



FIT5032 Design Report

Major Application Development

{Credit/Distinction/High Distinction}

SWEET HOME
STUDENT NAME:PEIYU LIU
ID:31153291

Contents**Page**

Your design report must include the following:

Credit Level

1. Web application title and description
2. User stories and a Use case diagram
3. Block/Functional diagram
4. Your selected approach when constructing the application

Additional Distinction Level (the above and the following)

5. Class Diagram or Entity Relation Diagram
6. Mockup prototypes and Implementation
7. Data dictionary
8. Usability Design Review

Additional High Distinction Level (the above and the following)

9. Development Methodology
10. Versioning
11. Checklist of site functionality.

1. Web Application Title and Description

Title: Sweet Home

Description: The purpose of making Sweet Home web application is to provide a series of house services for target users. The advice provided by this web application can cover a wide range of areas, including the following:

Sweet Home is a real estate website that provides real estate services such as “register account”, “log in”, “login in with google account”, "upload hotel sources", "upload house sources", "booking house" and “send email with attachment”.

1) Agency options aspect: admin users can update different agencies on the website to provide agency sources for normal users to select. Each agency contains a name, constructed year, star level, description, geolocation.

2) house options aspect: admin users can update different houses on the website to provide houses sources for normal users to select. Each house contains a house type, description and geolocation.

3) order options aspect: once normal users decide to book one house, users can create a booking list to make a contract. The booking list contains the user account, house, agency, and booking start date and end date.

Additional function: all users can send an email with attachments to others. Website has role authentication, it means that admin users can see all functions page, but normal users can only see normal pages. Admin users can create new agencies, new houses, normal users can only choose their options.

2. User stories and Use case diagram

As a normal user, I want to register an account so that I can log in to this website.

As a normal user, I want to book a house so that I can live in this house during my booking period.

As a normal user, I want to choose an agency so that I can use this agency to find house resources.

As a normal user, I want to send emails to others' email addresses so that I can contact other people.

As a normal user, I want to use geolocation to search for a house address so that I can see the location of this house.

As a normal user, I want to sign in to this website using google email so that I do not need to sign up for a new account.

As a normal user, I want to see five pieces of agencies on one view page so that I can see content easily.

As an admin user, I want to create a new agency so that I can update new agency resources on the website.

As an admin user, I want to create a new house so that I can update the new house resource on the website.

As an admin user, I want to modify booking information so that I can help users to update booking information.

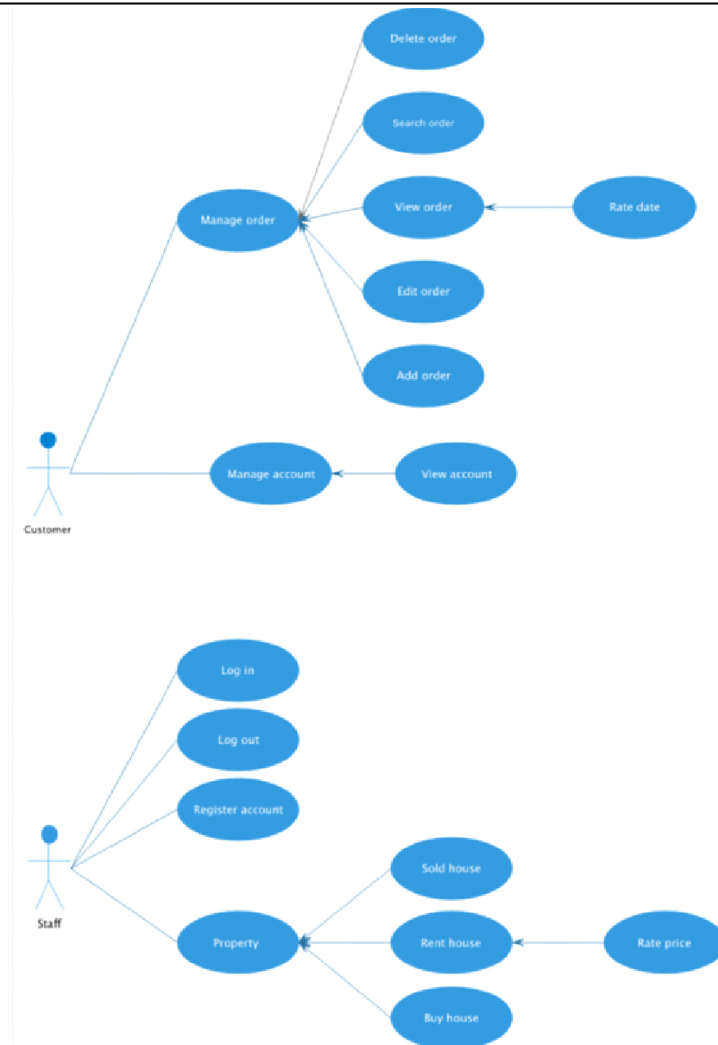


figure 1: use case diagram

3. Block/Functional diagram

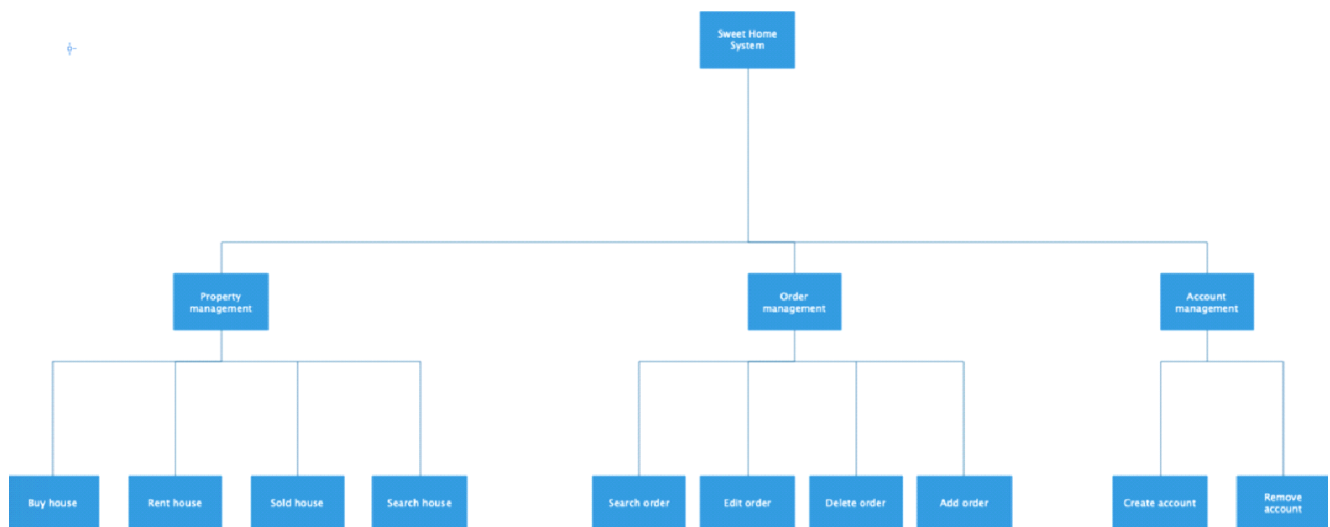


figure 2: 3-level block diagram

4. Your selected approach when constructing the application.

Model first

-more easy to release with beginner(know limited c#)

I chose the model first to make my ASP.NET MVC website application. I am a beginner at c# and MVC structure. So model first can help me to reduce program code. I can design my database and entity diagram firstly. Then generate a model view from my database design. Finally, create controllers from my model design.

5. Class Diagram or Entity Relation Diagram

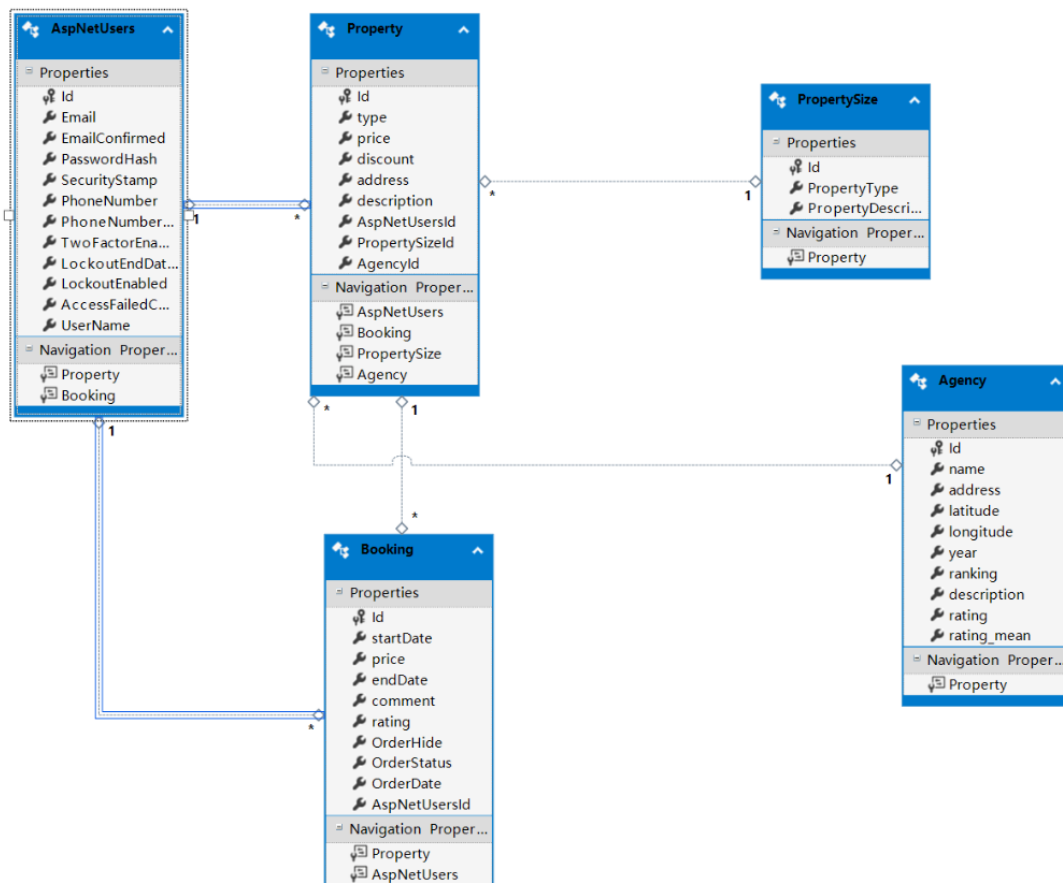


Figure 3: Entity relationship diagram

6. Data dictionary

AspNetUsers

Name	Description	Data type	Length	Null	Constraint
Id	\	nvarchar	128	no	primary key
Email	\	nvarchar	256	no	\
Password	\	nvarchar	256	no	\
PhoneNumber	\	nvarchar	128	no	\
UserName	\	nvarchar	256	no	\

Property

Name	Description	Data type	Length	Null	Constraint
Id	property id	int	increment	no	primary key
type	property type	nvarchar	\	no	\
price	price	float	\	no	\
discount	discount*price	float	\	yes	\
address	address	nvarchar	\	no	\
description	property info	nvarchar	\	no	\
AspNetUsersId	user account id	nvarchar	128	no	foreign key
PropertySizeId	property size Id	int	\	no	foreign key
AgencyId	agency id	int	\	no	foreign key

Booking

Name	Description	Data type	Length	Null	Constrain
Id	booking id	int	increment	no	primary key
startDate	order start date	nvarchar	\	no	\
price	price	float	\	no	\
endDate	order end date	nvarchar	\	no	\
comment	feedback	nvarchar	\	no	\
rating	give user grade	int	\	yes	\
OrderHide		nvarchar	\	yes	\
OrderStatus	status: processing/finis hed	nvarchar	\	yes	\
Property_Id	property id	int		no	foreign key
AspNetUsersId	account id	nvarchar	128	no	foreign key
OrderDate	order created date	nvarchar		no	\

PropertySize

Name	Description	Data type	Length	Null	Constrain
Id	propertySize id	int	increment	no	primary key
PropertyType	single/double/ki ng	nvarchar	\	no	\
PropertyDescri ption	small single size...	nvarchar	\	yes	\

Agency

Name	Description	Data type	Length	Null	Constrain
Id	agency id	int	increment	no	primary key
name	agency name	nvarchar	\	no	\
address	address	nvarchar	\	no	\
latitude	geo location	float	\	no	\
longitude	geo location	float	\	no	\
year	constructed year	int	\		\
ranking	agency star rating	int	\	yes	\
description	agency description	nvarchar	\	yes	\
rating	rating value	int	\	yes	\
rating_mean	average rating value	int	\	yes	\

Rating

Name	Description	Data type	Length	Null	Constrain
Id	rating record id	int	increment	no	primary key
content	user comment	nvarchar	\	no	\
date	rating created date	nvarchar	\	no	\
AspNetUsersId	account id	nvarchar	128	no	foreign key

7. Mockup prototypes and implementation with user registration and authentication

The mockup shows a web browser window with the URL `https://sweethomeservice.edu`. The registration form includes the following elements:

- A red error message: "Please enter email with right format" above the "Email address" input field.
- A "Password" input field.
- A red error message: "Twice password are no match" above the "Confirm password" input field.
- A "Register" button.
- A "Login quickly with google account" link.
- A "Google account" button.

Figure 4: Register mockup diagram

When a user registers a new account, the system will identify the inputs by user. To ensure the user can remember his/her password, the system will ask the user to enter password twice and authenticate passwords are the same. Email addresses should also follow email format.

```
// GET: /Account/ConfirmEmail
[AllowAnonymous]
public async Task<ActionResult> ConfirmEmail(string userId, string code)
{
    if (userId == null || code == null)
    {
        return View("Error");
    }
    var result = await UserManager.ConfirmEmailAsync(userId, code);
    return View(result.Succeeded ? "ConfirmEmail" : "Error");
}

//
// GET: /Account/ForgotPassword
[AllowAnonymous]
public ActionResult ForgotPassword()
{
    return View();
}

//
// POST: /Account/ForgotPassword
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindByNameAsync(model.Email);
        if (user == null || !(await UserManager.IsEmailConfirmedAsync(user.Id)))
        {

```

Figure 5: Screenshot of account controller(authentication)

Git: https://github.com/SukiGit2021/FIT5032/blob/16cf250e8365e49132585d365054fce1825dc21e/FIT5032_Ass_version7/Controllers/AccountController.cs

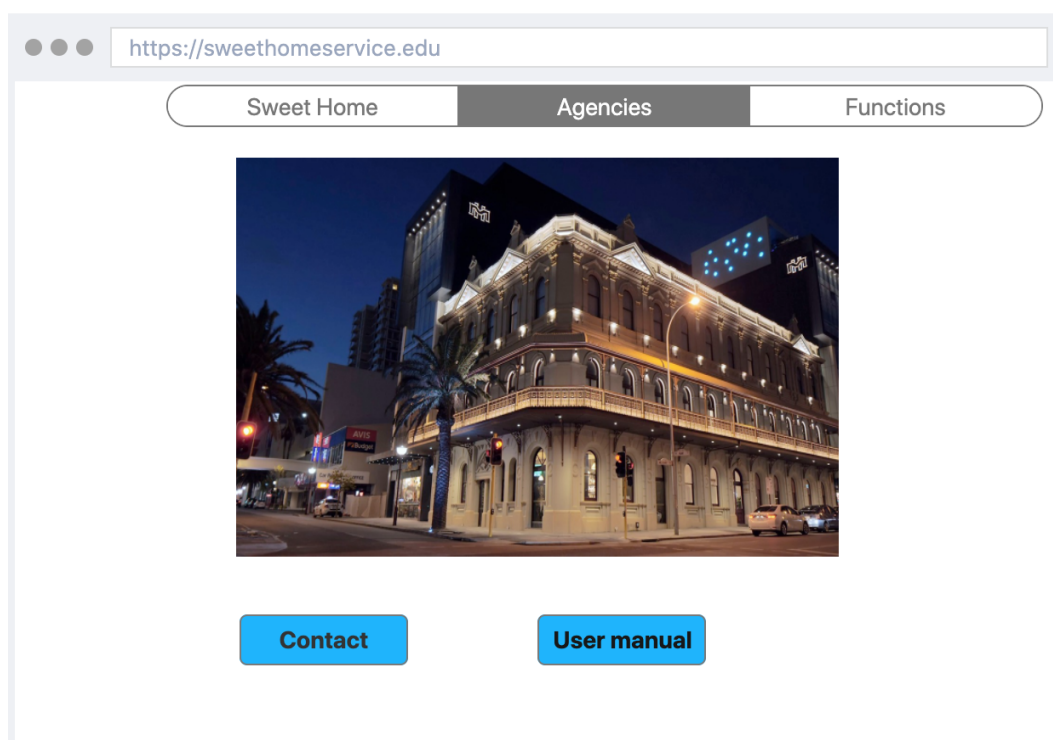


Figure 6: Main page of sweet home before login authentication

Before logging in successfully, users can only show the main page and agencies' information. users can not book property and create order.

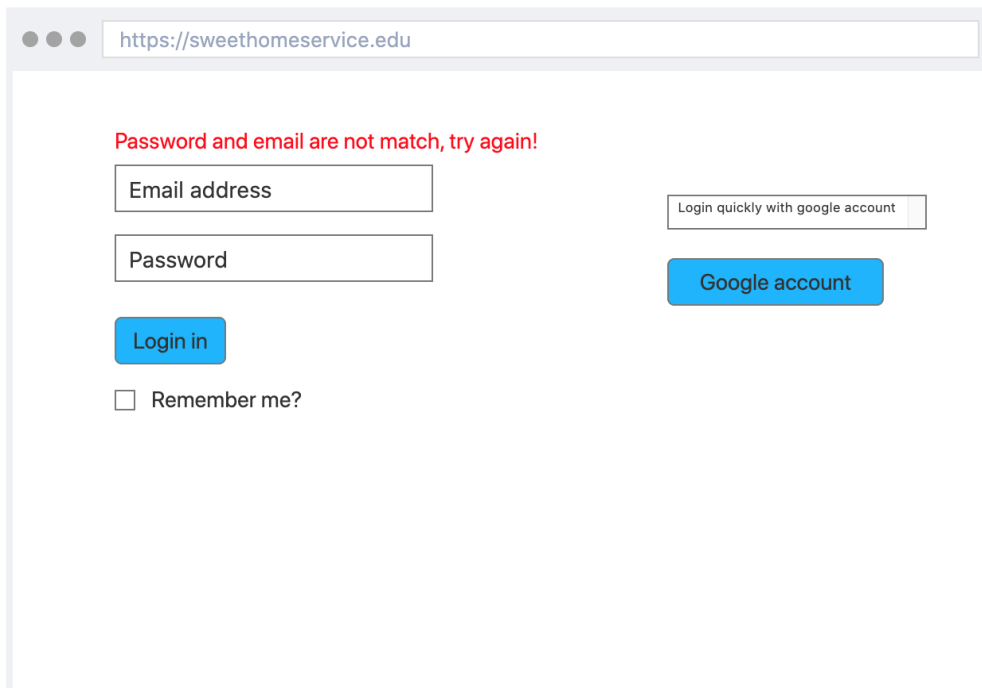
A mockup of a web browser window showing a login page. The address bar displays 'https://sweethomeservice.edu'. The page features a red error message: 'Password and email are not match, try again!'. Below this, there are input fields for 'Email address' and 'Password'. To the right of the password field is a button labeled 'Login quickly with google account'. Below the email field is a blue button labeled 'Login in'. At the bottom left, there is a checkbox labeled 'Remember me?'. A blue button labeled 'Google account' is positioned to the right of the 'Login in' button.

Figure 7: login authentication mockup

When a user finishes registration and tries to log in, the system will validate users' inputs. email should be already stored in the database and password must be matching. Otherwise users can use google accounts to login directly.

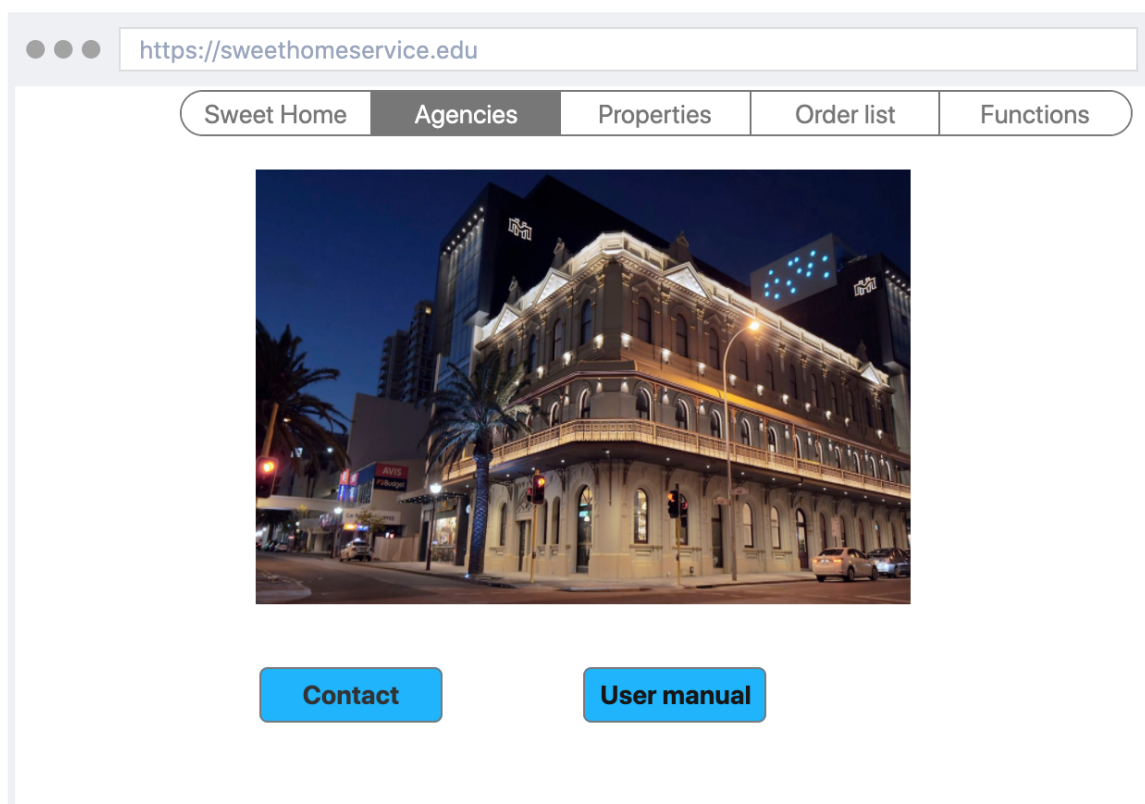


Figure 7: After login, show a different main page.

After login, admin users can see all functions like creating new agencies and properties, modifying finished

order information. a normal user can see the agency page, book properties and create new orders.

```

    @Html.ActionLink("Sweet Home", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
</div>
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        @if (User.Identity.IsAuthenticated)
        {
            <li> @Html.ActionLink("House Agency", "Index", "Agencies")</li>
        }

        @if (User.IsInRole("admin"))
        {
            <li>@Html.ActionLink("House List", "Index", "Properties", new { id = -99 }, null)</li>
        }
        @if (User.IsInRole("admin"))
        {
            <li> @Html.ActionLink("House Type", "Index", "PropertySizes")</li>
        }
        @if (User.Identity.IsAuthenticated)
        {
            <li>@Html.ActionLink("Order List", "Index", "Bookings")</li>
            <li> @Html.ActionLink("Open Day", "dateConstraint", "Home")</li>
        }
        <li>@Html.ActionLink("Multiple Services", "FunctionsPage", "Home")</li>

    </ul>
    @Html.Partial("_LoginPartial")
</div>

```

Figure 8: User roles authentication, show different main page

Git: https://github.com/SukiGit2021/FIT5032/blob/16cf250e8365e49132585d365054fce1825dc21e/FIT5032_Ass_version7/Views/Shared/_Layout.cshtml

8. Usability Design Review

Strive for consistency: each view used bootstrap CSS style to realise consistency.

Offer informative feedback: when the user sends an email successfully, the page will print a notification "Email sent successfully" to remind the user that the action is done.

design dialogs to yield closure;

Prevent errors: When users register a new account, each variable has validation to avoid user input in the wrong format. Description of agency and house also has string length validation.

9. Development Methodology

I use the code-and-fix model to design this website application, only to fix any problems along the way. After completing the initial design report, I went straight into the website application design. There are a lot of errors and bugs in the project implementation process, and I modify and fix them when they appear. This saves time, and I don't have enough time to finish all the designs before writing code. So I'm going to modify

my feature design while writing code. Complete the website application in a small amount of time. This Code and Fix method is suitable for novices like me to design the program.

10. Versioning

I use GitHub to ensure my version control.

Git: https://github.com/SukiGit2021/FIT5032/tree/Ass_final_version7.0/FIT5032_Ass_version7

11. Innovation and Research

Add geographical API on the website to help users see house locations and use users' current site to generate a driving line for users to arrive at this house. The user experience can be optimized, and system feedback prompts can be added. When the user clicks the button, the system needs a certain amount of time to react, and a loading icon can be added to remind the user that we are taking action.

12. Checklist of site functionality

	TICK if complete
1. (Layout Page)	
Good Design	
Stylesheet	
JavaScript	
Menu	
2. (Home page)	
Design and content	
Banner Image	
3. (User Log in)	
Web form and validation controls	
Formatted data entry display	
Overall page design	
4. (Customised Views and Controllers)	
Customised Views	
Customised Controllers	
Other customisations	
5. (Documentation)	
Code Comments	
Attribution of Source of any code used	
6 Business Requirements	
BR(A1): for P	
BR(A2): for P	
BR(B1): for C to C+	
BR(B2): for C to C+	
BR(C1): for C+ to C++	
BR(C2): for C+ to C++	
BR(C3): for C+ to C++	
BR(C4): for C+ to C++	
BR(D1): for D to D++	
BR(D2): for D to D++	
BR(D3): for D to D++	
BR(D4): for D to D++	
BR(E1): for HD to HD+	
BR(E2): for HD to HD+	
BR(F1): for HD+ to HD++	
Audit	
No breaking of copyright	