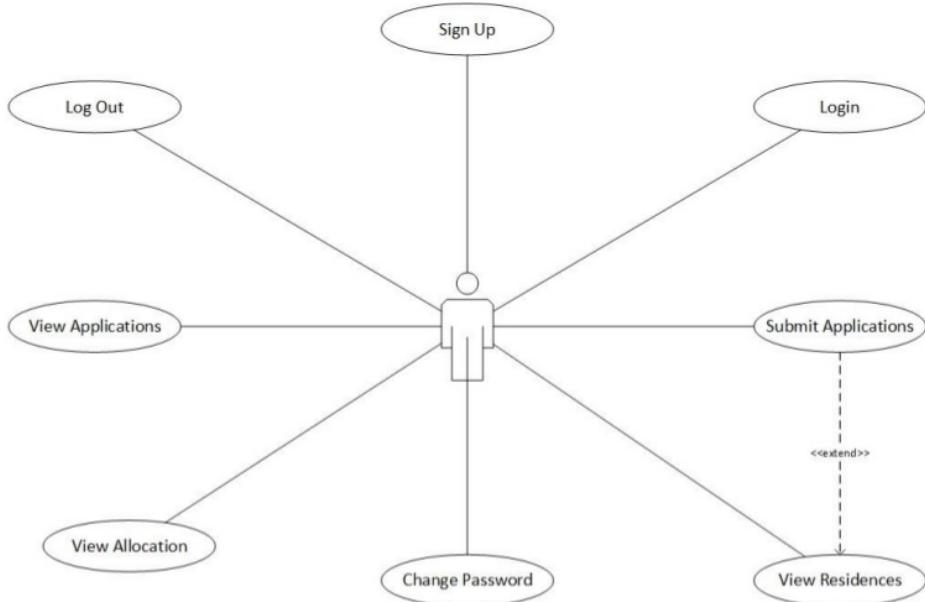
From Assignment 1 to 2, we have already update Use Case Diagram, Class Diagram, SSD, ERD, Database Design, Sitemap, and Wireframe. We also have several updates regarding previous Iteration that will be applied in the Iteration 2. The most problem we encounter is that we never expect such hard path or bugs, or errors that will happen when using Laravel framework, due to our short experience of coding. We also still have confusion regarding database design that is hard to implement on real framework such as Laravel that requires deeper coding skills.

The solution that we found is to try a workaround for the coding, and made several changes to designs that after a while, is considered reasonable than previous design.

## Use Case Diagram

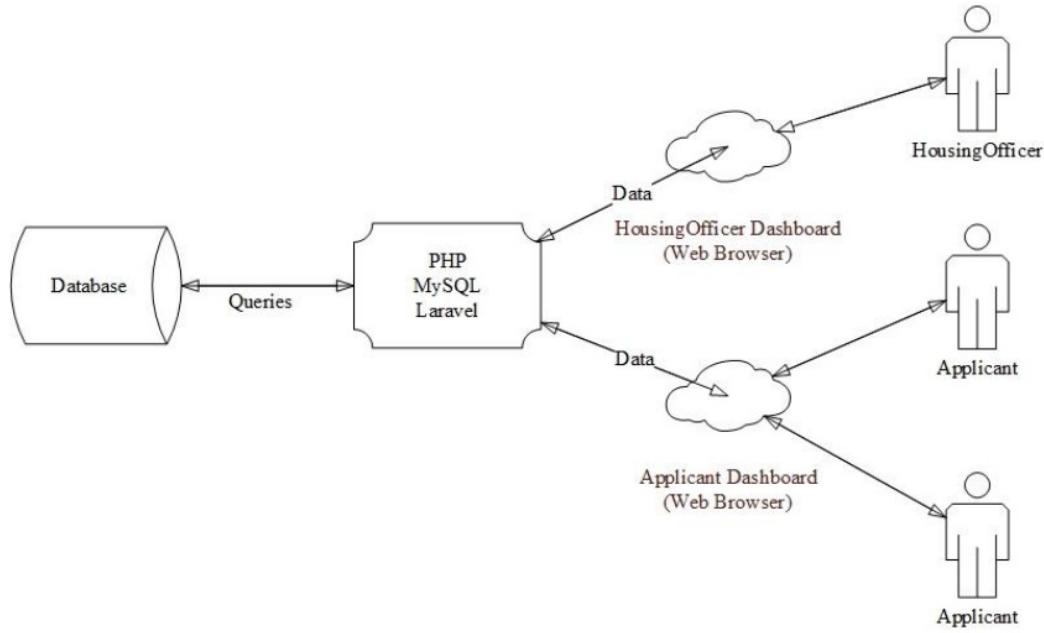


*Use Case Diagram with Actor HousingOfficer (Updated)*

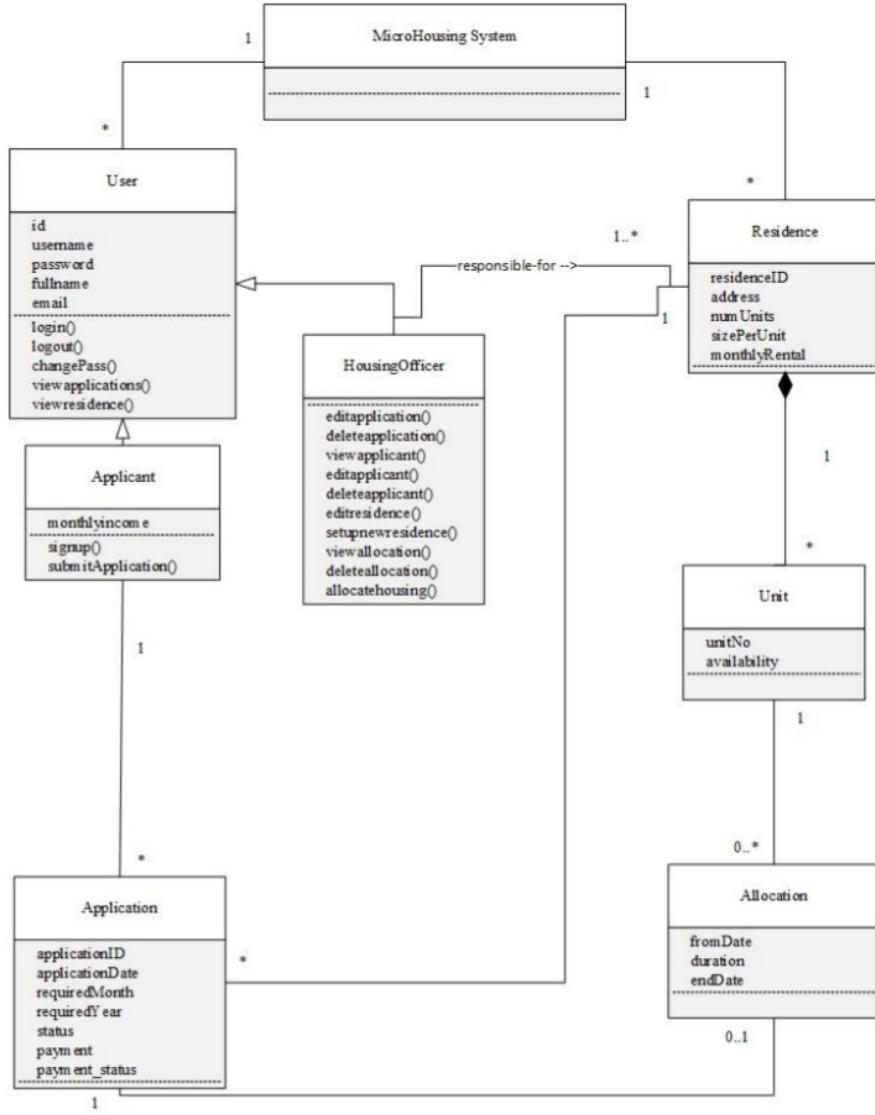


*Use Case Diagram with Actor Applicant (Updated)*

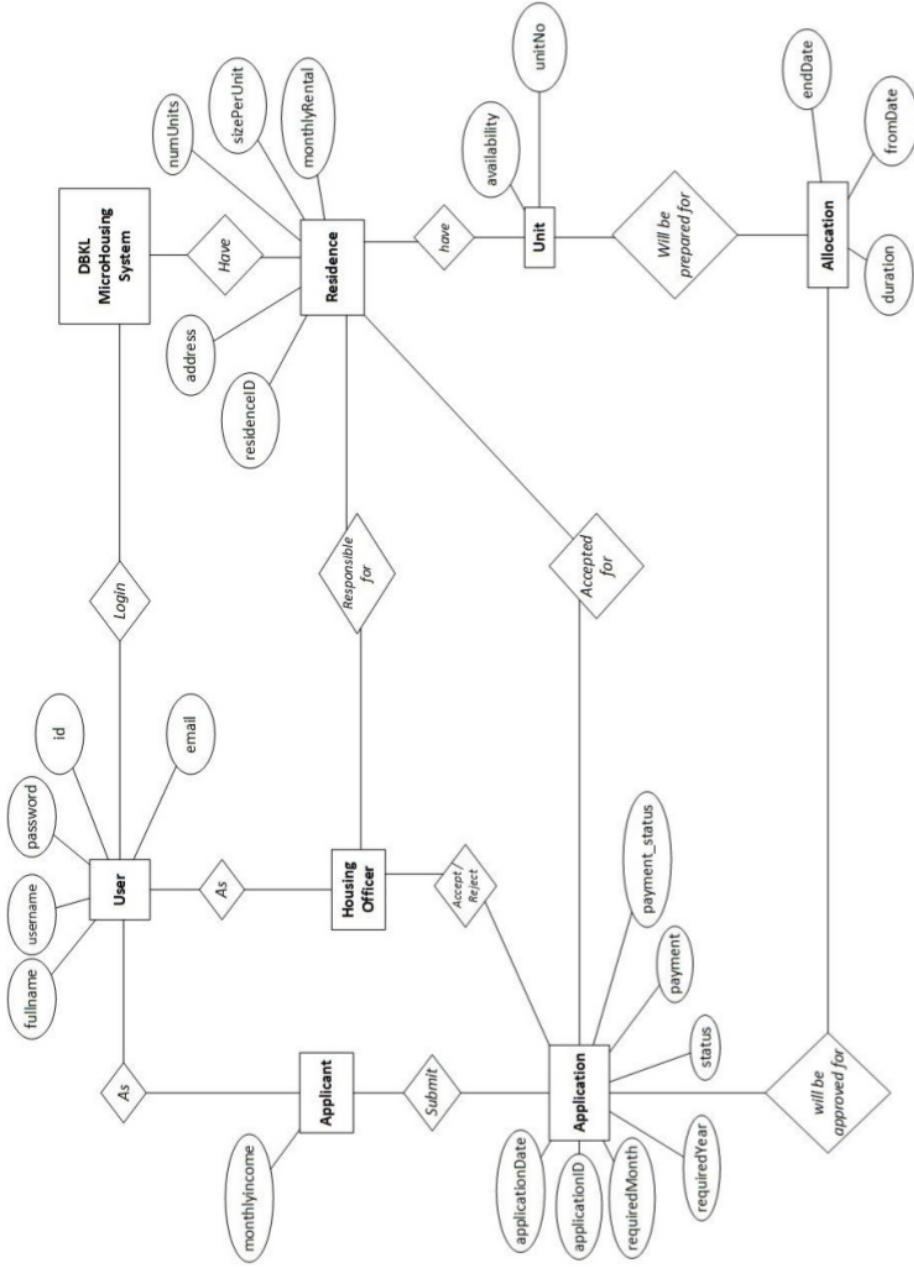
## Architectural Design



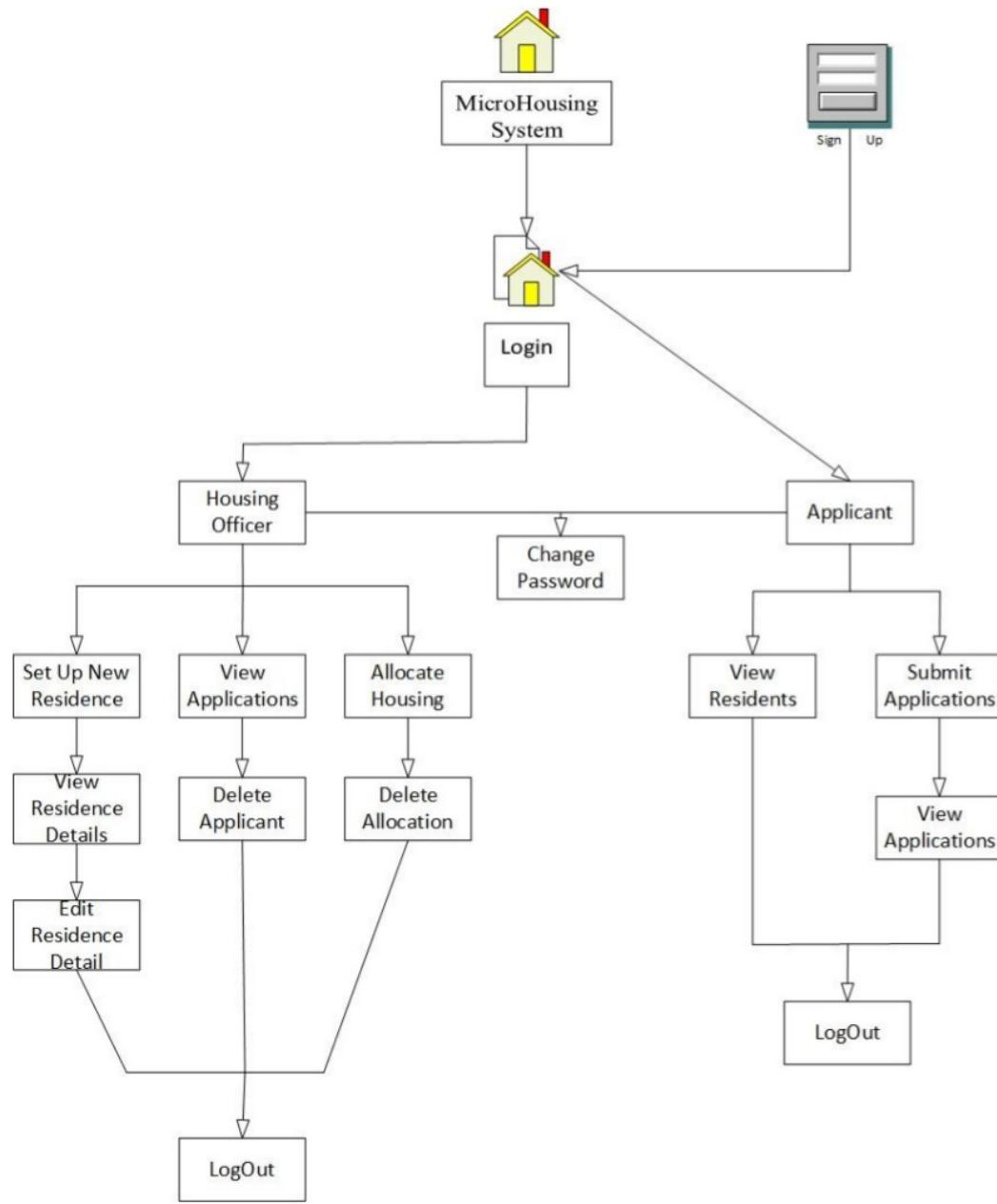
## Class Diagram Design



## Entity Relationship Diagram (ERD)



## Sitemap



## Wireframe

### Housing Officer

<b>Logo</b>  	<p>Account name</p> <p>Application ID</p> <p>Unit No</p> <p>From Date</p> <p>Duration</p> <p>End Date</p> <p style="text-align: right;"><b>Allocate</b></p>
--	---

*Allocate Housing*

<p>DBKL MicroHousing</p>	<p>Login      Register</p> <p>Change Password</p> <p>Current Password <input type="password"/></p> <p>New Password <input type="password"/></p> <p>Confirm New Password <input type="password"/></p> <p style="text-align: center;"><b>Change Password</b></p>
--------------------------	--

*Change Password*

Logo	Account	name
<b>Allocation</b>		
		<b>Delete</b>

*Delete Allocation*

Logo	Account	name
<b>Applicant</b>		
		<b>Delete</b>

*Delete Applicant*

Logo	Account	name
<b>Applications</b>		
		Delete

*Delete Applications*

Logo	Account	name
<b>Housing Officer</b>		
		Delete

*Delete Housing Officer*

**Logo**

Account name

Applicant ID

Full Name

Username

Monthly Rental

**Update**

This form is used to edit an applicant's details. It features a header with 'Logo' and 'Account name'. Below this are four input fields: 'Applicant ID', 'Full Name', 'Username', and 'Monthly Rental'. A pink oval-shaped 'Update' button is positioned at the bottom right.

*Edit Applicant*

**Logo**

Account name

User ID

Residence ID

Required Month

Required Year

**Update**

This form is used to edit application details. It features a header with 'Logo' and 'Account name'. Below this are four input fields: 'User ID', 'Residence ID', 'Required Month', and 'Required Year'. A pink oval-shaped 'Update' button is positioned at the bottom right.

*Edit Applications*

The screenshot shows a user interface for editing residence information. On the left, there is a placeholder for a logo consisting of six horizontal white bars. To the right, a light blue header bar contains the text "Account" and "name". Below this, a dark blue section contains five input fields: "Residence ID", "Address", "Number of Units", "Size of Unit", and "Monthly Rental". A pink oval-shaped button labeled "Update" is positioned at the bottom right of this section.

Logo

Account name

Residence ID

Address

Number of Units

Size of Unit

Monthly Rental

Update

*Edit Residence*

The screenshot shows the homepage of DBKL MicroHousing. At the top, a light blue header bar includes the website name "DBKL MicroHousing" and links for "Home", "About Us", "Gallery", "Contact Us", "Login", and "Register". The main content area below the header is a large, solid dark blue rectangle with a white rectangular placeholder in its center.

DBKL MicroHousing

Home About Us Gallery Contact Us Login Register

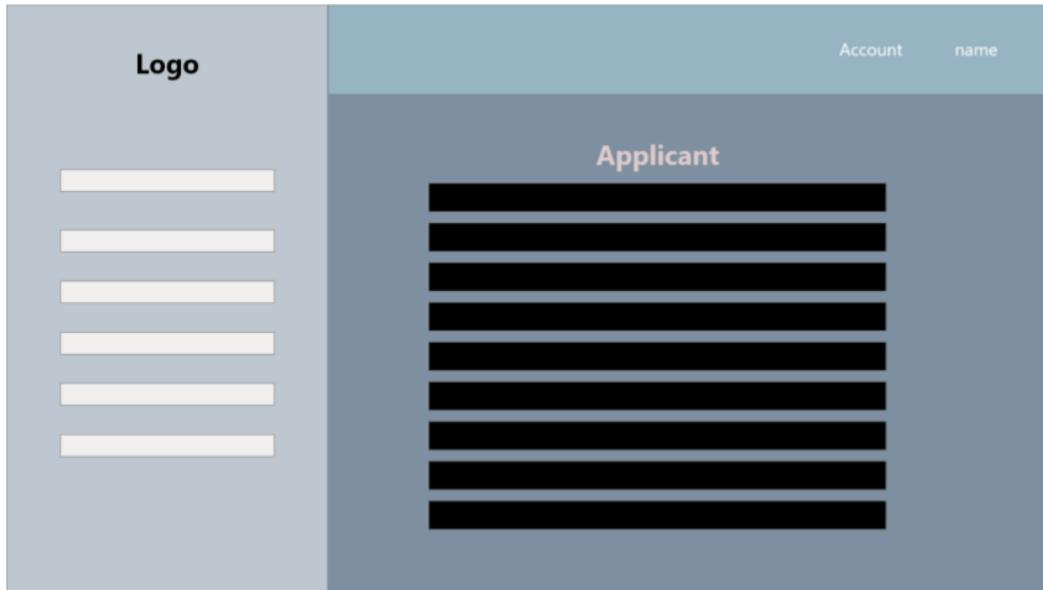
*Homepage*



*Logout Housing Officer*

A screenshot of a login page. At the top, there is a header bar with the text "DBKL MicroHousing" on the left and "Login" and "Register" on the right. Below the header is a large central form area. The form has a title "Login" at the top. It contains two input fields: "E-mail Address" and "Password". Below these fields is a checkbox labeled "Remember me". At the bottom of the form are two buttons: "Submit" and "Forgot Password?".

*Login*



*Registered Applicant*



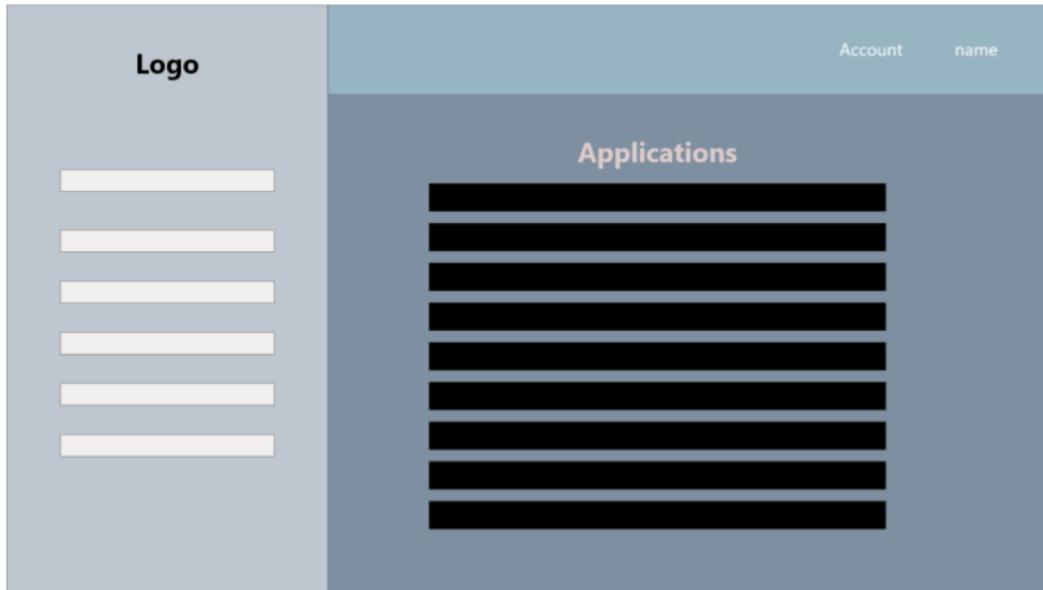
*Registered Housing Officer*

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

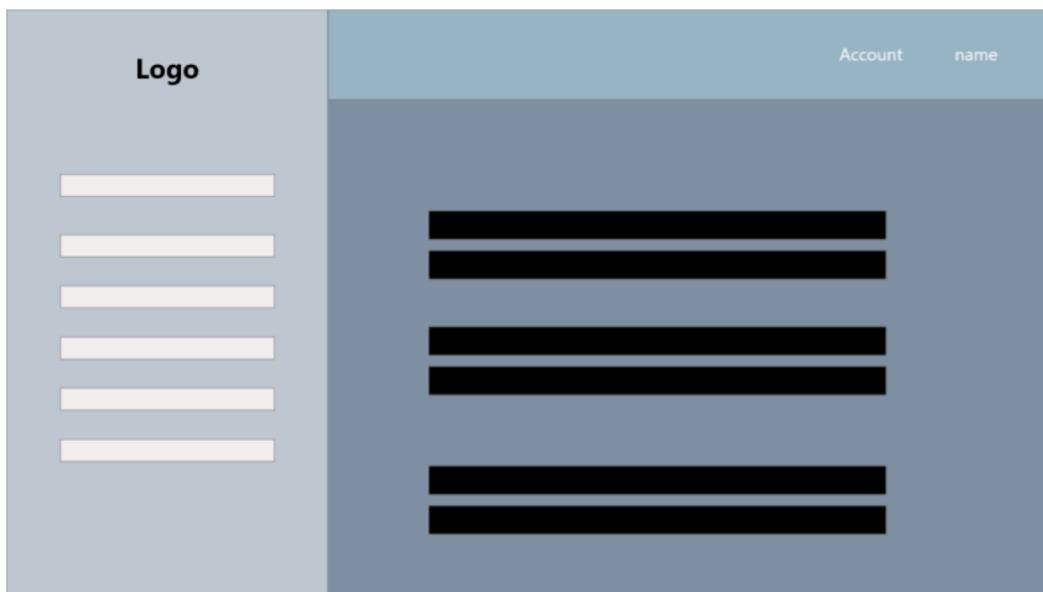
*Set Up New Residence*

A user interface showing a list of applications for allocation. On the left, there is a vertical grey sidebar with the word "Logo" at the top. The main area has a light blue header bar with the text "Account name". Below this is a dark blue section titled "Applications" which lists eight items, each represented by a black horizontal bar. To the right of each bar is a white rectangular button with the word "Allocate".

*View Allocation*

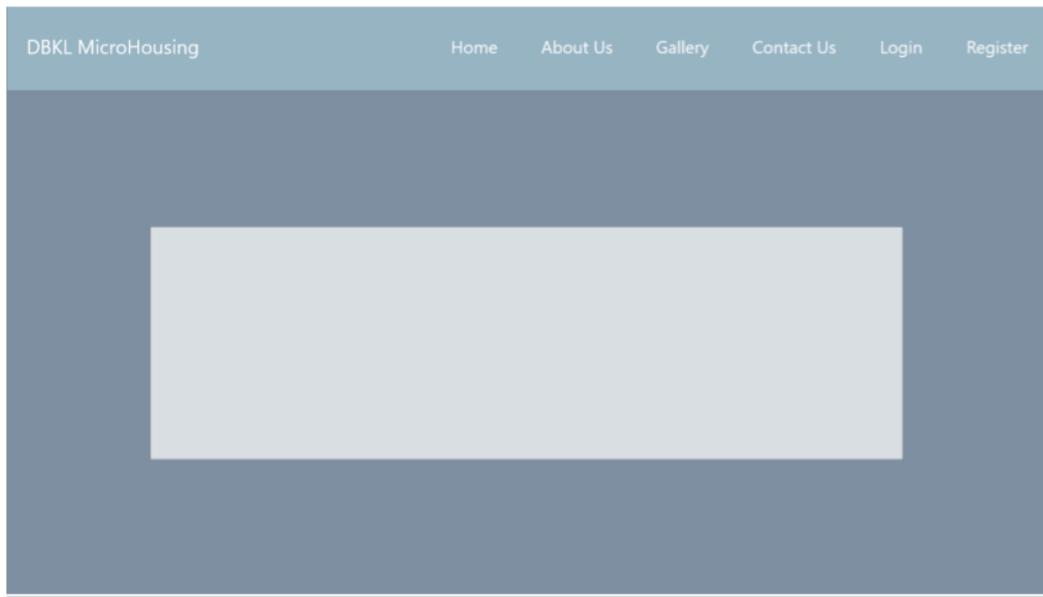


*View Applications*



*View Residence Housing Officer*

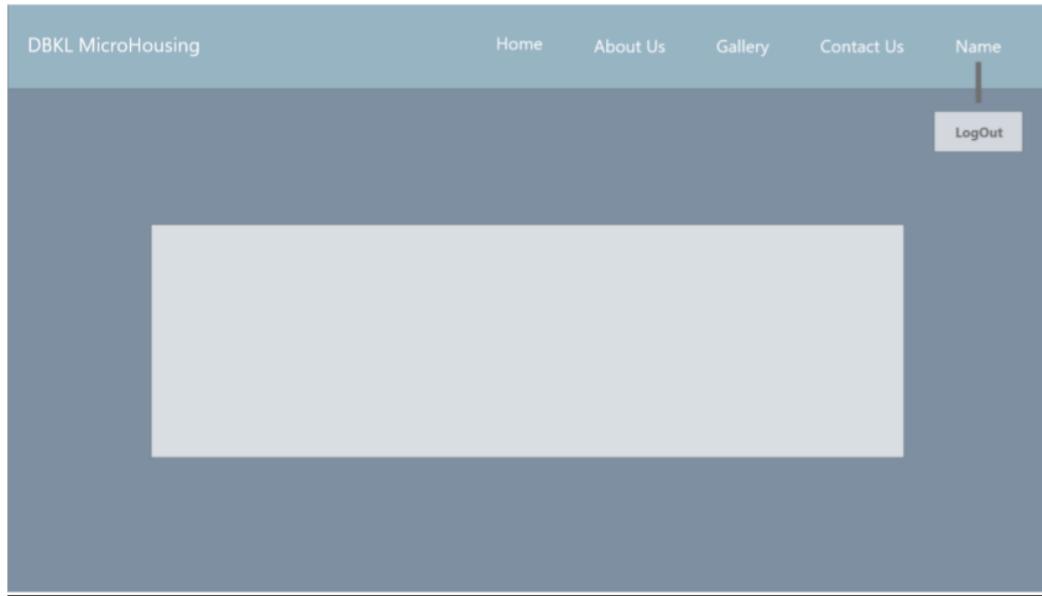
## Applicant



*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

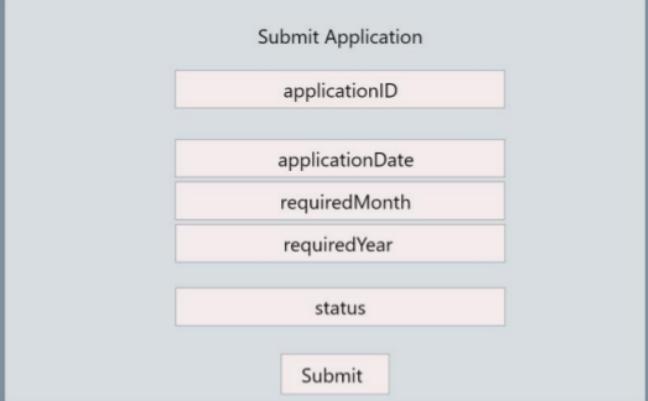
applicationDate

requiredMonth

requiredYear

status

Submit



This screenshot shows a 'Submit Application' form. It consists of five input fields: applicationID, applicationDate, requiredMonth, requiredYear, and status. Below these fields is a 'Submit' button.

*Submit Applications*

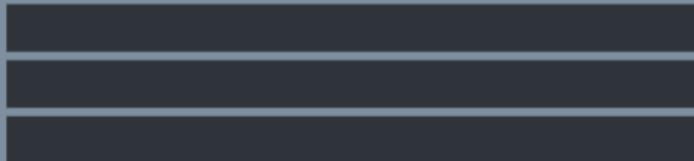
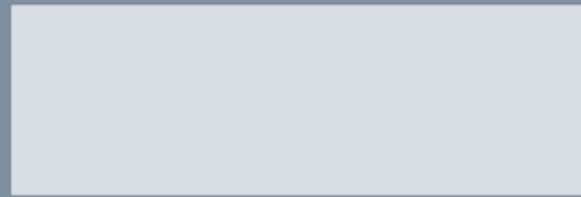
DBKL MicroHousing

Home   About Us   Gallery   Contact Us   Login   Register



This screenshot shows a 'View Applications' interface. It features a large white rectangular area with three dark grey horizontal bars stacked vertically in its center.

*View Applications*

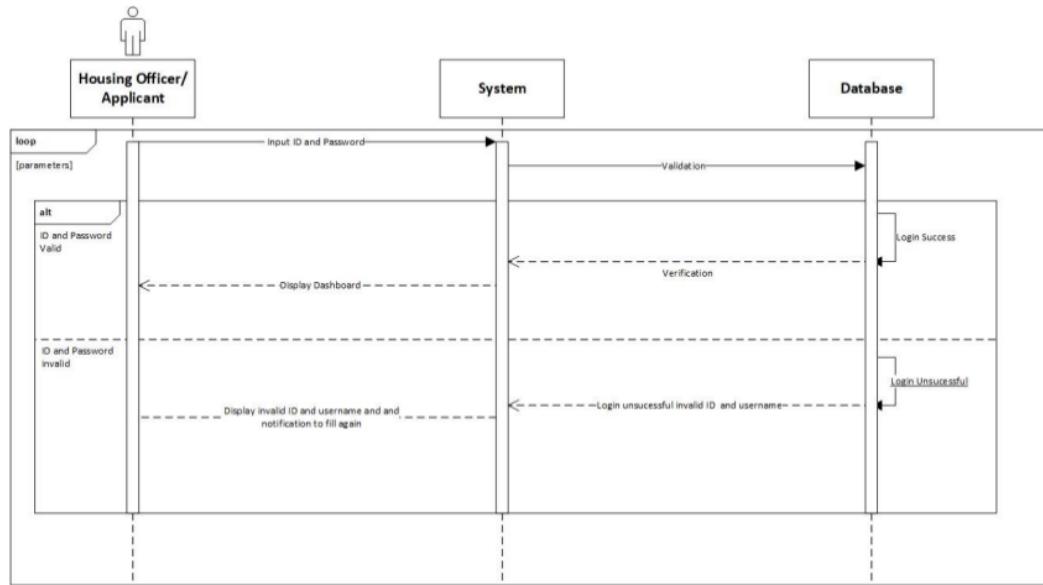


Submit Application  
Submit Application  
Submit Application

*View Residence*

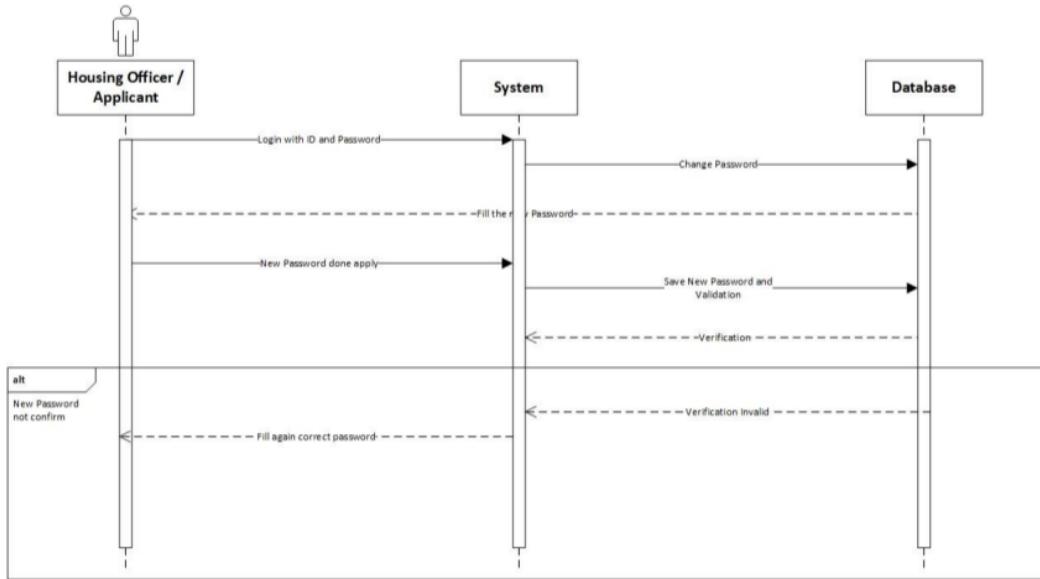
## System Sequence Diagram + Contracts

### 1. Login



Prepared by: Luh Wulandari Maharani

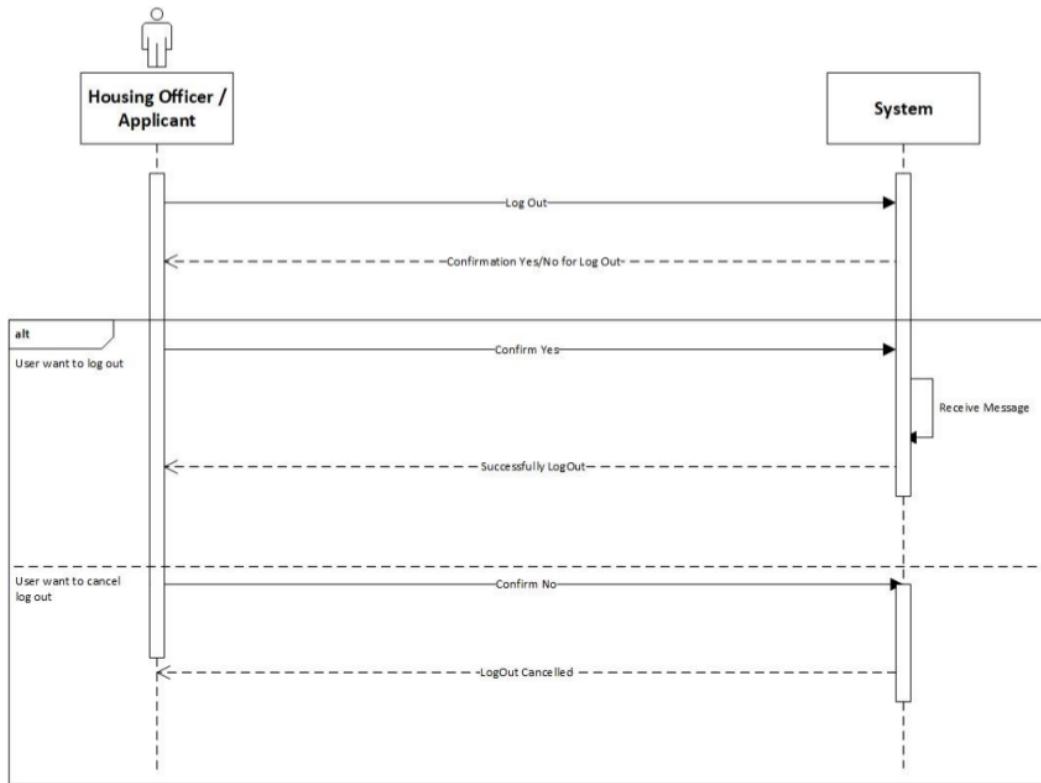
## 2. Change Password



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

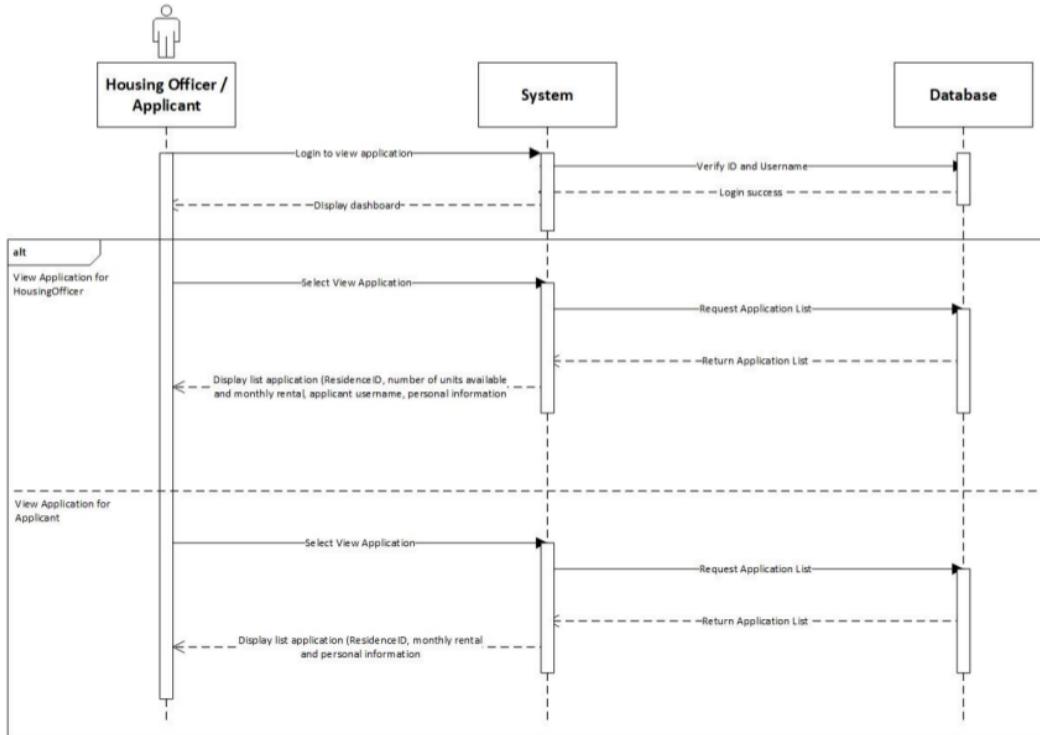
### 3. Log Out



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

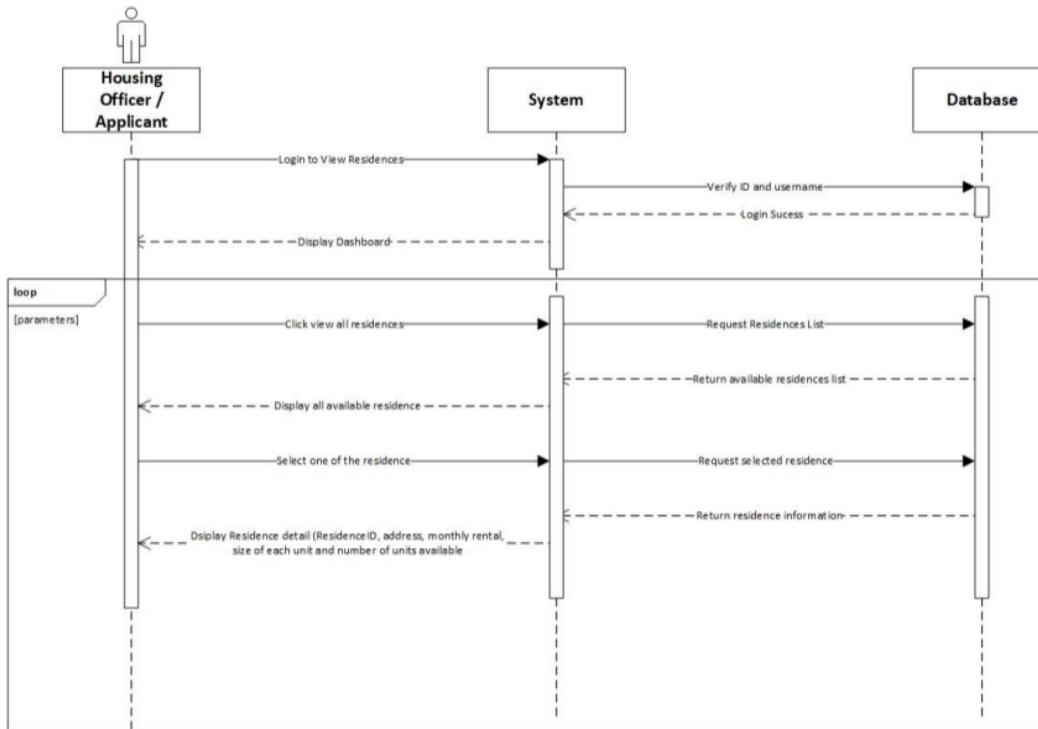


Prepared by: Luh Wulandari Maharani

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

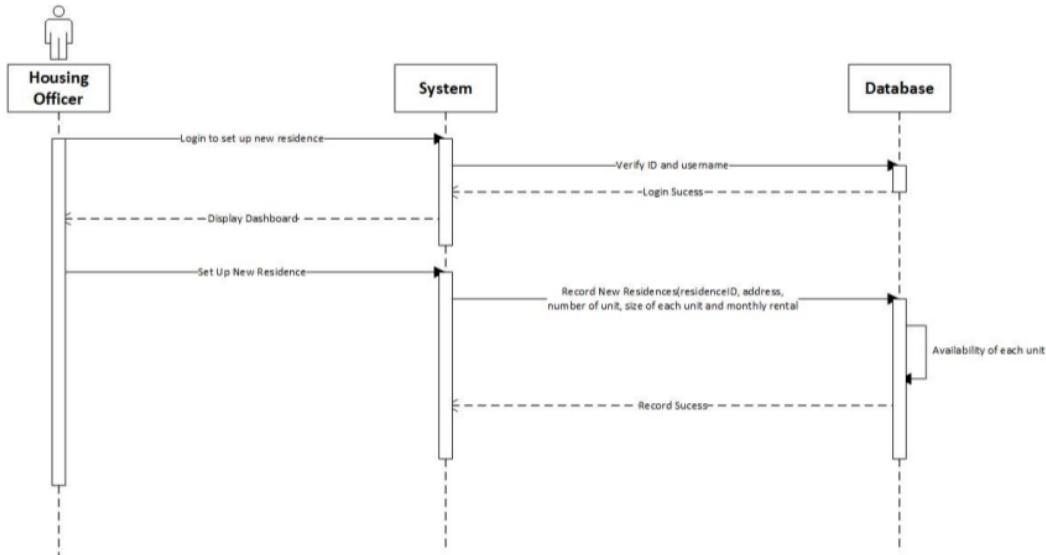


Prepared by: Luh Wulandari Maharani

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

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

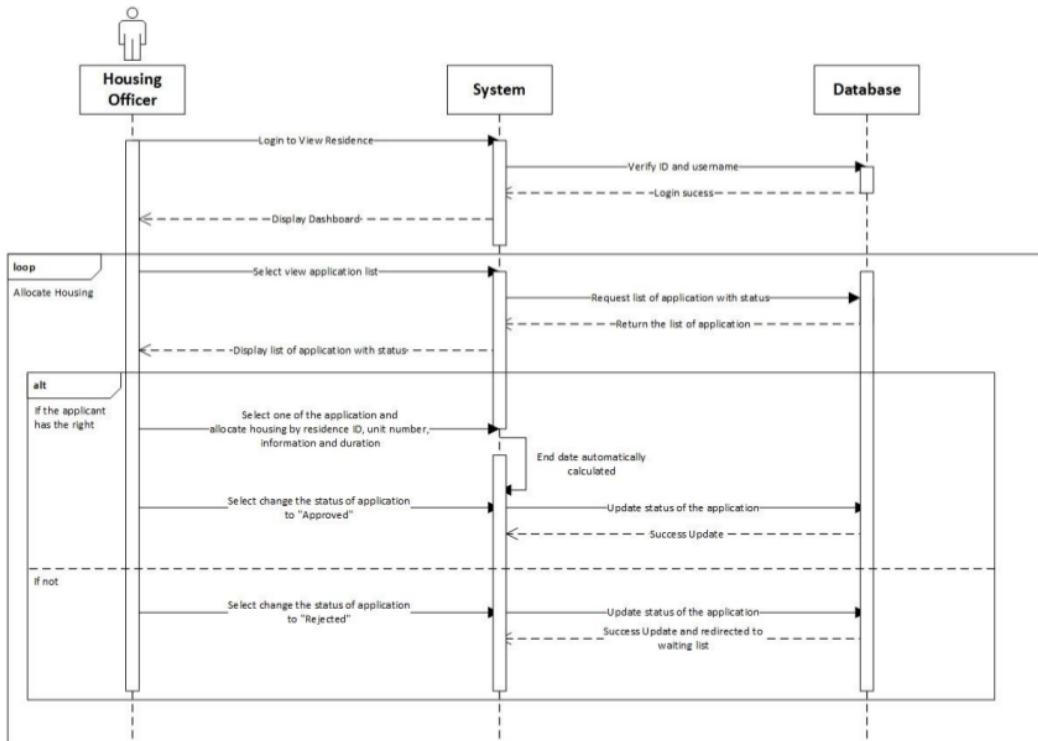


Prepared by: Luh Wulandari Maharani

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"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>

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

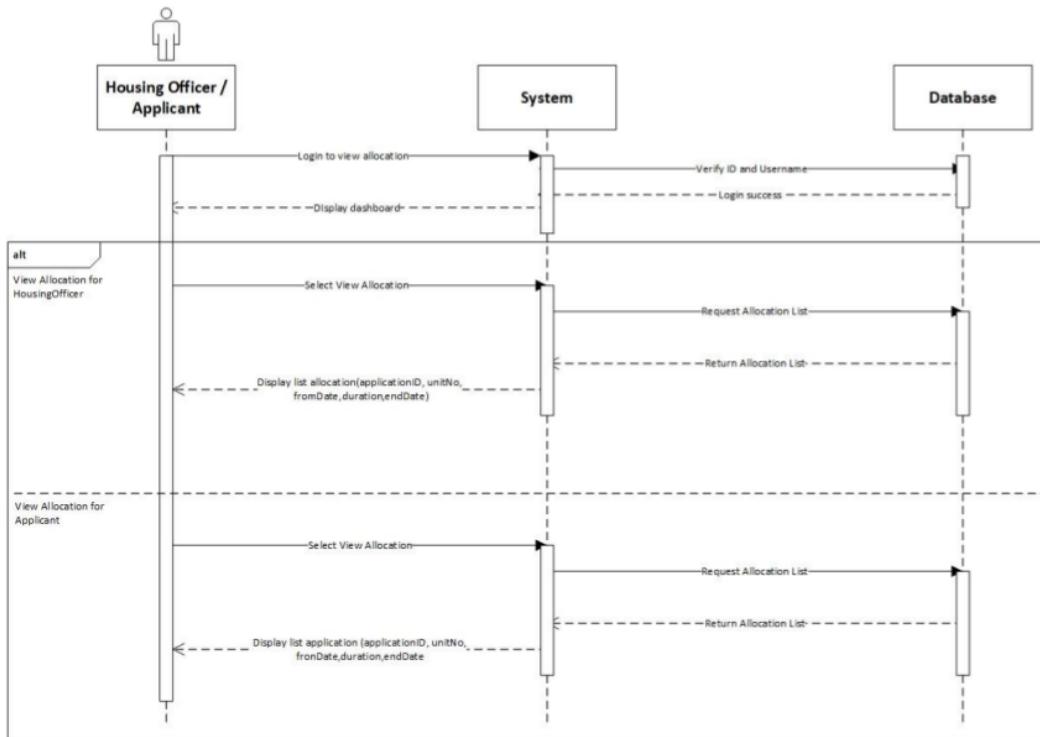


Prepared by: Luh Wulandari Maharani

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

<b>Cross References</b>	<b>Allocate Housing</b>
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
<b>Cross References</b>	<b>Allocate Housing</b>
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
<b>Cross References</b>	<b>Allocate Housing</b>
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
<b>Cross References</b>	<b>Allocate Housing</b>
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. View Allocation

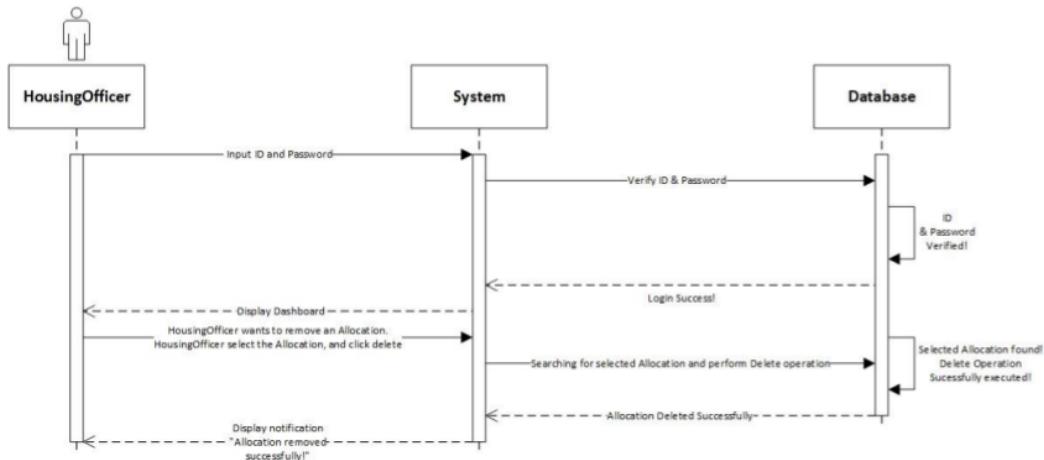


Prepared by: Luh Wulandari Maharani

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

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

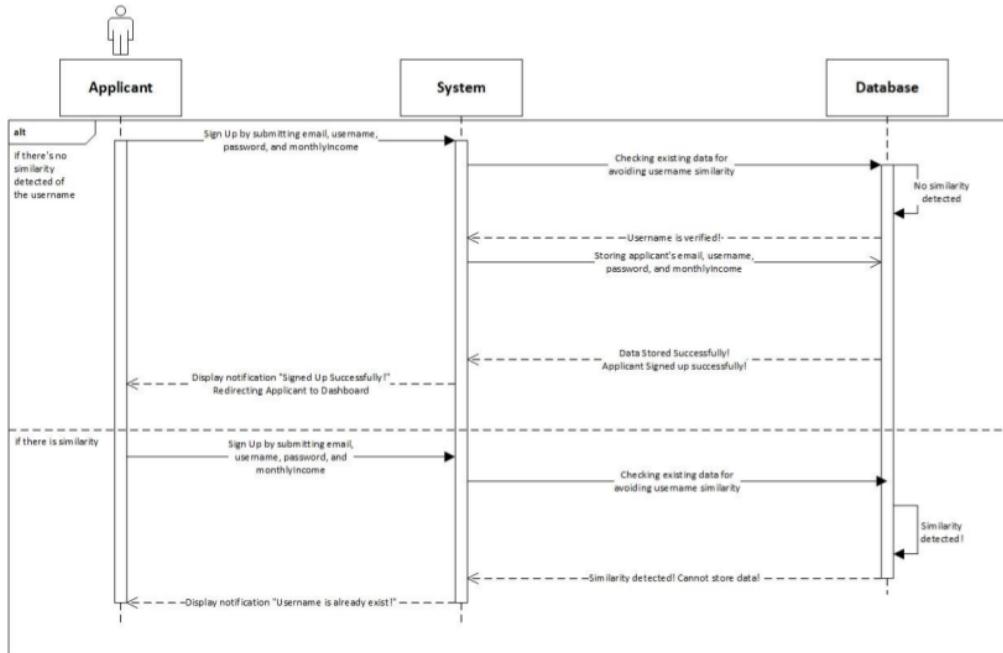
## 9. Delete Allocation



Prepared by: Luh Wulandari Maharani

Cross References	Delete Allocation
Operation	Login with ID and username
Responsible	To access dashboard, and delete allocations
Pre-conditions	<ul style="list-style-type: none"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
Cross References	Delete Allocation
Operation	Delete Allocation
Responsible	To remove an allocation
Pre-conditions	The allocation object must be available, and the allocation is no longer in place
Post-conditions	Allocation object will be removed successfully

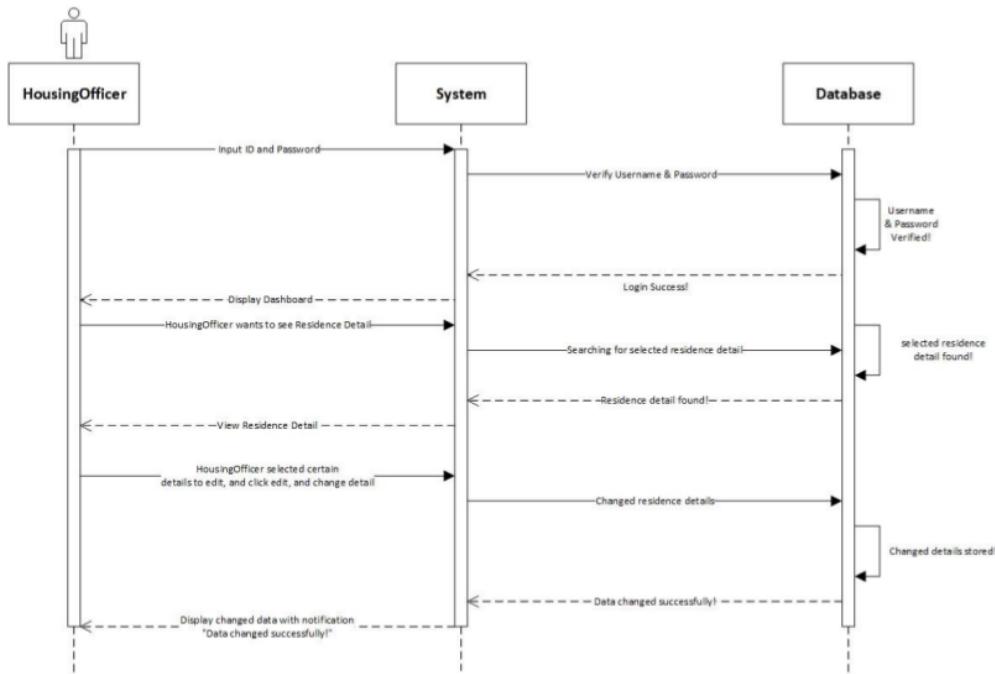
## 10. Sign Up Applicant



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"> <li>Email must be available</li> <li>Username must be available</li> <li>Password must be available</li> <li>MonthlyIncome must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>Username is verified (no similarity detected)</li> <li>Registered in System</li> <li>Display dashboard</li> </ul>
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"> <li>Email must be available</li> <li>Username must be available</li> <li>Password must be available</li> <li>MonthlyIncome must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>Username is not verified (already existed in system)</li> <li>Applicant must re-enter new username.</li> </ul>

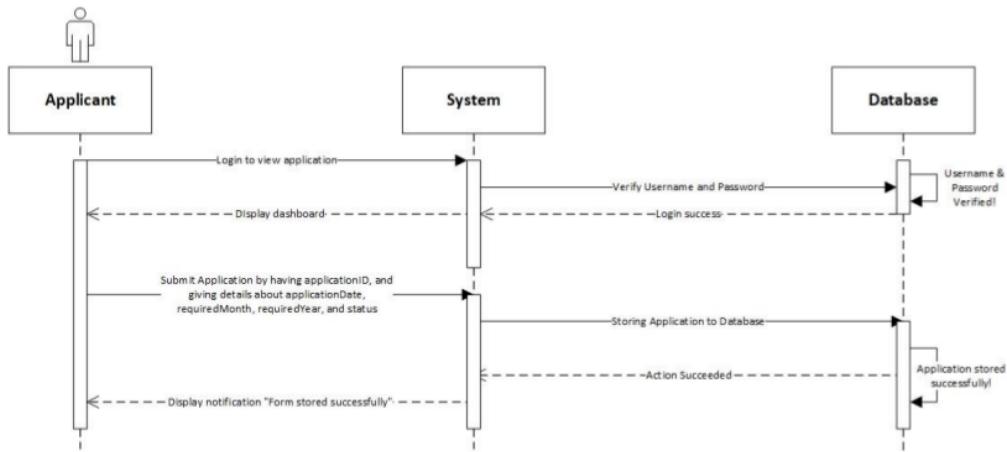
## 11. Edit Residence



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"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
Cross References	Edit Residence Detail
Operation	Edit residenceID, address, numUnits, sizePerUnit, monthlyRental
Responsible	To change certain Residence Detail
Pre-conditions	We will need <b>one of or all of</b> the details below: <ul style="list-style-type: none"> <li>• Email must be available</li> <li>• Username must be available</li> <li>• Password must be available</li> <li>• MonthlyIncome must be available</li> </ul>
Post-conditions	Residence Detail changed successfully

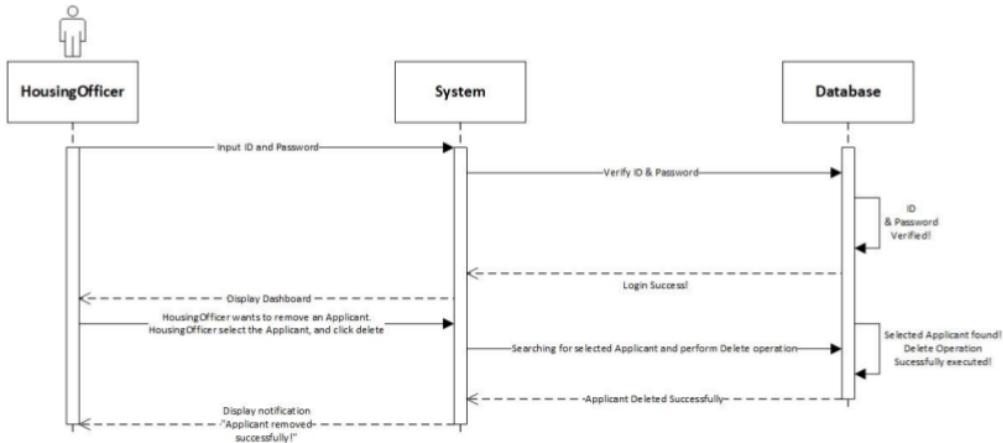
## 12. Submit Application



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"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
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"> <li>• Applicant is already logged in to the system</li> <li>• Applicant has already determine the desired residence</li> </ul>
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 <b>all of</b> the details below:</p> <ul style="list-style-type: none"> <li>• applicationID will be given by system automatically</li> <li>• applicationDate must be available</li> <li>• requiredMonth must be available</li> <li>• requiredYear must be available</li> <li>• status will be given by system automatically</li> </ul>
Post-conditions	Application will be stored successfully, and will be on waiting list for the approval & arrangement by HousingOfficer.

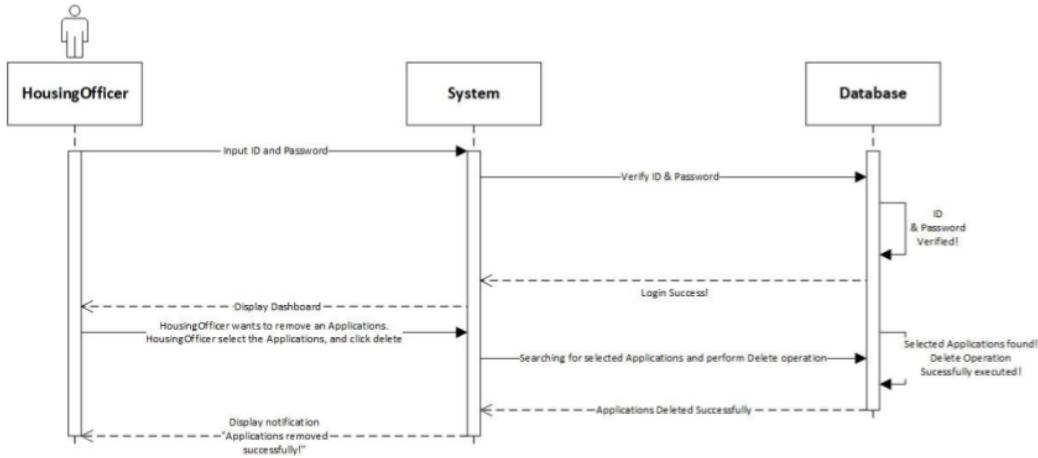
### 13. Delete Applicant



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"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
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

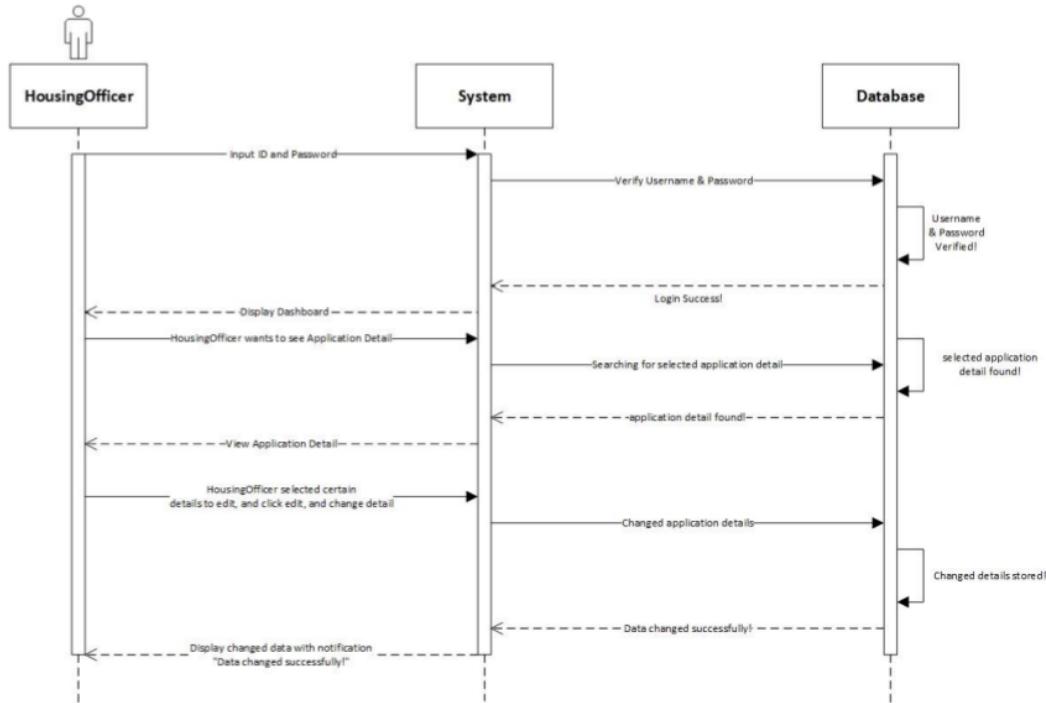
## 14. Delete Applications



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"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
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

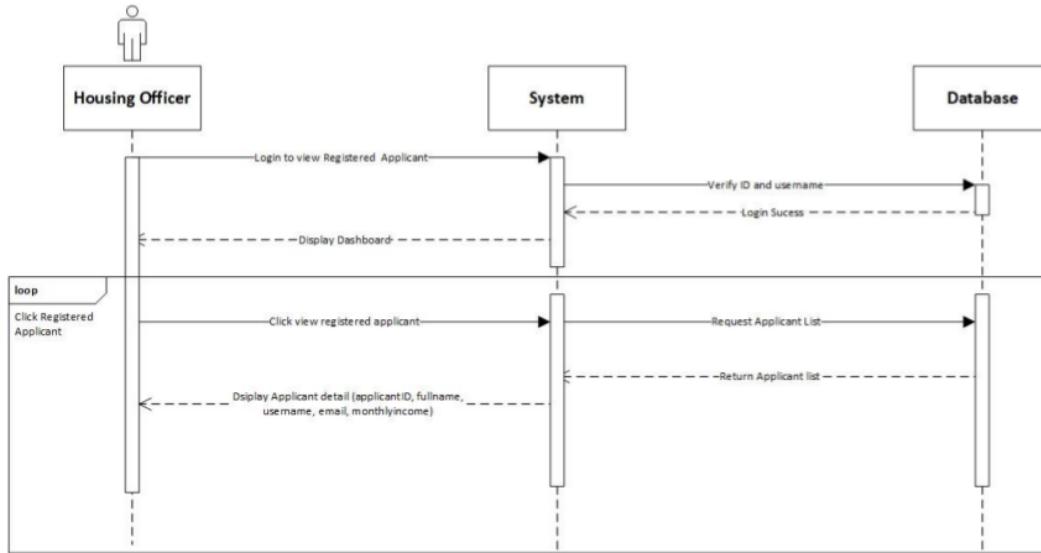
## 15. Edit Applications



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"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
Cross References	Edit Applications Detail
Operation	Edit status, payment and payment status
Responsible	To change certain Applications Detail
Pre-conditions	We will need <b>one of</b> or <b>all of</b> the details below: <ul style="list-style-type: none"> <li>• applicantID must be available</li> <li>• residenceID must be available</li> <li>• requiredMonth must be available</li> <li>• requiredYear must be available</li> </ul>
Post-conditions	Applications changed successfully

## 16. View Applicant

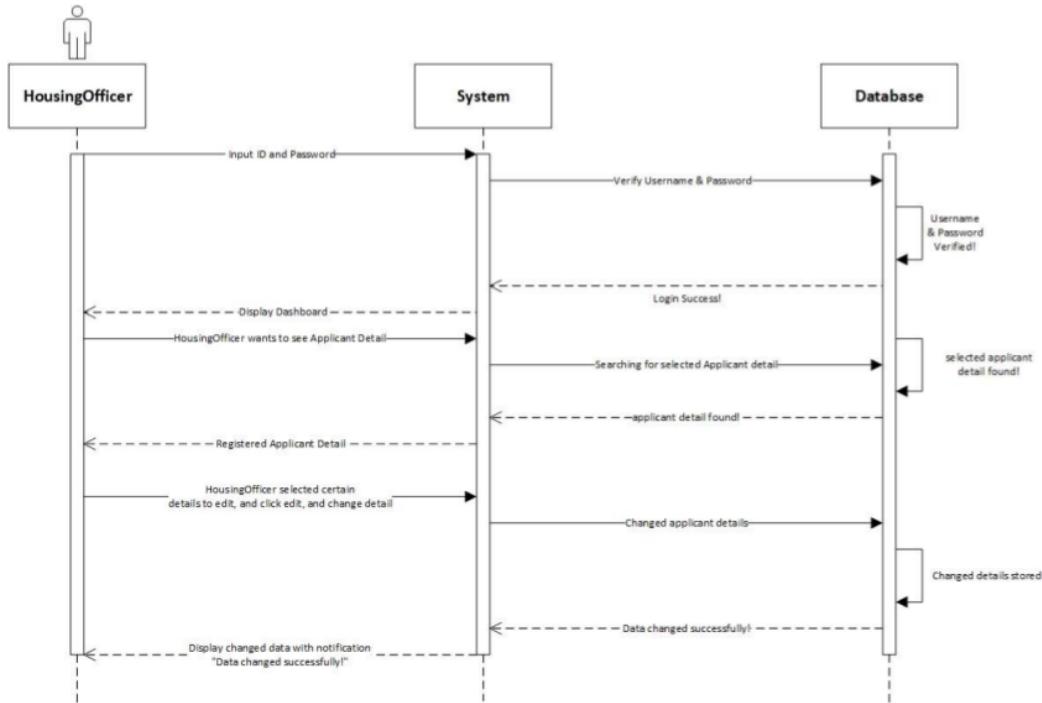


Prepared by: Luh Wulandari Maharani

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

Cross References	Registered Applicant
Operation	Click view all registered applicant
Responsible	To get all list of the applicant
Pre-conditions	The applicant object must be available
Post-conditions	Success to display all applicant

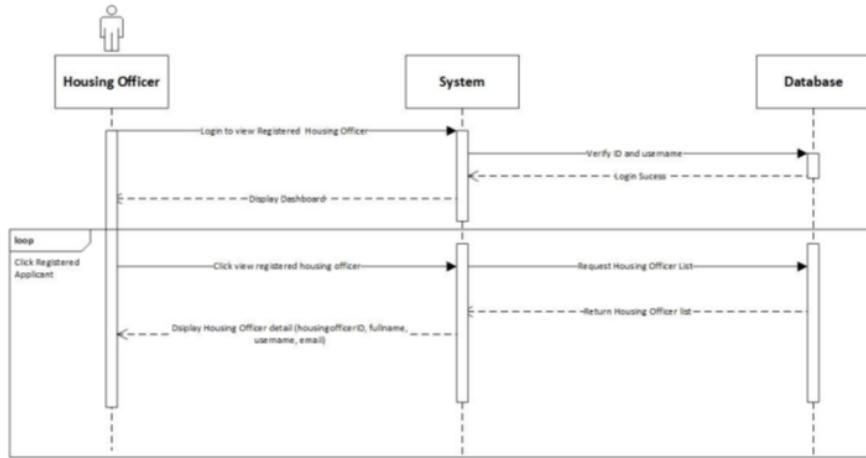
## 17. Edit Applicant



Prepared by: Luh Wulandari Maharani

Cross References	Edit Applicant
Operation	Login with username and password
Responsible	To get access to view the applications
Pre-conditions	<ul style="list-style-type: none"> <li>• Username must be available</li> <li>• Password must be available</li> </ul>
Post-conditions	<ul style="list-style-type: none"> <li>• Username is matched</li> <li>• Password must be match based on user's password input</li> <li>• Display dashboard</li> </ul>
Cross References	Edit Applicant
Operation	Edit status, payment and payment status
Responsible	To change certain Applications Detail
Pre-conditions	We will need <b>one of</b> or <b>all of</b> the details below: <ul style="list-style-type: none"> <li>• applicantID must be available</li> <li>• fullName must be available</li> <li>• userName must be available</li> <li>• email must be available</li> <li>• monthlyRental must be available</li> </ul>
Post-conditions	Applicant changed successfully

## 18. View Housing Officer

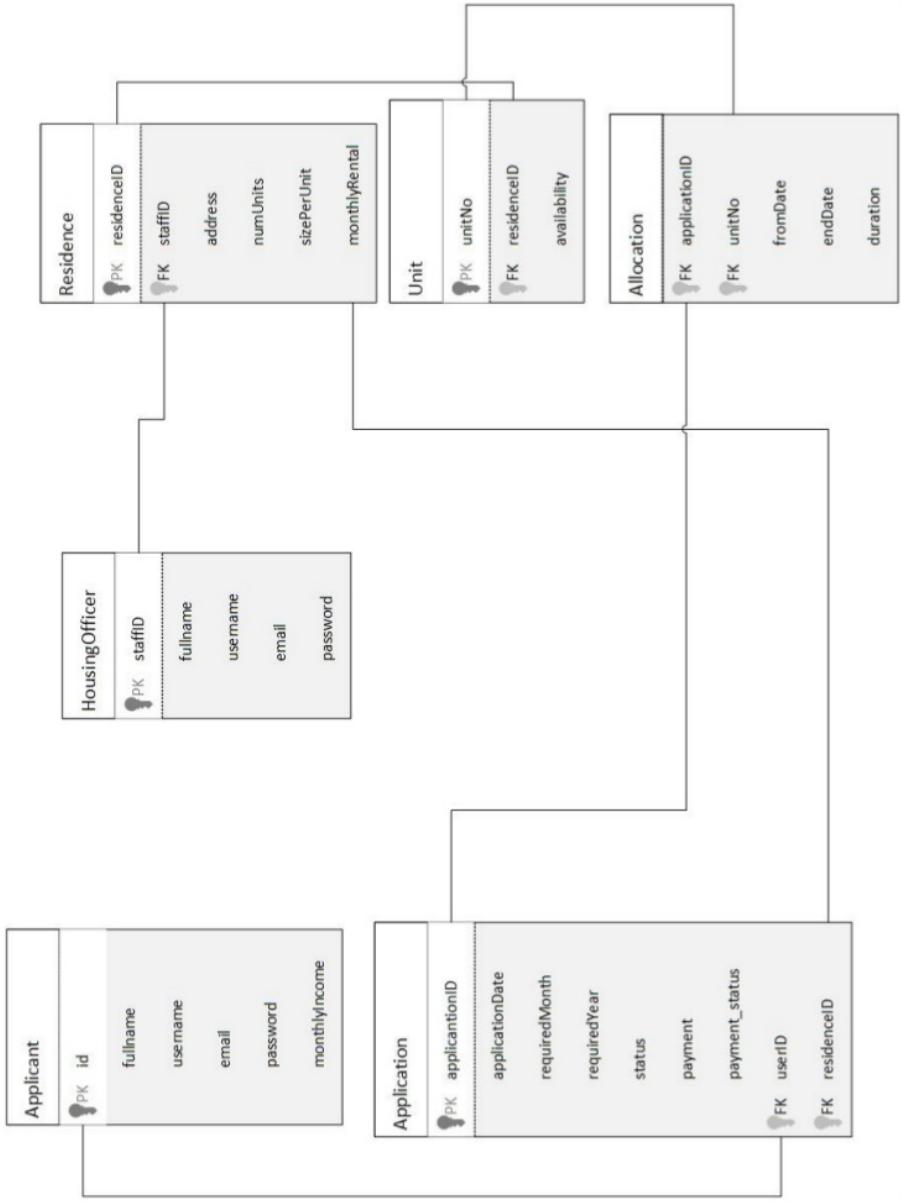


Prepared by: Luh Wulandari Maharani

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

Cross References	Registered Housing Officer
Operation	Click view all registered housing officer
Responsible	To get all list of the housing officer
Pre-conditions	The housing officer object must be available
Post-conditions	Success to display all housing officer

## Database Design



## Iteration 2

### Use Case

Remaining Use Case	Developed By	Testing
Change Password (HousingOfficer)	Aldo	Iteration 2 1
Change Password (Applicant)	Aldo	Iteration 2
Delete Application (HousingOfficer)	Aldo & Wulan	Iteration 2
Edit Application (HousingOfficer)	Aldo & Wulan	Iteration 2
View Allocation (HousingOfficer)	Aldo & Wulan	Iteration 2
View Allocation (Applicant)	Aldo	Iteration 2
Allocate Housing(HousingOfficer)	Aldo & Wulan	Iteration 2
Delete Allocation(HousingOfficer)	Aldo & Wulan	Iteration 2 1
View Applicant(HousingOfficer)	Aldo & Wulan	Iteration 2
Edit Applicant(HousingOfficer)	Aldo	Iteration 2
Delete Applicant(HousingOfficer)	Aldo & Wulan	Iteration 2
View Housing Officer (HousingOfficer)	Aldo & Wulan	Iteration 2

## **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.

Our previous testing already proved the system has work and fulfill our first objective. Now we are fully testing the functional system from each user:

- As Housing Officer, 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. Housing Officer 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.

## **Test Plan**

For the information system, we will be testing functions that will be performed by Housing Officer 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 Housing Officer 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.

A screenshot of a web-based registration form titled "Register". The form fields are as follows:

- Name: An empty input field with a placeholder icon and the message "Please fill out this field."
- Username: An empty input field with a placeholder icon and the message "Please fill out this field."
- E-Mail Address: An empty input field.
- Monthly Income: An empty input field.
- Password: An empty input field.
- Confirm Password: An empty input field.

At the bottom is a blue "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.**

A screenshot of a web-based registration form titled "Register". The form fields are as follows:

- Name: "wulan" (valid)
- Username: "wulan" (invalid, highlighted in red) with the message "The username has already been taken."
- E-Mail Address: "Admin" (invalid, highlighted in red) with the message "Please include an '@' in the email address. 'Admin' is missing an '@'."
- Monthly Income: An empty input field.
- Password: An empty input field.
- Confirm Password: An empty input field.

At the bottom is a blue "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.**

A screenshot of a web-based registration form titled "Register". The form fields are as follows:

- Name: "wulan" (valid)
- Username: "wulan" (valid)
- E-Mail Address: "wulan" (invalid, highlighted in red) with the message "Please include an '@' in the email address. 'wulan' is missing an '@'."
- Monthly Income: An empty input field.
- Password: ".." (invalid, highlighted in red) with the message "Please include an '@' in the email address. 'wulan' is missing an '@'."
- Confirm Password: ".." (invalid, highlighted in red) with the message "Please include an '@' in the email address. 'wulan' is missing an '@'."

At the bottom is a blue "Register" button.

The screenshot shows a registration form with the following fields and values:

- Name: wulan
- Username: wulandari
- E-Mail Address: wulandari@gmail.com
- Monthly Income: 10000
- Password: (empty field)
- Confirm Password: (empty field)

An error message "The password must be at least 8 characters." is displayed below the password input field.

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
aldobagus	Housingofficer	\$2y\$10\$raEeBv7zHvNbISqH0bNiOegCjPVFefJg7m5iNk6Yir...
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</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" value="

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

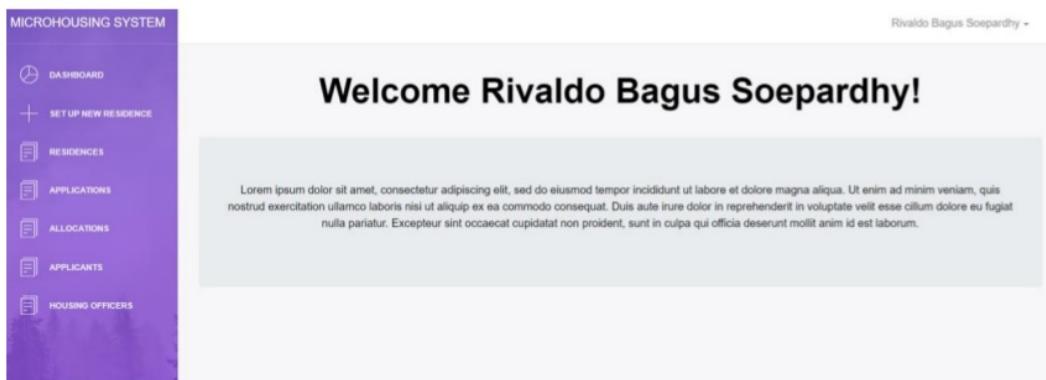

```

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



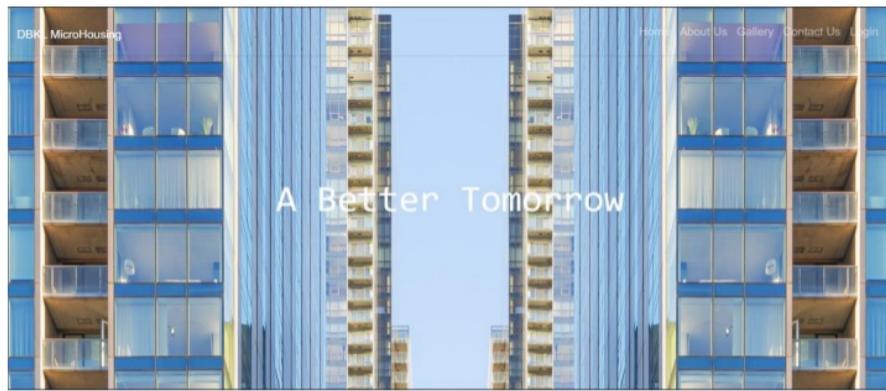
### 3. LogOut

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.

```
<div 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>
</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 addresidence.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>


```

For the num of unit connected to unit database and make a model for the declaration of unit table

```
namespace App;

use Illuminate\Database\Eloquent\Model;

class Unit extends Model{
    protected $table='unit';
    protected $primaryKey = 'unitNo';
    protected $fillable=['availability','residenceID'];
}
```

And adding unit function into to addresidence.blade.php for calculate the num of units

```
public function store(Request $request){
    $residences = new Residences();
    $residences->admin_id = $request->input('admin_id');
    $residences->address = $request->input('address');
    $residences->numunits = $request->input('numunits');
    $residences->sizeperunits = $request->input('sizeperunits');
    $residences->monthlyrental = $request->input('monthlyrental');

    $residences->save();
    for ($x = 1; $x <= $residences->numunits;$x++){
        $units = new Unit();
        $units->availability = "Available";
        $units->residenceID = $residences->residenceID;
        $units->save();
    }
    return view('Housingofficer.addresidences')->with('Housingofficer.addresidences',$residences);
}
```

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

This is the residence data in ‘microhousing’ database.

Options	residenceID	admin_id	address	numunits	sizeperunits	monthlyrental	created_at	updated_at
<a href="#"></a> <a href="#"></a> <a href="#"></a>	100040	2020100	Kuala Lumpur	4	302 m2	RM200	2020-05-03 10:24:19	2020-05-03 10:24:19
<a href="#"></a> <a href="#"></a> <a href="#"></a>	100041	2020100	Penang	8	280 m2	RM155	2020-05-03 10:24:44	2020-05-03 10:24:44
<a href="#"></a> <a href="#"></a> <a href="#"></a>	100042	2020100	Kuantan	3	400 m2	RM145	2020-05-03 10:25:13	2020-05-03 10:25:13

The view residence in the housing officer page

STAFF ID (IN CHARGE)	RESIDENCE ID	ADDRESS	NUMBER OF UNITS	SIZE PER UNIT (METER SQUARE)	MONTHLY RENTAL (RM/UNIT)	EDIT
2020100	100040	Kuala Lumpur	4	302 m2	RM200	<button>EDIT</button>
2020100	100041	Penang	8	280 m2	RM155	<button>EDIT</button>
2020100	100042	Kuantan	3	400 m2	RM145	<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.

```
<section('content')
<div class="container">
    <div class="card-header">
        <h4 class="card-title">Edit Housing Officer</h4>
    </div>
    <div class="jumbotron">
        <div class="container">
            <div class="row">
                &foreach($housingofficers as $user)
                <div class="col">
                    <form action="/updatehousingofficer({$user->id})" method="POST" enctype="multipart/form-data">
                        {{csrf_field()}}
                        {{method_field('PUT')}}
                    <table class="table table-hover">
                        <tr>
                            <th><label> Housing Officer ID</label></th>
                            <td><input type="text" class="form-control" name="id" value="{{($user->id)}}" readonly></td>
                        </tr>
                        <tr>
                            <th><label> Full Name </label></th>
                            <td><input type="text" class="form-control" name="fullname" value="{{($user->fullname)}}" required></td>
                        </tr>
                        <tr>
                            <th><label> Username </label></th>
                            <td><input type="text" class="form-control" name="username" value="{{($user->username)}}" required></td>
                        </tr>
                        <tr>
                            <th><label> Email </label></th>
                            <td><input type="email" class="form-control" name="email" value="{{($user->email)}}" required></td>
                        </tr>
                    </table>
                    <br><br>
                    <button type="submit" name="submit" class="btn btn-primary"> Update Data </button>
                </div>
            </div>
        &endforeach
    </div>
</div>
```

## This is display form edit residence

**MICROHOUSING SYSTEM**

**Edit Residence**

**Address**  
Kuala Lumpur

**Num of Units**  
4

**Size of Units**  
302 m<sup>2</sup>

**Monthly Rental**  
RM200

**Update Data**

## 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.

```
<div class="container-fluid">
    <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>ALLOCATE</th>
                            <th>EDIT</th>
                            <th>REJECT</th>
                        </thead>
                        <tbody>
                            @foreach ($applications as $app)
                                @foreach ($residencies as $r)
                                    @if($app->residenceID==$r->residenceID)
                                        @if($r->admin_id==Auth::user()->id)
                                            <tr>
                                                <td>{{ $app->userID}}</td>
                                                <td>{{ $app->applicationID}}</td>
                                                <td>{{ $app->residenceID}}</td>
                                                <td>{{ $app->applicationDate}}</td>
                                                <td>{{ $app->requiredMonth}}</td>
                                                <td>{{ $app->requiredYear}}</td>
                                                <td>{{ $app->status}}</td>
                                                <td><a href="/allocatehousing[{$app->applicationID}]" class="btn btn-success">ALLOCATE</a></td>
                                                <td><a href="/editapplication[{$app->applicationID}]" class="btn btn-warning">EDIT</a></td>
                                                <td><a href="/deleteapplication[{$app->applicationID}]" class="btn btn-danger">REJECT</a></td>
                                            </tr>
                                        @endif
                                    @endif
                                @endforeach
                            @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
```

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

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

This display for view applications from input by applicant

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

APPLICANT ID	APPLICATION ID	RESIDENCE ID	APPLICATION DATE	REQUIRED MONTH	REQUIRED YEAR	STATUS	ALLOCATE	EDIT	REJECT
2020101	1750101	100041	2020-05-03 18:31:01	June	2020	Processing	ALLOCATE	EDIT	REJECT
2020101	1750102	100042	2020-05-03 18:31:20	July	2020	Processing	ALLOCATE	EDIT	REJECT
2020101	1750103	100040	2020-05-03 18:33:16	May	2020	Processing	ALLOCATE	EDIT	REJECT

## 8. Edit Applications

In this test will editapplication.blade.php function runs properly. This function useful for editing the applications, 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 Application model

```
<?php  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Application extends Model{  
    protected $table='application';  
    protected $primaryKey = 'applicationID';  
    protected $fillable=['applicantID','residenceID','applicationDate','requiredMonth','requiredYear'];  
}
```

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

```
public function editapplication($applicationID){  
    $applicationss = Application::where('applicationID',$applicationID)->get();  
    return view('Housingofficer.editapplication')->with('application', $applicationss);  
}
```

This is the editapplication.blade.php is form for the edit application

```
<div class="container">
    <div class="jumbotron">
        Edit Application
    </div>
    <div id="residence" class="row p-5">
        <div class="container">
            <div class="row">
                @foreach($application as $r)
                    <div class="col">
                        <form action="/updateapplication({$r->applicationID})" method="POST" enctype="multipart/form-data">
                            {{csrf_field()}}
                            {{method_field('PUT')}}
                            <div class="form-group">
                                <label> User ID </label>
                                <td><input type="text" name="userID" value="{{ Auth::user()->id }}></td>
                            </div>

                            <div class="form-group">
                                <label> Residence ID </label>
                                <td><input type="text" name="residenceID" value="{{ $r->residenceID }}></td>
                            </div>

                            <div class="form-group">
                                <label> Required Month </label>
                                <td><input type="text" name="requiredMonth" required="required" placeholder="E.g. May, June..."></td>
                            </div>

                            <div class="form-group">
                                <label> Required Year </label>
                                <td><input type="text" name="requiredYear" required="required" placeholder="E.g. 2020"></td>
                            </div>
                            <br><br>
                            <button type="submit" name="submit" class="btn btn-primary"> Update Application </button>
                        </form>
                    </div>
                @endforeach
            </div>
        </div>
    </div>
```

This is display form edit applications

The screenshot shows the MicroHousing System application interface. On the left is a sidebar with a purple gradient background containing navigation links: DASHBOARD, SET UP NEW RESIDENCE, RESIDENCES, APPLICATIONS, ALLOCATIONS, APPLICANTS, and HOUSING OFFICERS. The main content area has a light gray background. At the top, it says "Edit Application". Below that is a form with four input fields: "User ID" (2020100), "Residence ID" (100041), "Required Month" (E.g. May, June...), and "Required Year" (E.g. 2020). At the bottom of the form is a blue "Update Application" button. In the top right corner of the main window, there is a user profile placeholder.

## 9. Delete Applications

This test dashboardcontroller, will be able to delete the application which is already input by the applicant

```
public function deleteapplication($applicationID){  
    $applicationss = Application::find($applicationID);  
    $applicationss->delete();  
    return redirect ('/viewapplications')->with('application',$applicationss);  
}
```

## 10. Allocate Housing

In this test will allocatethousing.blade.php function runs properly. This function useful for allocating the residence for every unit in the residences, which is already input. For the allocate we need ‘Model’ that will receive data input from the allocation model and then update the data into the database.

This is the Allocation model

```
<?php  
  
namespace App;  
  
use Illuminate\Database\Eloquent\Model;  
use Carbon\Carbon;  
  
class Allocation extends Model{  
    protected $table='allocation';  
    protected $fillable=['unitNo','fromDate','endDate','duration'];  
}
```

This is the allocationhousing.blade.php is display all the unit which one wants to allocate for every application

```
@extends ('layouts.master')  
@section('title')  
    Allocation  
@endsection  
@section('content')  
    <div class="container">  
        <div class="jumbotron">  
            <h1> Unit Allocation </h1><br>  
            <table class="table table-bordered table-hover">  
                <thead>  
                    <tr>  
                        <th>Application ID</th>  
                        <th>Unit Number</th>  
                        <th>Allocate</th>  
                    </tr>  
                </thead>  
                @foreach ($applicationss as $app)  
                    @foreach ($units as $u)  
                        <tr>  
                            <td><input type="text" name="applicationID" class="form-control" value="{{ $app->applicationID }}" readonly></td>  
                            <td><input type="text" name="unitNo" class="form-control" value="{{ $u->unitNo }}" readonly></td>  
                            <td><a class="btn btn-primary btn-sm" href="/confirmallocate/{{ $app->applicationID }}/{{ $u->unitNo }}"/>Allocate</a></td>  
                        </tr>  
                    @endforeach  
                @endforeach  
                </table>  
                <br><br>  
            </div>  
        </div>  
    @endsection  
    @section('scripts')  
    @endsection
```

This function to call the application and unit in the database for allocating house

```
public function allocatehousing($id){  
    $applicationss = Application::where('applicationID',$id)->get();  
    $units = Unit::where('residenceID', $applicationss[0]->residenceID)->get();  
    return view('Housingofficer.allocatehousing')->with('units',$units)->with('applicationss',$applicationss);  
}
```

This is display for every unit per applications to allocate

Application ID	Unit Number	Allocate
1750101	t204	<button>Allocate</button>
1750101	t205	<button>Allocate</button>
1750101	t206	<button>Allocate</button>
1750101	t207	<button>Allocate</button>
1750101	t208	<button>Allocate</button>
1750101	t209	<button>Allocate</button>
1750101	t210	<button>Allocate</button>
1750101	t211	<button>Allocate</button>

This function in confirmallocate.blade.php to display the form for allocate

```
<div class="container">  
    <div class="jumbotron">  
        <h1>Confirmation </h1><br>  
        @foreach ($applications as $app)  
            @foreach ($units as $u)  
                <form action="/allocate" method="POST" enctype="multipart/form-data">  
                    {{csrf_field()}}  
                    <table class="table table-bordered table-hover">  
                        <tr>  
                            <th>Application ID</th>  
                            <td><input type="text" name="applicationID" class="form-control" value="{{ $app->applicationID }}" readonly></td>  
                        </tr>  
                        <tr>  
                            <th>Unit Number</th>  
                            <td><input type="text" name="unitNo" class="form-control" value="{{ $u->unitNo }}" readonly></td>  
                        </tr>  
                        <tr>  
                            <th>From Date</th>  
                            <td><input type="date" name="fromDate" class="date form-control"></td>  
                        </tr>  
                        <tr>  
                            <th>Duration (Days)</th>  
                            <td><input type="text" name="duration" class="form-control" value="365" readonly></td>  
                        </tr>  
                        <tr>  
                            <th>Change Acceptance Status</th>  
                            <td><input type="text" name="status" class="form-control" value="{{ $app->status }}></td>  
                        </tr>  
                        <tr>  
                            <th>Change Payment Status</th>  
                            <td><input type="text" name="payment_status" class="form-control" value="{{ $app->payment_status }}></td>  
                        </tr>  
                    </table>  
                    <center>  
                        <table table-bordered table-hover>  
                            <tr>  
                                <td>Are you sure you want to allocate?</td>  
                            </tr>  
                        </table>  
                    </center>  
                </form>  
            @endforeach  
        @endforeach  
    </div>
```

This is the controller for confirmallocate to get unitNo and applicationID in the applications and unit database.

```
public function confirmallocate($applicationID,$unitNo){  
    $applicationss = Application::where('applicationID',$applicationID)->get();  
    $units = Unit::where('unitNo', $unitNo)->get();  
    return view('Housingofficer.confirmallocate')->with('units',$units)->with('applicationss',$applicationss);
```

```

public function storeallocate(Request $request){
    $allocation = new Allocation();
    $allocation->applicationID = $request->input('applicationID');
    $allocation->unitNo = $request->input('unitNo');
    $allocation->fromDate = new Carbon($request->input('fromDate'));
    $toAdd = new Carbon($request->input('toDate'));
    $allocation->duration=$request->input('duration');
    $duration = $allocation->duration-$request->input('duration');
    $allocation->endDate = $toAdd->addDays($duration);
    $allocation->save();
    //application changing status
    $applicationID = $allocation->applicationID = $request->input('applicationID');
    $application = Application::find($applicationID);
    $application->status = $allocation->status = $request->input('status');
    $application->payment_status = $allocation->payment_status = $request->input('payment_status');
    $application->save();
    return view('Housingofficer.dashboard');
}

```

This is display for form allocate every applications

The screenshot shows a confirmation dialog box titled "Confirmation". It contains the following fields:

Application ID	1750101
Unit Number	1204
From Date	dd/mm/yyyy
Duration (Days)	365
Change Acceptance Status	Processing
Change Payment Status	Unavailable

Below the fields, a message asks: "Are you sure you want to allocate?". There are two buttons: "Yes" (blue) and "No" (red).

If you choose “No” button the system will back to the view applications and for the “Yes” button the system will save to allocation in the database.

## 11. Submit Application

This test application controller, will be able to submit application and each variable will input to ‘microhousing’ database in application table.

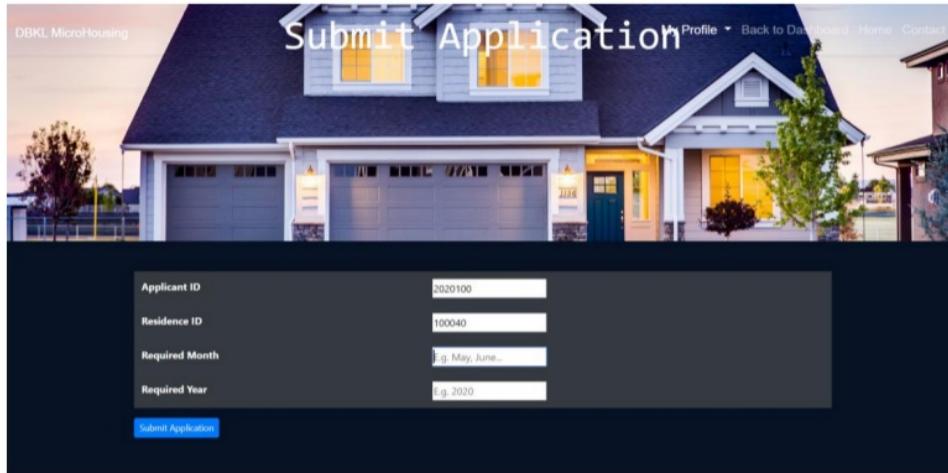
```
public function SubmitApplication($id)
{
    $residence = DB::table('residences')->where('residenceID',$id)->get();
    return view('Applicant.SubmitApplication',[ 'residences' => $residence]);
}

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

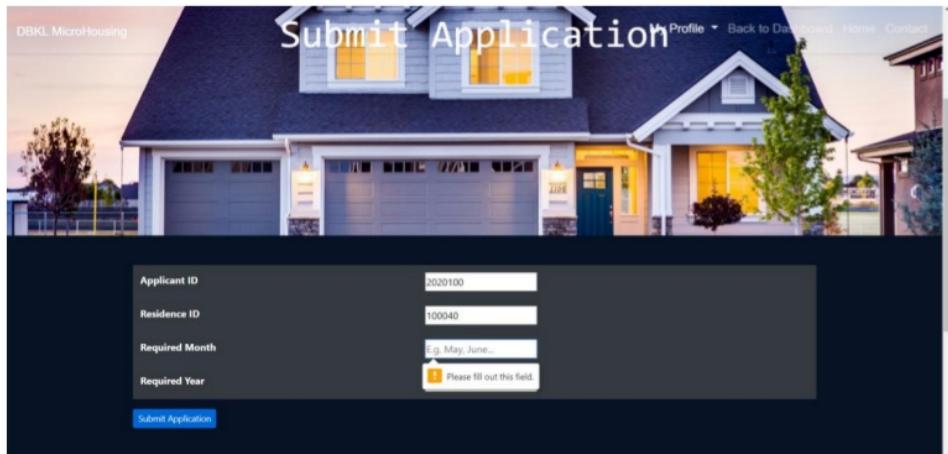
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Submit</title>
        <link rel="stylesheet" type="text/css" href="{{ asset('assets/css/bootstrap.css') }}>
        <link rel="stylesheet" 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">
                            My Profile
                            <span class="caret"></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>
</html>
```

This is display submit application page



The screenshot shows the 'Submit Application' page for DBKL MicroHousing. At the top, there's a banner featuring a blue house with a garage and a yellow door. Below the banner, the title 'Submit Application' is centered. On the right side of the header, there are links for 'Profile', 'Back to Dashboard', 'Home', and 'Contact'. The main form area has four input fields: 'Applicant ID' (2020100), 'Residence ID' (100040), 'Required Month' (E.g. May, June...), and 'Required Year' (E.g. 2020). A blue 'Submit Application' button is located at the bottom left of the form.

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



This screenshot shows the same 'Submit Application' page as above, but with invalid input in the 'Required Month' field. The 'Required Month' field contains the placeholder text 'E.g. May, June...' and has a red border, indicating it is required. A red error message 'Please fill out this field.' is displayed below the field. All other fields ('Applicant ID', 'Residence ID', and 'Required Year') are filled correctly with their respective values (2020100, 100040, E.g. 2020).

## 12. 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

```
<div class="container-fluid contents">
    <div id="home" class="row clearfix" data-ibg-bg="{{ asset('assets/img/15.jpg') }}>
        <div class="title-home">
            Residences
        </div>
    </div>
    <div id="residence" class="row p-5">
        <div class="container">
            <div class="row">
                <div class="col">
                    <table class="table table-borderless table-dark table-hover">
                        <tr>
                            <th>Residence ID</th>
                            <th>Address</th>
                            <th>Number of Units</th>
                            <th>Size Per Unit (Meter Square)</th>
                            <th>Monthly Rental (Malaysian Ringgit)</th>
                            <th></th>
                        </tr>
                        @foreach($residencies as $r)
                        <tr>
                            <td>{{ $r->residenceID}}</td>
                            <td>{{ $r->address}}</td>
                            <td>{{ $r->nnumunits}}</td>
                            <td>{{ $r->sizeperunit}}</td>
                            <td>{{ $r->monthlyrental}}</td>
                            <td>
                                <a class="btn btn-primary btn-sm" href="/SubmitApplication/{{ $r->residenceID}}">Submit Application</a>
                            </td>
                        </tr>
                    @endforeach
                </div>
            </div>
        </div>
    </div>

```

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

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

This is display view residences

Residence ID	Address	Number of Units	Size Per Unit (Meter Square)	Monthly Rental (Malaysian Ringgit)	
100040	Kuala Lumpur	4	302 m2	RM200	<a href="#">Submit Application</a>
100041	Penang	8	280 m2	RM155	<a href="#">Submit Application</a>
100042	Kuantan	3	400 m2	RM145	<a href="#">Submit Application</a>

### 13. 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">
          <thead>
            <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 (Ringgit)</th>
              <th>Payment Status</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              &foreach($application as $app)
                &if($app->userID==Auth::user()->id)
              <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
        </tbody>
      </table>
    </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 = Application::all();
  $allocate = Allocation::all();
  return view('Applicant.dashboard_user')->with('application',$application)->with('allocate',$allocate);
}
```

This display the application which is already input

The screenshot shows a web application for DBKL MicroHousing. At the top, there's a navigation bar with links for 'My Profile', 'Home', 'Application', 'Residence', and 'Contact'. The main header features the text 'Hello Wulandari Maharaní!' over a background image of colorful buildings. Below the header, a large button labeled 'Application' is visible. A table below the button displays three rows of application data:

Application ID	Residence ID	Application Date	Required Month	Required Year	Status	Payment (Ringgit)	Payment Status
1750101	100041	2020-05-03 18:31:01	June	2020	Processing	RM155	Unavailable
1750102	100042	2020-05-03 18:31:20	July	2020	Processing	RM145	Unavailable
1750103	100040	2020-05-03 18:33:16	May	2020	Processing	RM200	Unavailable

#### 14. Change Password

In this test, reset.blade.php, which is the folder function, runs properly, to reset or change the password.

```
<div class="card-body">
  <form method="POST" action="{{ route('password.update') }}"
    @csrf

    <input type="hidden" name="token" value="{{ $token }}>

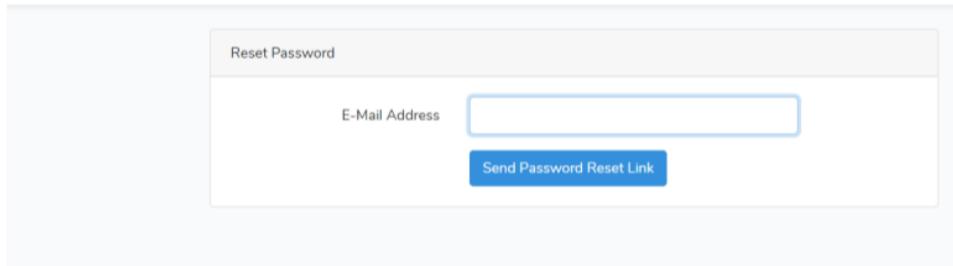
    <div class="form-group row">
      <label for="email" class="col-md-4 col-form-label text-md-right">{{ __('E-Mail Address') }}</label>
      <div class="col-md-6">
        <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email" value="{{ $email }}>
        @error('email')
          <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" value="">
        @error('password')
          <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
          </span>
        @enderror
      </div>
    </div>

    <div class="form-group row">
      <label for="password-confirm" class="col-md-4 col-form-label text-md-right">{{ __('Confirm Password') }}</label>
      <div class="col-md-6">
        <input id="password-confirm" type="password" class="form-control" name="password_confirmation">
      </div>
    </div>
  </form>
</div>
```

This is display for change password

DBKL Microhousing



## 15. View Allocation

In this test, viewallocation.blade.php which is the folder function runs properly, displaying all the allocation which is input by the applicant and all the allocation data in the database.

The form for view allocation

```
@section('title')
    View Allocations
@endsection
@section('content')


#### View Allocations



| APPLICATION ID            | UNIT NO            | FROM DATE            | DURATION             | END DATE            | DELETE                                                                           |
|---------------------------|--------------------|----------------------|----------------------|---------------------|----------------------------------------------------------------------------------|
| {{ \$al->applicationID }} | {{ \$al->unitNo }} | {{ \$al->fromDate }} | {{ \$al->duration }} | {{ \$al->endDate }} | <a class="btn btn-danger" href="/deleteallocation({\$al-&gt;unitNo})">DELETE</a> |


```

The function for call the allocation in the database

```
public function viewallocation(){
    $allocation = Allocation::all();
    return view('Housingofficer.viewallocation')->with('allocation',$allocation);
}
```

This is display for view allocation

APPLICATION ID	UNIT NO	FROM DATE	DURATION	END DATE	DELETE
1750101	1204	2020-05-04 00:00:00	365	2021-05-04 00:00:00	<button>DELETE</button>

## 16. Delete Allocation

This test dashboardcontroller, will be able to delete the allocation which is already input by the housing officer.

```
public function deleteallocate($applicationID){  
    $allocation = Allocation::where('unitNo',$applicationID)->delete();|  
    return redirect ('/viewallocation')->with('allocation',$allocation);  
}
```

## 17. Registered Applicant

This test, viewapplicant.blade.php function runs properly, the applicant are displayed in Housing Officer menu.This is the form for the view applicant

```
@section('title')  
    View Applicant  
@endsection  
@section('content')  
<div class="container-fluid">  
    <div class="row">  
        <div class="col-md-12">  
            <div class="card striped-tabled-with-hover">  
                <div class="card-header ">  
                    <h4 class="card-title">Registered Applicants</h4>  
                </div>  
                <div class="card-body table-full-width table-responsive">  
                    <table class="table table-hover table-striped">  
                        <thead>  
                            <th>Applicant ID</th>  
                            <th>Full Name</th>  
                            <th>Username</th>  
                            <th>Email</th>  
                            <th>Monthly Income</th>  
                            <th>EDIT</th>  
                            <th>DELETE</th>  
                        </thead>  
                        <tbody>  
                            @foreach ($applicants as $user)  
                                <tr>  
                                    <td>{{ $user->id}}</td>  
                                    <td>{{ $user->fullname}}</td>  
                                    <td>{{ $user->username}}</td>  
                                    <td>{{ $user->email}}</td>  
                                    <td>{{ $user->monthlyIncome}}</td>  
                                    <td><a href="/editapplicant({{$user->id}})" class="btn btn-warning">EDIT</a></td>  
                                    <td><a href="/deleteapplicant({{$user->id}})" class="btn btn-danger">DELETE</a></td>  
                                </tr>  
                            @endforeach  
                        </tbody>  
                    </table>  
                </div>  
            </div>  
        </div>  
    </div>  
</div>
```

This function is to call the table applicant in the database, which is register by the applicant and will display in the Housing Officer Menu

```
public function viewapplicant(){
    $filter = 'Applicant';
    $applicants = User::where('usertype',$filter)->get();
    return view('Housingofficer.viewapplicant')->with('applicants',$applicants);
}
```

This is display for the registered applicant

APPLICANT ID	FULL NAME	USERNAME	EMAIL	MONTHLY INCOME	EDIT	DELETE
2020101	Wulan Diani Maharani	wulan	wulan@gmail.com	10000	<button>EDIT</button>	<button>DELETE</button>

## 18. Edit Applicant

In this test will editapplicant.blade.php function runs properly. This function useful for editing the applicant which is already input and then update the data into the database.

```
@section('title')
    Edit Applicant
@endsection
@section('content')


<div class="jumbotron">
        Edit Applicant
    </div>
</div>


<div class="row">
        @foreach($applicants as $user)
            <div class="col">
                <form action="/updateapplicant({$user->id})" method="POST" enctype="multipart/form-data">
                    {{csrf_field()}}
                    {{method_field('PUT')}}
                    <div class="form-group">
                        <label> Applicant ID </label>
                        <td><input type="text" name="id" value="{{$user->id}}" readonly></td>
                    </div>

                    <div class="form-group">
                        <label> Full Name </label>
                        <td><input type="text" name="fullname" value="{{$user->fullname}}" required></td>
                    </div>

                    <div class="form-group">
                        <label> Username </label>
                        <td><input type="text" name="username" value="{{$user->username}}" required></td>
                    </div>

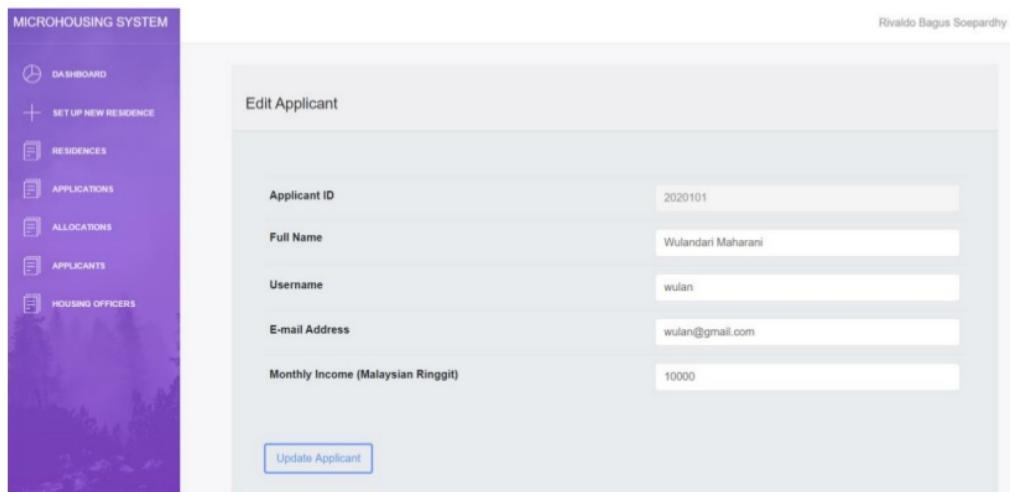
                    <div class="form-group">
                        <label> Monthly Income (Malaysian Ringgit) </label>
                        <td><input type="text" name="monthlyIncome" value="{{$user->monthlyIncome}}" required></td>
                    </div>
                    <br><br>
                    <button type="submit" name="submit" class="btn btn-primary"> Update Application </button>
                </form>
            </div>
        @endforeach
    </div>


```

This function is to call the table applicant in the database, which is want to edit

```
public function editapplicant($id){  
    $applicants = User::where('id',$id)->get();  
    return view('Housingofficer.editapplicant')->with('applicants', $applicants);  
}  
  
public function updateapplicant(Request $request,$id){  
    $applicants = User::find($id);  
    $applicants->fullname = $request->input('fullname');  
    $applicants->username = $request->input('username');  
    $applicants->monthlyIncome = $request->input('monthlyIncome');  
    $applicants->email = $request->input('email');  
    $applicants->save();  
    return redirect('/viewapplicant')->with('applicants', $applicants);  
}
```

The display for editing the applicant



The screenshot shows the MicroHousing System application interface. On the left, there is a sidebar with a purple background containing the following menu items:

- DASHBOARD
- SET UP NEW RESIDENCE
- RESIDENCES
- APPLICATIONS
- ALLOCATIONS
- APPLICANTS
- HOUSING OFFICERS

The main content area has a light gray background. At the top, it says "Edit Applicant". Below that, there are several input fields:

Applicant ID	2020101
Full Name	Wulandari Maharani
Username	wulan
E-mail Address	wulan@gmail.com
Monthly Income (Malaysian Ringgit)	10000

At the bottom of the form is a blue rectangular button labeled "Update Applicant".

## 19. Delete Applicant

This test dashboardcontroller, will be able to delete the application which is already input by the applicant in the housing officer menu.

```
public function deleteapplicant($id){  
    $applicants = User::find($id);  
    $applicants->delete();  
    return redirect ('/viewapplicant')->with('applicants', $applicants);  
}
```

## 20. Registered Housing Officer

In this test, viewhousingofficer.blade.php function runs properly, the housing officer are displayed in the Housing Officer menu. This is the form for the view Housing Officer

```
@extends ('layouts.master')
@section('title')
    View Admins
@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">Registered HousingOfficer</h4>
                </div>
                <div class="card-body table-full-width table-responsive">
                    <table class="table table-hover table-striped">
                        <thead>
                            <th>Housing Officer ID</th>
                            <th>Full Name</th>
                            <th>Username</th>
                            <th>Email</th>
                        </thead>
                        <tbody>
                            @foreach ($housingofficers as $user)
                            <tr>
                                <td>{{ $user->id}}</td>
                                <td>{{ $user->fullname}}</td>
                                <td>{{ $user->username}}</td>
                                <td>{{ $user->email}}</td>
                            </tr>
                            @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
@endsection
@section('scripts')


```

This function is to call the table housing officer in the database, which is register by the housing and will display in the Housing Officer Menu

```
public function viewhousingofficer(){
    $filter = 'Housingofficer';
    $housingofficers = User::where('usertype',$filter)->get();
    return view('Housingofficer.viewhousingofficer')->with('housingofficers',$housingofficers);
}
public function editapplicant($id){
```

This display for view housing officer in the housing officer menu

The screenshot shows a user interface for a housing system. On the left is a purple sidebar menu titled "MICROHOUSING SYSTEM" with the following items:

- DASHBOARD
- SET UP NEW RESIDENCE
- RESIDENCES
- APPLICATIONS
- ALLOCATIONS
- APPLICANTS
- HOUSING OFFICERS

The main content area has a header "Registered HousingOfficer" and a table with the following data:

HOUSING OFFICER ID	FULL NAME	USERNAME	EMAIL
2020100	Rivaldo Bagus Soepardhy	aidobagus	aidobagus@hotmail.co.id

In the top right corner of the main area, there is a small profile picture and the text "Rivaldo Bagus Soepardhy".

## 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
<b>'LOGIN APPLICANT' completed successfully</b>	

## 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 <b>OK</b>	
12. type on id=password with value 12345678 <b>OK</b>	
13. type on id=password-confirm with value 12345678 <b>OK</b>	
14. click on css=.btn <b>OK</b>	
15. click on id=navbarDropdown <b>OK</b>	
16. click on linkText=Logout <b>OK</b>	
<b>'REGISTER APPLICANT' completed successfully</b>	

### 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
<b>Click the drop-down arrow to toggle column's visibility.</b>						
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

## 7. Edit Application – HousingOfficer

```
def test_editApplication(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1144, 760)
    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, "li:nth-child(4) > .nav-link").click()
    self.driver.find_element(By.LINK_TEXT, "EDIT").click()
    self.driver.find_element(By.NAME, "requiredMonth").click()
    self.driver.find_element(By.NAME, "requiredMonth").send_keys("July")
    self.driver.find_element(By.NAME, "requiredYear").click()
    self.driver.find_element(By.NAME, "requiredYear").send_keys("2021")
    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()
```

7. click on css=.btn-primary OK

8. click on css=li:nth-child(4) > .nav-link OK

9. click on linkText=EDIT OK

10. click on name=requiredMonth OK

11. type on name=requiredMonth with value July OK

12. click on name=requiredYear OK

13. type on name=requiredYear with value 2021 OK

14. click on name=submit OK

15. click on id=navbarDropdown OK

16. click on linkText=Logout OK

'Edit Application' completed successfully balphokd/index.html#

8. Delete Application – HousingOfficer

```
def test_deleteApplication(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1144, 760)
    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, "li:nth-child(4) > .nav-link")
    self.driver.find_element(By.CSS_SELECTOR, "tr:nth-child(2) .btn-danger")
    self.driver.find_element(By.ID, "navbarDropdown").click()
    self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

LOG EXECUTION  
3. click on linkText=Login OK  
4. click on id=username OK  
5. type on id=username with value aldobagus OK  
6. click on id=password OK  
7. type on id=password with value 12345678 OK  
8. click on css=.btn-primary OK  
9. click on css=li:nth-child(4) > .nav-link OK  
10. click on css=tr:nth-child(2) .btn-danger OK  
11. click on id=navbarDropdown OK  
12. click on linkText=Logout OK

**'Delete Application' completed successfully**

9. Allocate Housing & View Allocation – HousingOfficer

```
def test_allocateHousing(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1146, 762)
    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, "li:nth-child(4) > .nav-link").cli
    self.driver.find_element(By.LINK_TEXT, "ALLOCATE").click()
    self.driver.find_element(By.LINK_TEXT, "Allocate").click()
    self.driver.find_element(By.NAME, "fromDate").click()
    self.driver.find_element(By.NAME, "fromDate").send_keys("2020-05-20")
    self.driver.find_element(By.NAME, "status").click()
    self.driver.find_element(By.NAME, "status").send_keys("Approved")
    self.driver.find_element(By.NAME, "payment_status").click()
    self.driver.find_element(By.NAME, "payment_status").click()
    self.driver.find_element(By.NAME, "payment_status").send_keys("Unpaid")
    self.driver.find_element(By.NAME, "submit").click()
    self.driver.find_element(By.CSS_SELECTOR, "li:nth-child(5) > .nav-link").cli
    self.driver.find_element(By.ID, "navbarDropdown").click()
    self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

Log Reference

15. type on name=date with value 2020-05-20 OK

14. click on name=status OK

15. type on name=status with value Approved OK

16. click on name=payment\_status OK

17. click on name=payment\_status OK

18. type on name=payment\_status with value Unpaid OK

19. click on name=submit OK

20. click on css=li:nth-child(5) > .nav-link OK

21. click on id=navbarDropdown OK

22. click on linkText=Logout OK

**'Allocate Housing' completed successfully**

#### 10. Delete Allocation – HousingOfficer

```
def test_deleteAllocation(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1149, 764)
    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, "li:nth-child(5) p").click()
    self.driver.find_element(By.LINK_TEXT, "DELETE").click()
    self.driver.find_element(By.ID, "navbarDropdown").click()
    self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

Log	Reference
3. click on link=Login OK	
4. click on id=username OK	
5. type on id=username with value aldobagus OK	
6. click on id=password OK	
7. type on id=password with value 12345678 OK	
8. click on css=.btn-primary OK	
9. click on css=li:nth-child(5) p OK	
10. click on linkText=DELETE OK	
11. click on id=navbarDropdown OK	
12. click on linkText=Logout OK	

**'Delete Allocation' completed successfully**

## 11. View & Edit Applicant – HousingOfficer

```
class TestViewEditApplicant():
    def setup_method(self, method):
        self.driver = webdriver.Chrome()
        self.vars = {}

    def teardown_method(self, method):
        self.driver.quit()

    def test_viewEditApplicant(self):
        self.driver.get("http://localhost:8000/")
        self.driver.set_window_size(1153, 767)
        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.ID, "password").send_keys(Keys.ENTER)
        self.driver.find_element(By.CSS_SELECTOR, "li:nth-child(6) > .nav-link").cli
        self.driver.find_element(By.CSS_SELECTOR, "tr:nth-child(2) .btn-warning").cl
        self.driver.find_element(By.NAME, "fullname").click()
        self.driver.find_element(By.NAME, "fullname").send_keys("I Putu Senna Hakkin
        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

8. sendKeys on id=password with value `KEY_ENTER` **OK**
9. click on css=li:nth-child(6) > .nav-link **OK**
10. click on css=tr:nth-child(2) .btn-warning **OK**
11. click on name=fullname **OK**
12. type on name=fullname with value I Putu Senna Hakkinanda **OK**
13. click on name=submit **OK**
14. click on id=navbarDropdown **OK**
15. click on linkText=Logout **OK**

**'View & Edit Applicant' completed successfully**

## 12. Delete Applicant – HousingOfficer

```
def test_deleteApplicant(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1157, 768)
    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, "li:nth-child(6) > .nav-link").cli
    self.driver.find_element(By.CSS_SELECTOR, "tr:nth-child(2) .btn-danger").cli
    self.driver.find_element(By.ID, "navbarDropdown").click()
    self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

### Log

### Reference

5. type on id=username with value aldobagus **OK**
6. click on id=password **OK**
7. type on id=password with value 12345678 **OK**
8. click on css=.btn-primary **OK**
9. click on css=li:nth-child(6) > .nav-link **OK**
10. click on css=tr:nth-child(2) .btn-danger **OK**
11. click on id=navbarDropdown **OK**
12. click on linkText=Logout **OK**

**'Delete Applicant' completed successfully**

### 13. View Housing Officer – HousingOfficer

```
def test_viewHO(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1160, 770)
    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, "li:nth-child(7) p").click()
    self.driver.find_element(By.CSS_SELECTOR, "td:nth-child(3)").click()
    self.driver.find_element(By.ID, "navbarDropdown").click()
    self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

Log	Reference
5. type on id=username with value aldobagus	OK
6. click on id=password	OK
7. type on id=password with value 12345678	OK
8. click on css=.btn-primary	OK
9. click on css=li:nth-child(7) p	OK
10. click on css=td:nth-child(3)	OK
11. click on id=navbarDropdown	OK
12. click on linkText=Logout	OK
<b>'View HO' completed successfully</b>	

#### 14. View Allocation – Applicant

```
def test_viewAllocationUser(self):
    self.driver.get("http://localhost:8000/")
    self.driver.set_window_size(1144, 760)
    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.CSS_SELECTOR, ".row:nth-child(5) > .col").click()
    self.driver.find_element(By.CSS_SELECTOR, ".table:nth-child(2) tr").click()
    self.driver.find_element(By.CSS_SELECTOR, ".row:nth-child(5) > .col").click()
    self.driver.find_element(By.CSS_SELECTOR, ".row:nth-child(5) > .col").click()
    self.driver.find_element(By.CSS_SELECTOR, ".row:nth-child(5) > .col").click()
    self.driver.find_element(By.ID, "navbardrop").click()
    self.driver.find_element(By.LINK_TEXT, "Logout").click()
```

Log	Reference
8. click on css=.btn-primary	OK
9. click on css=.row:nth-child(5) > .col	OK
10. click on css=.table:nth-child(2) tr	OK
11. click on css=.row:nth-child(5) > .col	OK
12. click on css=.row:nth-child(5) > .col	OK
13. click on css=.row:nth-child(5) > .col	OK
14. click on id=navbardrop	OK
15. click on linkText=Logout	OK
<b>'View Allocation User' completed successfully</b>	

## 15. Change Password – Applicant / Housing Officer (Same System)

Note : Due to Selenium behavior, we were unable to test through the automated system, as the result, we have to test it manually. We use Mailtrap.io for the need of dummy test of change password



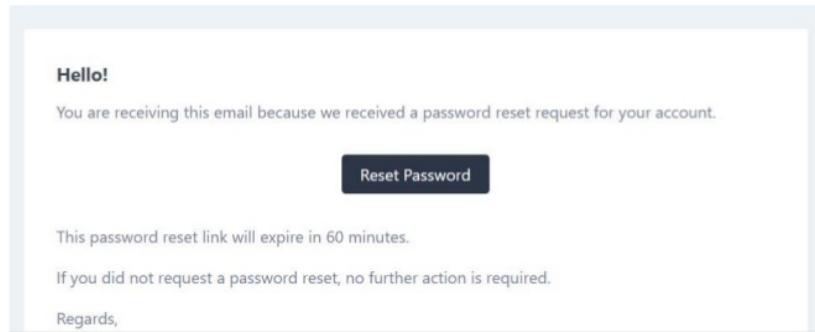
*Clicking Change Password*

A screenshot of a 'Reset Password' form. It has a header 'Reset Password'. Below it is an input field labeled 'E-Mail Address' containing the value 'wulan@wulan.com'. To the right of the input field is a blue button labeled 'Send Password Reset Link'.

*Enter your e-mail address linked to the current account.*

A screenshot of a 'Reset Password' form. It has a header 'Reset Password'. A green success message box contains the text 'We have emailed your password reset link!'. Below the message is an input field labeled 'E-Mail Address' with a placeholder 'wulan@wulan.com'. To the right of the input field is a blue button labeled 'Send Password Reset Link'.

*The system will send an e-mail containing link for password-reset*



*Here we use mailtrap.io for our dummy testing. The email has been received. Clicking the link.*

Reset Password

E-Mail Address

Password

Confirm Password

The link will redirect you to a form for changing your password. You just have to fill in the new one you have to use.

Reset Password

E-Mail Address

Password

Confirm Password

My old password is "12345678". I changed it to "master123"

Landing Page

Your password has been reset!

Click this link to go to [Applicant dashboard!](#)  
Click this link to go to [Housing Officer dashboard!](#)

Password has been changed. From here, I try to logout and go to login form.

Login

Username/Email

Password

Remember Me

[Forgot Your Password?](#)

I try to log in with my old password.

Login

Username/Email

Password

Remember Me

[Forgot Your Password?](#)

*However, the system rejected. This proves the password changes.*

Login

Username/Email

Password

Remember Me

[Forgot Your Password?](#)

*Now, I am trying to use my new password "master123"*



## System Testing

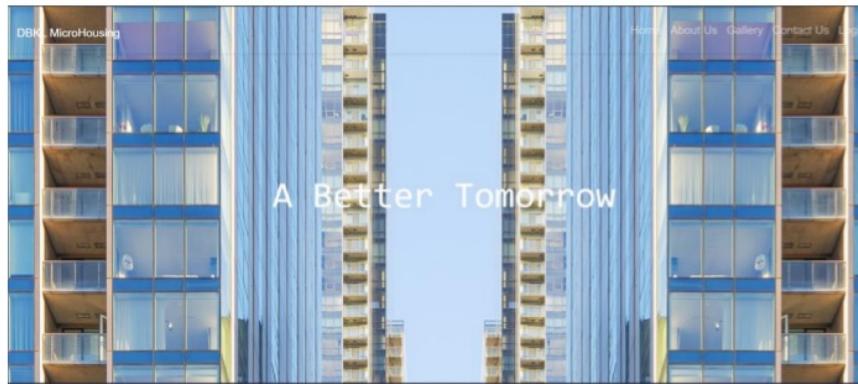
Functional Requirements (updated)

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"/>
Username	<input type="text"/>
E-Mail Address	<input type="text"/>
Monthly Income	<input type="text"/>
Password	<input type="text"/>
Confirm Password	<input type="text"/>
<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'. Below the header is a 'Register' form with the following fields:

- Name: An empty text input field.
- Username: An empty text input field.
- E-Mail Address: An empty text input field.
- Monthly Income: An empty text input field.
- Password: An empty text input field.
- Confirm Password: An empty text input field.

At the bottom of the form is a blue 'Register' button.

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

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'. Below the header is a 'Register' form with the following fields:

- Name: 'wulan' (empty)
- Username: 'wulan' (highlighted in red)
- E-Mail Address: 'Admin' (highlighted in blue)
- Monthly Income: '1000' (highlighted in yellow)
- Password: (empty)
- Confirm Password: (empty)

Validation messages are displayed next to the error fields:

- 'The username has already been taken.'
- 'Please include an '@' in the email address. Admin' is missing an '@'.'

At the bottom of the form is a blue 'Register' button.

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

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'. Below the header is a 'Register' form with the following fields:

- Name: 'wulan' (empty)
- Username: 'wulan' (highlighted in red)
- E-Mail Address: 'Admin' (highlighted in blue)
- Monthly Income: '1000' (highlighted in yellow)
- Password: (empty)
- Confirm Password: (empty)

Validation messages are displayed next to the error fields:

- 'The username has already been taken.'
- 'Please include an '@' in the email address. Admin' is missing an '@'.'

At the bottom of the form is a blue 'Register' button.

DBKL Microhousing

Login Register

Register

Name	<input type="text" value="iwulan"/>
Username	wulandari
E-Mail Address	wulandari@gmail.com
Monthly Income	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

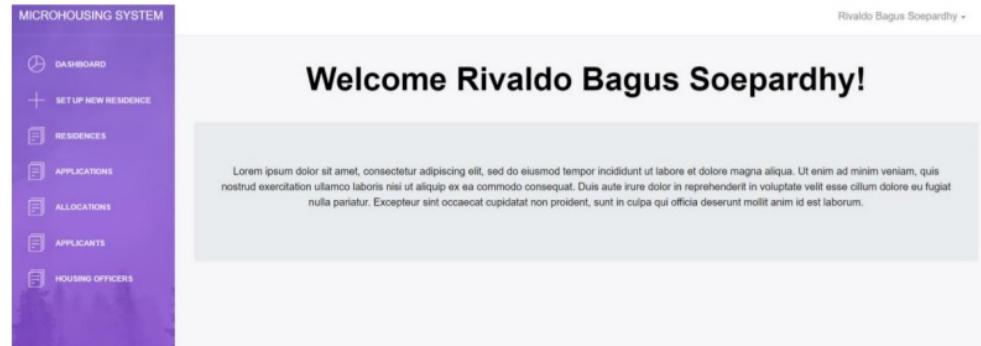
Login

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

Applicant Homepage



## Housing Officer Homepage

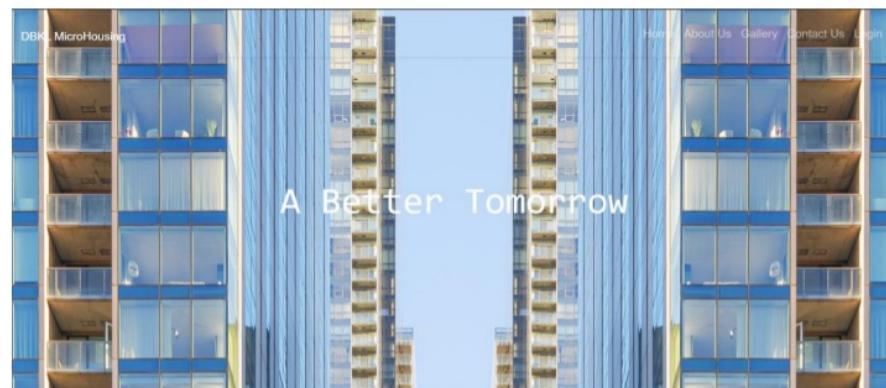


### 3. LogOut

After login there are logout button above the page



Back to the dashboard



#### 4. Change Password

Menu change password for user which will change the password

The screenshot shows a 'Reset Password' page. At the top, it says 'Reset Password'. Below that is an 'E-Mail Address' input field with a placeholder 'Enter the Email'. At the bottom is a blue 'Send Password Reset Link' button.

#### 5. Set Up New Residence

Set Up New Residence menu located in housingofficer page

The screenshot shows a 'Set Up New Residence' page. On the left is a sidebar with 'MICROHOUSING SYSTEM' and icons for Dashboard, Set Up New Residence (highlighted), Residences, Applications, Allocations, Applicants, and Housing Officers. The main area has a title 'Set Up New Residence'. It contains several input fields: 'Staff ID (in charge) for the residence' (2020100), 'Address' (placeholder 'Enter the Address'), 'Num of Units' (placeholder 'Enter the Number'), 'Size of Units (Meter Square)' (placeholder 'Enter the Size'), and 'Monthly Rental (Malaysian Ringgit)' (placeholder 'Enter the Price'). At the bottom is a blue 'Save Data' button.

#### 6. View Residence Detail

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

The screenshot shows a 'View Residences' page. On the left is a sidebar with 'MICROHOUSING SYSTEM' and icons for Dashboard, Set Up New Residence, Residences (highlighted), Applications, Allocations, Applicants, and Housing Officers. The main area has a title 'View Residences' and a table with the following data:

STAFF ID (IN CHARGE)	RESIDENCE ID	ADDRESS	NUMBER OF UNITS	SIZE PER UNIT (METER SQUARE)	MONTHLY RENTAL (RINGGIT)	EDIT
2020100	100040	Kuala Lumpur	4	302 m2	RM200	<button>EDIT</button>
2020100	100041	Penang	8	280 m2	RM155	<button>EDIT</button>
2020100	100042	Kuantan	3	400 m2	RM145	<button>EDIT</button>

## 7. Edit Residence Detail

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

The screenshot shows the 'Edit Residence' page of the Microhousing System. On the left is a sidebar with icons for Dashboard, Set Up New Residence, Residences, Applications, Allocations, Applicants, and Housing Officers. The main area has a header 'Edit Residence'. It contains four input fields: 'Address' (Kuala Lumpur), 'Num of Units' (4), 'Size of Units' (302 m2), and 'Monthly Rental' (RM200). At the bottom is a blue 'Update Data' button.

## 8. View Applications

To view the application which does the applicant already input.

APPLICANT ID	APPLICATION ID	RESIDENCE ID	APPLICATION DATE	REQUIRED MONTH	REQUIRED YEAR	STATUS	ALLOCATE	EDIT	REJECT
2020101	1750101	100041	2020-05-03 19:38:30	June	2020	Accepted	<span>ALLOCATE</span>	<span>EDIT</span>	<span>REJECT</span>
2020101	1750102	100042	2020-05-03 18:31:20	July	2020	Processing	<span>ALLOCATE</span>	<span>EDIT</span>	<span>REJECT</span>
2020101	1750103	100040	2020-05-03 18:33:16	May	2020	Processing	<span>ALLOCATE</span>	<span>EDIT</span>	<span>REJECT</span>

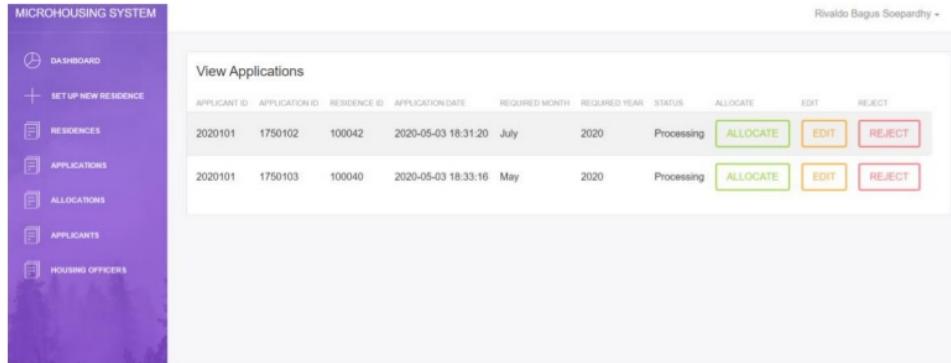
## 9. Edit Applications

To edit the applications and update the data in the database

The screenshot shows the 'Edit Application' page of the Microhousing System. On the left is a sidebar with icons for Dashboard, Set Up New Residence, Residences, Applications, Allocations, Applicants, and Housing Officers. The main area has a header 'Edit Application'. It contains four input fields: 'User ID' (2020100), 'Residence ID' (100041), 'Required Month' (E.g. May, June...), and 'Required Year' (E.g. 2020). At the bottom is a blue 'Update Application' button.

## 10. Delete Applications

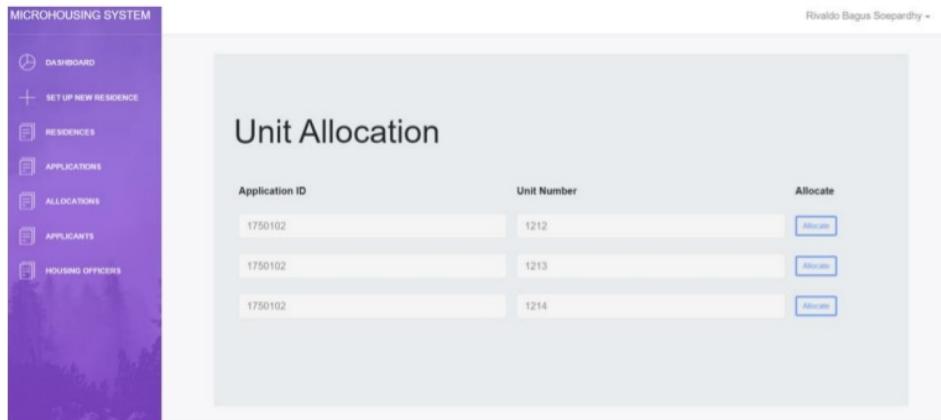
To delete the applications in the database



APPLICANT ID	APPLICATION ID	RESIDENCE ID	APPLICATION DATE	REQUIRED MONTH	REQUIRED YEAR	STATUS	ALLOCATE	EDIT	REJECT
2020101	1750102	100042	2020-05-03 18:31:20	July	2020	Processing	<span style="background-color: #90EE90; border: 1px solid #00AEEF; padding: 2px 10px; border-radius: 5px;">ALLOCATE</span>	<span style="background-color: #FFD966; border: 1px solid #E6C900; padding: 2px 10px; border-radius: 5px;">EDIT</span>	<span style="background-color: #FFB3B3; border: 1px solid #E63935; padding: 2px 10px; border-radius: 5px;">REJECT</span>
2020101	1750103	100040	2020-05-03 18:33:16	May	2020	Processing	<span style="background-color: #90EE90; border: 1px solid #00AEEF; padding: 2px 10px; border-radius: 5px;">ALLOCATE</span>	<span style="background-color: #FFD966; border: 1px solid #E6C900; padding: 2px 10px; border-radius: 5px;">EDIT</span>	<span style="background-color: #FFB3B3; border: 1px solid #E63935; padding: 2px 10px; border-radius: 5px;">REJECT</span>

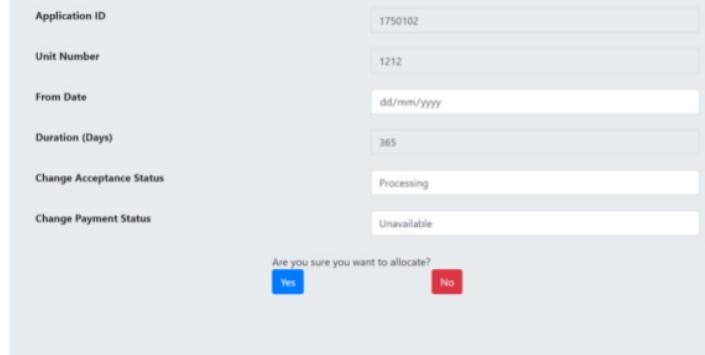
## 11. Allocate Housing

To allocate the housing for every unit in the residence to every applications



APPLICATION ID	UNIT NUMBER	ALLOCATE
1750102	1212	<span style="background-color: #00AEEF; border: 1px solid #00AEEF; padding: 2px 10px; border-radius: 5px;">Allocate</span>
1750102	1213	<span style="background-color: #00AEEF; border: 1px solid #00AEEF; padding: 2px 10px; border-radius: 5px;">Allocate</span>
1750102	1214	<span style="background-color: #00AEEF; border: 1px solid #00AEEF; padding: 2px 10px; border-radius: 5px;">Allocate</span>

## Confirmation



Application ID: 1750102

Unit Number: 1212

From Date: dd/mm/yyyy

Duration (Days): 365

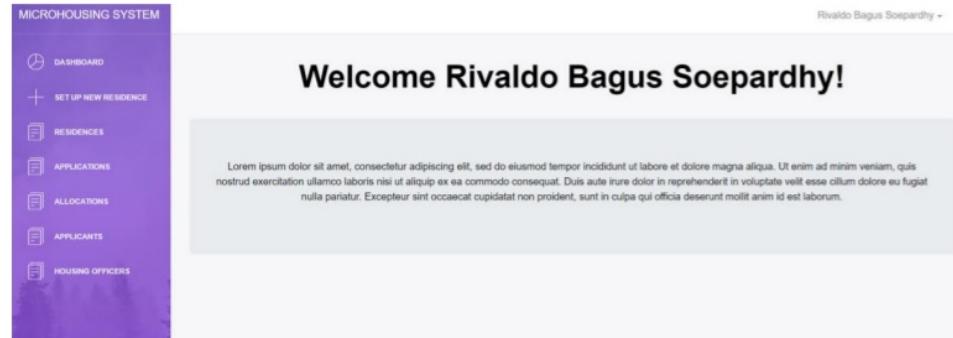
Change Acceptance Status: Processing

Change Payment Status: Unavailable

Are you sure you want to allocate?

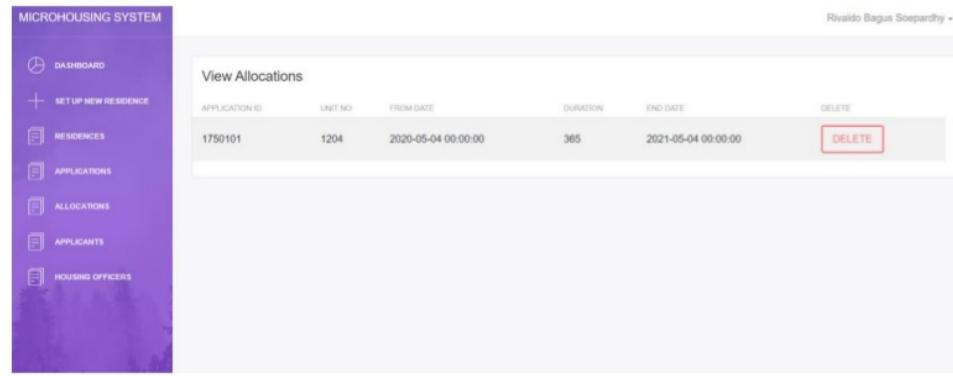
Yes No

If the housing officer click the “No” button it will be back to dashboard for housing officer



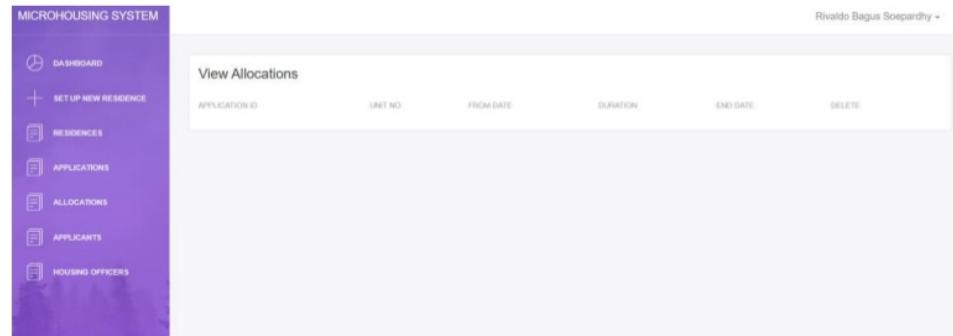
## 12. View Allocate Housing

To view all the allocation that input by the housing officer



## 13. Delete Allocate Housing

To delete the allocation in the database



## 14. Submit Application

To submit the application

DBKL MicroHousing My Profile Back to Dashboard Home Contact

Submit Application

Applicant ID	2029100
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

DBKL MicroHousing My Profile Back to Dashboard Home Contact

Submit Application

Applicant ID	2029100
Residence ID	100040
Required Month	E.g. May, June...
Required Year	E.g. 2020

Please fill out this field

Submit Application

## 15. View Residence (Applicant)

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

DBKL MicroHousing My Profile Back to Dashboard Home Contact

Residences

Residence ID	Address	Number of Units	Size Per Unit (Meter Square)	Monthly Rental (Malaysian Ringgit)	
100040	Kuala Lumpur	4	302 m <sup>2</sup>	RM200	<a href="#">Submit Application</a>
100041	Perang	8	280 m <sup>2</sup>	RM155	<a href="#">Submit Application</a>
100042	Kuantan	3	400 m <sup>2</sup>	RM145	<a href="#">Submit Application</a>

## 16. View Applications

To view submitted application by the applicant

The screenshot shows a dashboard for DBKL MicroHousing. At the top, there's a banner with the text "Hello Wulandari Maharani!" and a profile picture. Below the banner, the word "Application" is centered. A table lists three applications:

Application ID	Residence ID	Application Date	Required Month	Required Year	Status	Payment (Ringgit)	Payment Status
1750101	100041	2020-05-03 18:31:01	June	2020	Processing	RM155	Unavailable
1750102	100042	2020-05-03 18:31:20	July	2020	Processing	RM145	Unavailable
1750103	100040	2020-05-03 18:33:16	May	2020	Processing	RM200	Unavailable

## 17. Registered Applicant

To display the entire applicant that already registered for every applicant

The screenshot shows the MICROHOUSING SYSTEM dashboard. On the left, a sidebar menu includes options like DASHBOARD, SET UP NEW RESIDENCE, RESIDENCES, APPLICATIONS, ALLOCATIONS, APPLICANTS, and HOUSING OFFICERS. The main area is titled "Registered Applicants" and displays a table with two rows of data:

APPLICANT ID	FULL NAME	USERNAME	EMAIL	MONTHLY INCOME	EDIT	DELETE
2020101	Wulandari Maharani	wulan	wulan@gmail.com	10000	<button>EDIT</button>	<button>DELETE</button>
2020102	Monica Angela	Monica	Monica@gmail.com	300000	<button>EDIT</button>	<button>DELETE</button>

## 18. Edit Applicant

To edit the applicant in the form

Rivaldo Bagus Soepardhy +

Edit Applicant

Applicant ID	2020102
Full Name	Monica Angela
Username	Monica
E-mail Address	Monica@gmail.com
Monthly Income (Malaysian Ringgit)	30000

Update Applicant

## 19. Delete Applicant

To delete the applicant when it's not in the system

Rivaldo Bagus Soepardhy +

Registered Applicants

APPLICANT ID	FULL NAME	USERNAME	EMAIL	MONTHLY INCOME	EDIT	DELETE
2020101	Wulandari Maharani	wulan	wulan@gmail.com	10000	<button>EDIT</button>	<button>DELETE</button>

## 20. Registered Housing Officer

To display the entire housing officer that already registered for every housing officer

Rivaldo Bagus Soepardhy +

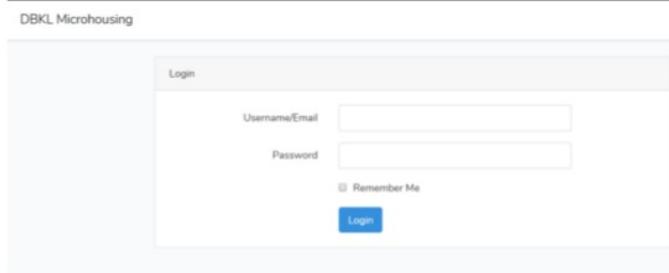
Registered HousingOfficer

HOUSING OFFICER ID	FULL NAME	USERNAME	EMAIL
2020100	Rivaldo Bagus Soepardhy	aldobagus	aldobagus@hotmail.co.id

## Non-functional Requirements

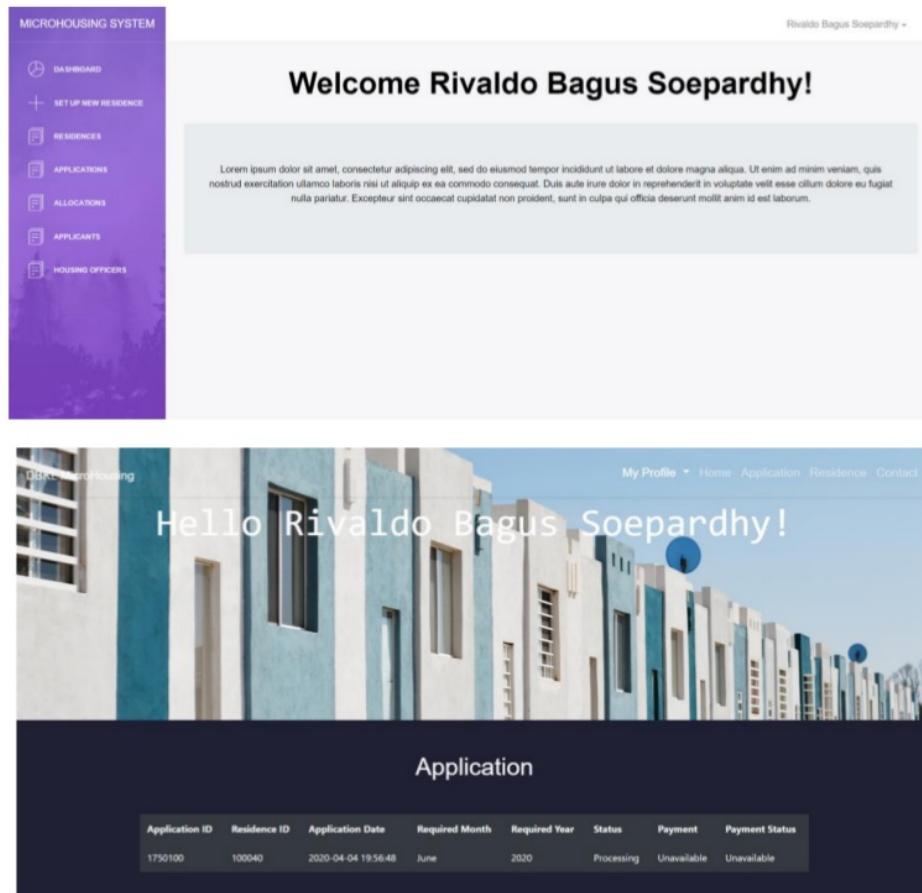
We will test the system according to the non-functional requirements that have been determined previously.

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



The image shows a screenshot of the DBKL Microhousing login page. The title bar says "DBKL Microhousing". Below it is a "Login" form with fields for "Username/Email" and "Password", a "Remember Me" checkbox, and a blue "Login" button.

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



The image displays three screenshots of the Microhousing System interface. The top screenshot shows the "Welcome Rivaldo Bagus Soepardhy!" screen with a purple sidebar containing navigation links like Dashboard, Setup New Residence, Residences, Applications, Allocations, Applicants, and Housing Officers. The middle screenshot shows a banner with a colorful building image and the text "Hello Rivaldo Bagus Soepardhy!". The bottom screenshot shows an "Application" table with one row of data:

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 its 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()
```



- 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
<b>Click the drop-down arrow to toggle column's visibility.</b>						
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



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

The image consists of three vertically stacked screenshots of a web-based application named "MICROHOUSING SYSTEM".

- Screenshot 1: Set Up New Residence**  
This screen shows a form for setting up a new residence. It includes fields for Staff ID (2020100), Address (Enter the Address), Num of Units (Enter the Number), Size of Units (Meter Square) (Enter the Size), and Monthly Rental (Malaysian Ringgit) (Enter the Price). A "Save Data" button is at the bottom.
- Screenshot 2: View Applications**  
This screen displays a table of applications. The columns are labeled: APPLICANT ID, APPLICATION ID, RESIDENCE ID, APPLICATION DATE, REQUIRED MONTH, REQUIRED YEAR, STATUS, ALLOCATE, EDIT, and REJECT. One row is shown with values: 2020101, 1750103, 100040, 2020-05-03 20:34:44, December, 2020, Approved, ALLOCATE (highlighted in green), EDIT, and REJECT.
- Screenshot 3: Edit Residence**  
This screen shows a form for editing a residence. It includes fields for Address (Denpasar), Num of Units (3), Size of Units (40), and Monthly Rental (150). A "Update Data" button is at the bottom.

## **Test Analysis Report**

After conducting all of the required tests with full functionality, 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.

## **Conclusion**

Here is our review for the project:

- By Rivaldo Bagus Soepardhy | E1700882

After all of the process of developing, I think we already did a very good job. Although we are inexperienced, still new to programming, we still managed to finish the project. From my perspective, supported with all of the testing has been done, we managed to reach our goal in Assignment 1. We must admit that we are still inconsistent in our timing because of our personal interest, lack of skill, and a lot of miscommunication. However, that does not hinder the fact that we can manage to finish the web with very functional system.

Something that went wrong with this project is our timing, skill, and communication. Even though we already try to make schedules (Gantt chart, etc.), we are still left behind due to our lack of skill in coding, resulting in unexpected long work hours, because there are many unexpected obstacles, such as errors, bugs, and false-logic of code. As for our communication, miscommunication still present. In fact, due to recently Covid19 pandemic, making the communication even harder, because we have to communicate through devices instead of direct meeting. This even leads to more miscommunication, misinterpretation, and misunderstanding.

Although so, things that went right is our cooperation. With all of the challenges we have in our project, we still managed to keep our cooperation and better understanding in mind. The perseverance from each of us to keep working hard in the project is paying off. We also managed to keep helping each other when there is trouble existed. We really understand the meaning of cooperation, because this is the key for our team to finish the project.

Some of the things that I would have done differently is to carefully design. This is a very critical part as this is our big picture of how our system will become. Inconsistency may present, but we still need to keep the basic design unchanged, to make sure the system will work as what we want to. Time-persistent is also what I would need to keep fighting with, as in this case of condition, communication and personal interest is changed drastically. I may also have just upgraded my skill in coding, so we can be less stressed due to our lack of skill in coding.

➤ By Luh Wulandari Maharani | E1700873

**1. Did the group meet the objectives defined in Assignment 1?**

From Assignment 1 our system is running well and beyond our expectations like the system is right - means all the way, the UI looks great and is easily accessed by the user.

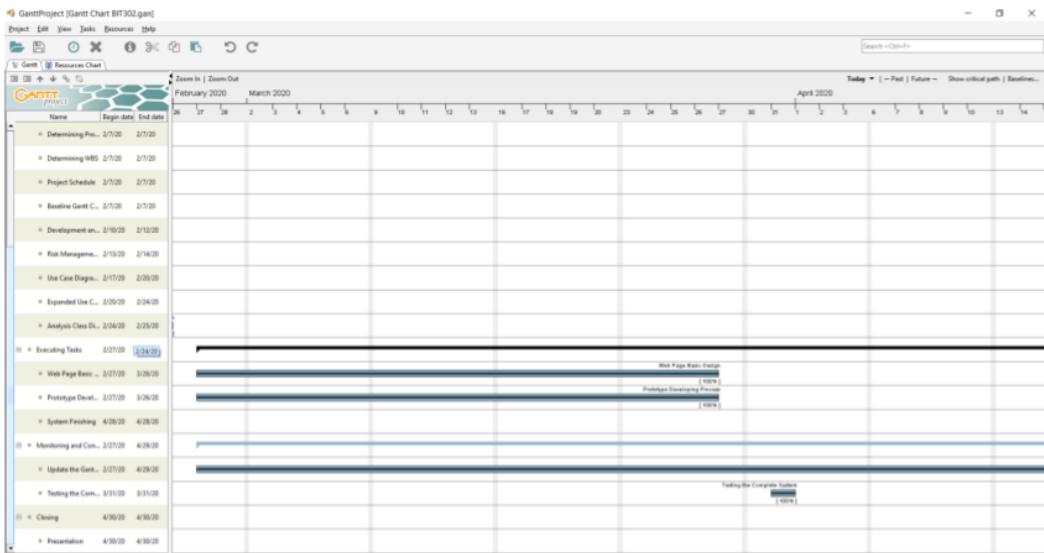
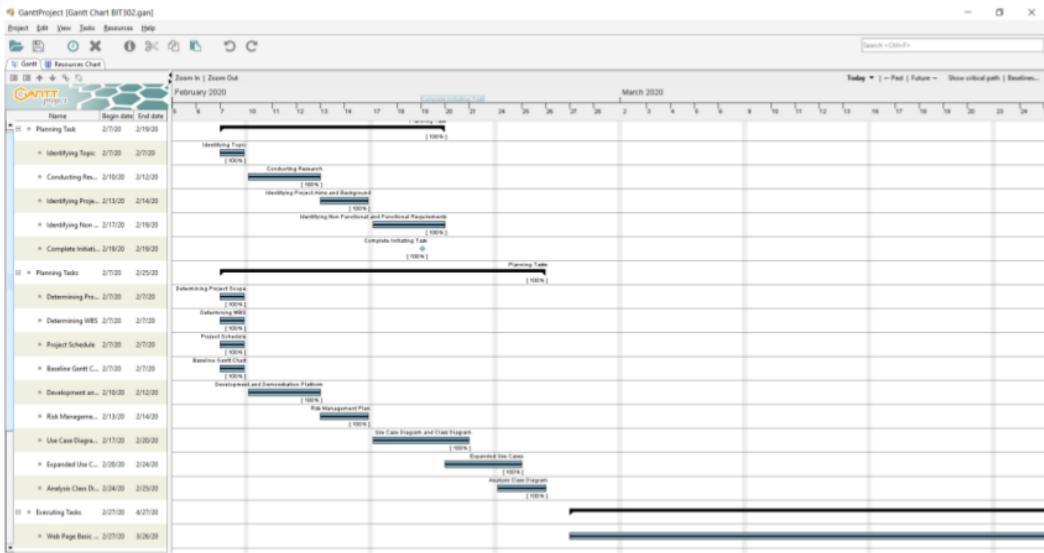
**2. What went wrong and what went right?**

- What went wrong from our project is we are late one day in making use cases and web design due to a religious ceremony, in a skill I could say I still learning to use laravel here I need time to understand the framework that I use
- What went right from our project that is we have good cooperation to make this project and communication is very smooth there are no obstacles at all.

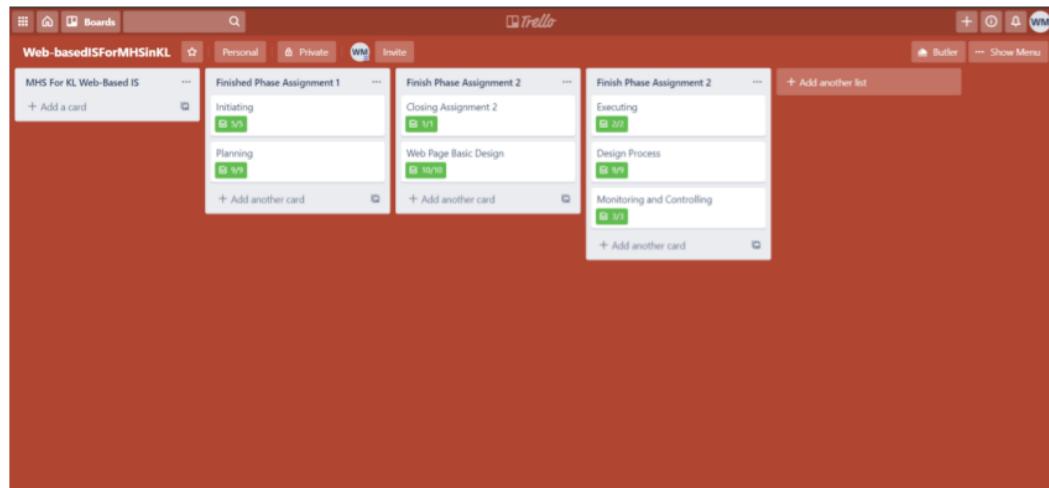
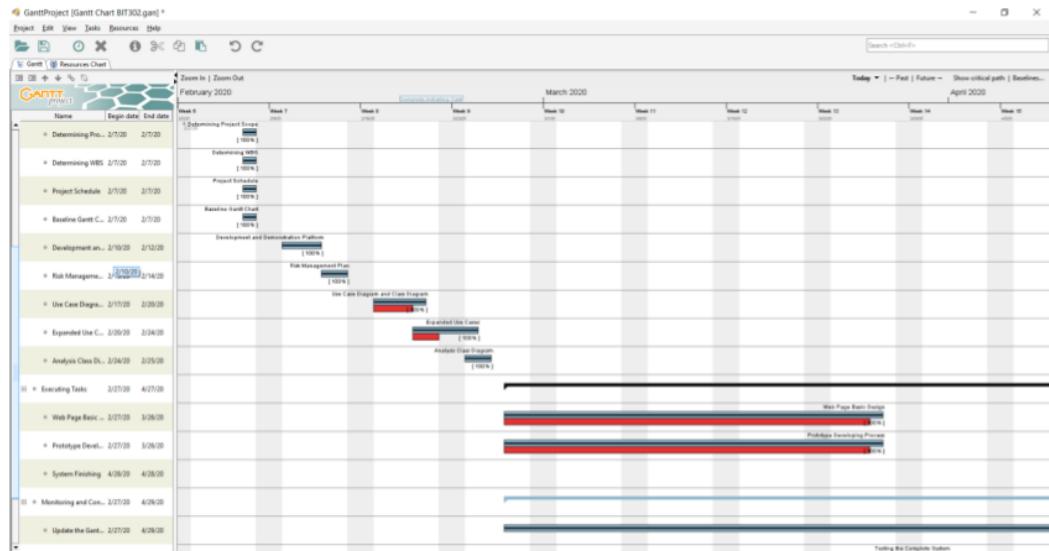
**3. What you would have done differently?**

Although I do not understand and have never touched the framework, I still want to learn to use it in addition to our system project running, I can also add to my experience in the framework.

# Updated Gantt Chart, Trello Board, GitHub



## Baseline Gantt Chart



[https://github.com/aldobagus725/WebBased\\_InfoSys\\_MHSinKL](https://github.com/aldobagus725/WebBased_InfoSys_MHSinKL)

The screenshot shows the GitHub repository page for 'WebBased\_InfoSys\_MHSinKL' owned by 'aldobagus725'. The repository has 133 commits, 3 branches, 0 packages, 0 releases, and 2 contributors. A yellow banner at the top indicates a potential security vulnerability in one of the dependencies. The commit history for April 4, 2020, is displayed, showing various updates and merges.

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

aldobagus725 ORET ORET Latest commit ae24a42 4 hours ago

Commit	Message	Time
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
...	ITERATION 1 GOOD	...

-o 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_Info... [diff]	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_Info... [diff]	aldobagus725 committed 2 hours ago	4481948	🔗
PPT	aldobagus725 committed 2 hours ago	107087c	🔗
Merge branch 'master' of https://github.com/aldobagus725/WebBased_Info... [diff]	LuhWulandari committed 2 hours ago	5b7521c	🔗



# ASS3\_BIT302\_E1700882\_E1700873\_GroupAssignment - revised

---

## ORIGINALITY REPORT

---



## PRIMARY SOURCES

---

1	dl.dropboxusercontent.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