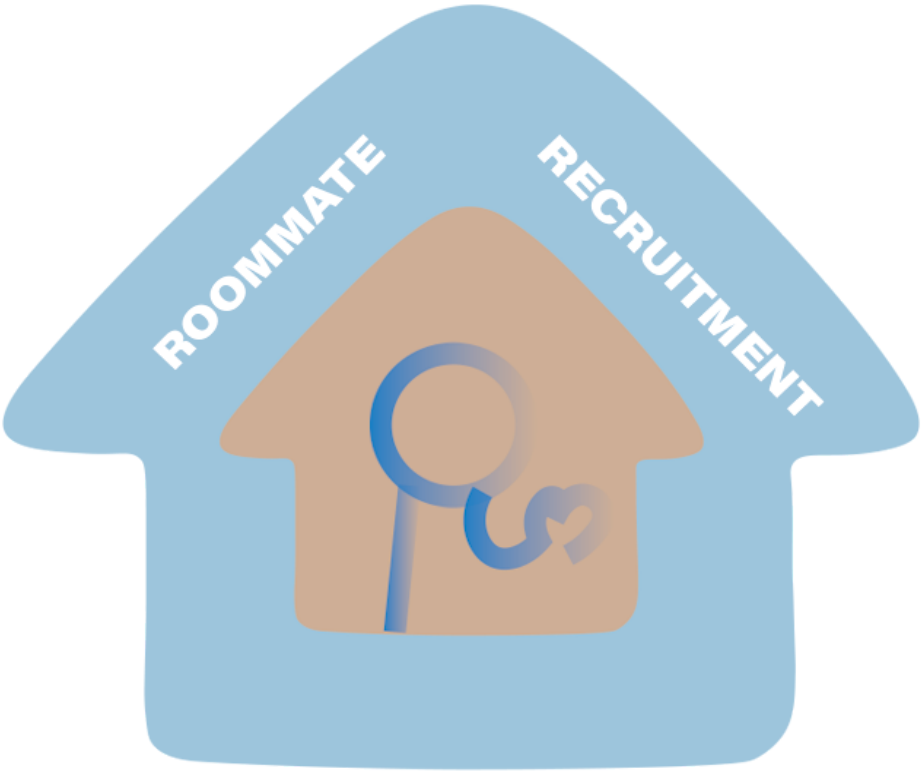
Portfolio



Team 60 :

Liheyu.Zhang Han.Jiang Zhicheng.Wang

Jiaming.Zhang Yuqi.Xiao Zexun.Lan

UNIVERSITY OF LIVERPOOL

GROUP PROJECT DECLARATION OF ACADEMIC INTEGRITY

NAME (Print)	LIHEYU.ZHANG
STUDENT NUMBER	201522650
GROUP NAME	TEAM 60
MODULE TITLE/CODE	COMP 208
TITLE OF WORK	ROOMMATE RECRUIMENT

This form should be completed by one student on behalf of the group and appended to any piece of work that is submitted for summative assessment. Submission of the form by electronic means by a student constitutes their confirmation of the terms of the declaration.

Students should familiarise themselves with Section 9 of the Code of Practice on Assessment and Appendix L of the University's Code of Practice on Assessment which provide the definitions of academic malpractice and the policies and procedures that apply to the investigation of alleged incidents.

Students found to have committed academic malpractice are liable to receive a mark of zero for the assessment or the module concerned. Unfair and dishonest academic practice will attract more severe penalties, including possible suspension or termination of studies.

STUDENT DECLARATION

I confirm that the group has read and understood the University's Academic Integrity Policy.

I confirm that the group has acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that no member of the group has copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work.

I confirm that no member of the group has previously presented the work or part thereof for assessment for another University of Liverpool module.

I confirm that no member of the group has copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that no member of the group has incorporated into this assignment material that has been submitted by any other person in support of a successful application for a degree of this or any other University or degree awarding body.

SIGNATURE……Liheyu.Zhang……

DATE……11.5.2021……

Table of Contents

1.0 Overview of Application

Roommate recruitment (RR) is the website that allows the target group in need to choose preferred roommates. The main purpose of RR is to provide a platform where users post and also find attractive posts under four distinct categories (area, flat name, rent prices, room capacity) and post comments or apply to join under the posts they are interested in. The site allows users to have more autonomy in finding roommates in the case of renting student accommodation, as most student accommodation is randomly arranged.

We set up the site to make it easier for students at the University of Liverpool. When using Facebook, you can see that there are groups of University of Liverpool students looking for flatmates, where users post information about their flat and their flatmates in general. However, Facebook does not categorize the posts and show if there are any new roommates currently in the group. RR's original intention was to present the information to users in a more efficient way so that they could filter the stream themselves and see the number of people under the post is full or not. We want the site to be available to all students at the University of Liverpool. Perhaps it could be made available to all UK universities in the future, the intended audience for the website includes but is not limited to the U.O.L.

2.0 People and Roles

Our group consisted of six members Liheyu zhang, Zexun Lan, Han Jiang, Yuqi Xiao, Zhicheng Wang, Jiaming Zhang. All tasks were carried out as a group in order to give everyone a complete and clear understanding of our project. We organized meetings when we needed to discuss different aspects of the websites and make an agreement with these questions. Every member never missed a meeting session. Below is a breakdown of the main tasks performed by each team member, as well as any additional contributions they made.

MEMBER	Main Tasks
Yuqi Xiao	· Requirement's elicitation and analysis · Website Design and Implementation · Acceptance/user testing · System testing & debug · Documentation

Jiaming Zhang	· Requirement's elicitation and analysis · Website Design and Implementation · Acceptance/user testing · System testing · Documentation & Design user manual
Zexun Lan	· Requirement's elicitation and analysis · Website Design and Implementation · Collection and entry of sample data · System testing · Documentation
Liheyu zhang	· Requirement's elicitation and analysis · Database design and implementation · Website Design and Implementation · System testing · Artwork design · Documentation & Meeting notes
Han Jiang	· Requirement's elicitation and analysis · Database design and implementation · Website Design and Implementation · System testing · Documentation
Zhicheng Wang	· Requirement's elicitation and analysis · Database design and implementation · Website Design and Implementation · System testing · Documentation

3.0 Achievements of Objectives

The Roommate Recruitment (RR) is a great system, and it consists of six different subsystems and other additional features. The project team created a multi-user database application complete with a fully functional server and web-based client applications. The webpages are written in HTML, JavaScript.

The RR system has met all requirements and main mission objectives mentioned in the requirements specification. All queries used and reports produced by the system fulfill the specified requirements. The database structure has been slightly modified based on design specifications. But functionality closely reflects it.

However, due to the time constraint on the project, we failed to complete the recommendation algorithm which used to sorting posts and used vacancy as an alternative. As well as autofill data and version control.

3.1 Account Management

The Account Management system includes account registration and login. Users are required to use their university email and strong password when registering an account. The validation code will be sent to the email user has filled in. Registration can only be completed after entering the verification code. When registering an account, we also provide a variety of personalized avatars to choose from.

3.2 Post and Comment Management

The Post and Comment Management system is the core component of the whole system. It manages all the operations (create, edit, delete and apply for entering a room group) with post and related comments. Users are restricted to edit their posts and comments only. In the database design here, foreign keys are used to ensure comments are related to posts and users. If the related post is deleted, all the comments under this post will also be deleted.

```
57  CREATE TABLE `Comment_Post` (
58    `commentID` int(11) NOT NULL,
59    `postID` int(11) DEFAULT NULL,
60    `userID` int(11) DEFAULT NULL,
61    `content` varchar(2000) DEFAULT NULL,
62    `date` varchar(50) DEFAULT NULL,
63    PRIMARY KEY (`commentID`),
64    KEY `fk_postID` (`postID`),
65    KEY `fk_userID` (`userID`),
66    CONSTRAINT `fk_postID` FOREIGN KEY (`postID`) REFERENCES `post` (`postID`),
67    CONSTRAINT `fk_userID` FOREIGN KEY (`userID`) REFERENCES `user` (`userID`)
68  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Figure 1: Foreign Key Constraint

3.3 Search and Sift Engine

Search and sift engines are designed to help user to sift out their interesting posts efficiently and precisely. Users can use searching boxes by selecting their preference conditions such as area, flat, and price interval. The engine will simply traversal and sift out those posts under these conditions from the database [5].

3.4 Personal Information Management

```
CREATE TABLE `User` (
  `userID` int(11) NOT NULL,
  `email` varchar(50) DEFAULT NULL,
  `username` varchar(50) DEFAULT NULL,
  `password` varchar(50) DEFAULT NULL,
  `gender` varchar(10) DEFAULT NULL,
  `profile` int(11) DEFAULT NULL,
  `major` varchar(30) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `religion` varchar(30) DEFAULT NULL,
  `nationality` varchar(50) DEFAULT NULL,
  `sexualOrientation` varchar(50) DEFAULT NULL,
  `marriedOrNot` int(11) DEFAULT NULL,
  `sleepingPreference` varchar(50) DEFAULT NULL,
  `mealPreference` varchar(50) DEFAULT NULL,
  `user_note` varchar(2000) DEFAULT NULL,
  PRIMARY KEY (`userID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Figure 2: User profile

Users can set and edit their personal information like avatar, religion, gender, nationality, sexual orientation, etc. Foundational information cannot be null but optional information can. It is worth noting that the username is unique and cannot be edited. This component is related to the user profile.

3.5 System Security

To protect user information, a regular expression is used to check if the user's password is a strong password which must contain at least one of "_", "-", "+", "@", one numerical number, and the length must be between 8 and 20 characters.

```
var validatePass = (rule, value, callback) => {
  if (value === '') {
    callback(new Error('Please enter password.'));
  } else if(!/^[a-zA-Z0-9-_+@]{8,20}$/.test(value)){
    callback(new Error("Password only contains 8-20 characters, numbers and '_', '-', '+', '@' notations"));
  }
  else {
    if (this.ruleForm.checkPass !== '') {
      this.$refs.ruleForm.validateField( props: 'checkPass');
    }
    callback();
  }
};
```

Figure 3: Check if the password is strong

3.6 Database Design and Management

The Roommate Recruitment system uses MySQL as its database. The database contains six tables (Figure 4). The physical data model is shown as well that the blue line indicates the relation between user and other tables and the red line is to the

post table (Figure 4.1). For other details about database design, please view Design Documentation to see more information.

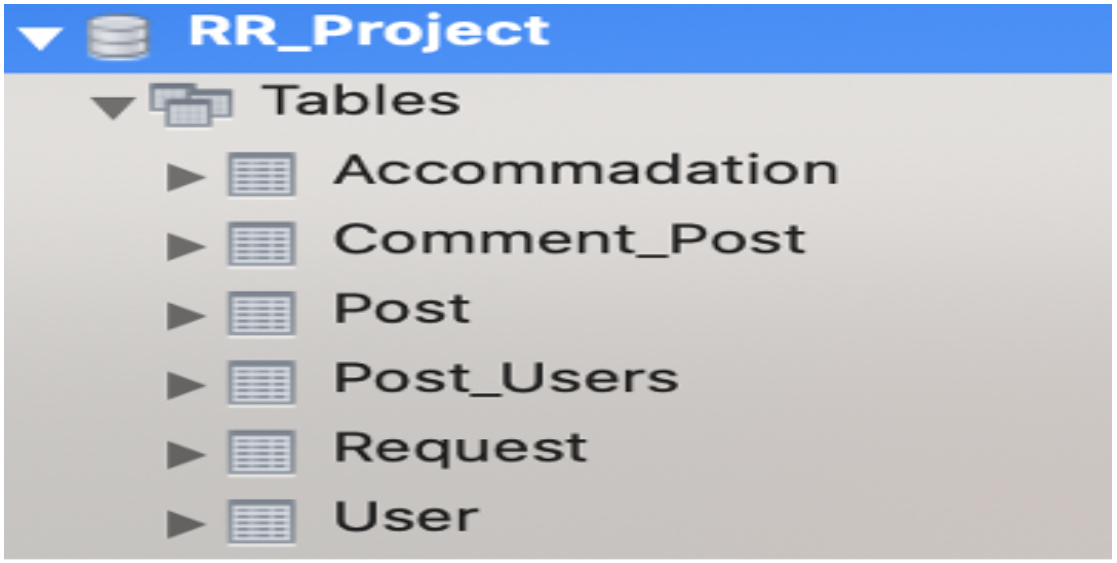


Figure 4: Tables Overview

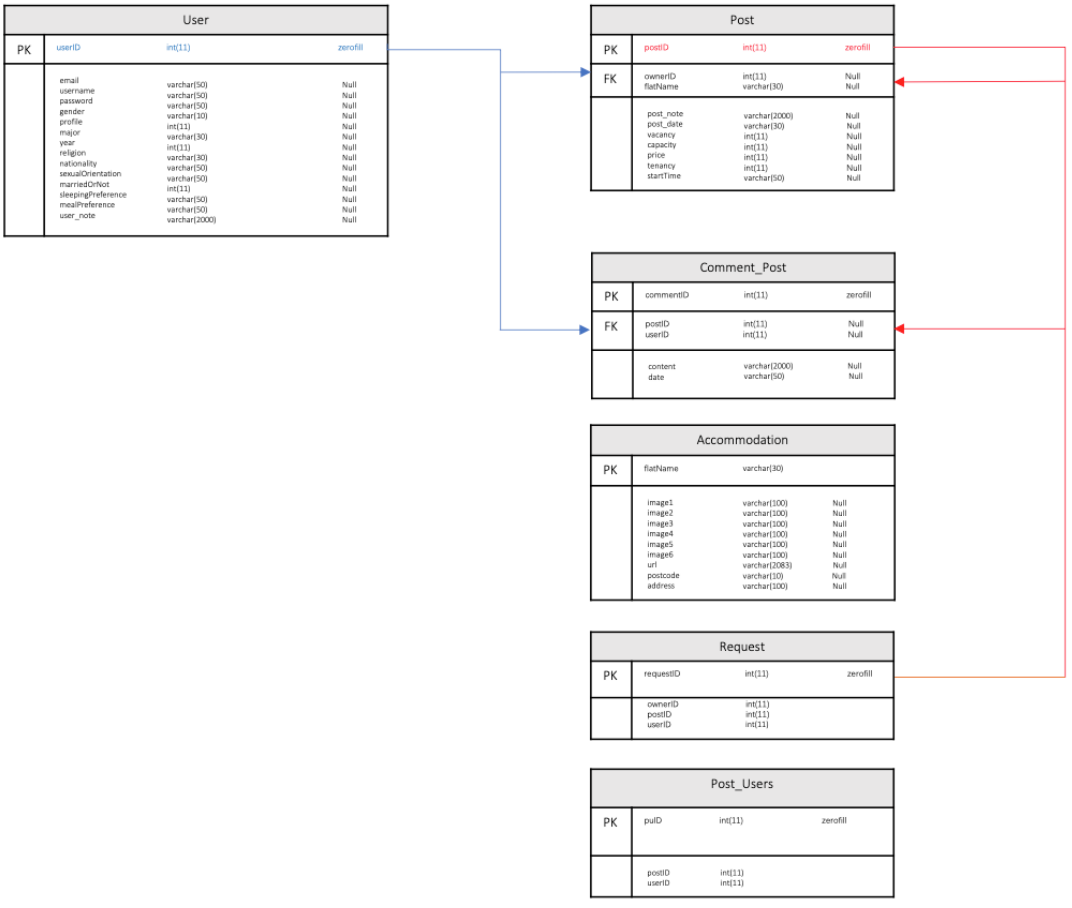


Figure 4.1: physical data model

The database has a simple structure. However, it may lead to inefficient. For example, a mass of time will be taken to find a specific post among the whole post table if there are large amounts of posts. Therefore, as the number of users increases, the cost of management may increase. Many optimization methods, for example, using index in B+ trees, could be used to improve the performance in the future.

3.7 Additional Features

Many other additional features are also implemented to improve the user experience.

3.7.1 Delete Confirmation

When users delete their posts, a pop-up window of confirmation will be shown in case they delete them by accident. Besides, once users delete their posts, the related comments, and the viewed/saved post history (even in other users' accounts) will also be deleted.

3.7.2 Feedback Email

Users can provide their feedback or contact the administrator by email: 443908400@qq.com which will help in gathering information regarding the site and clients' opinions on what should be done to improve on website services. This will help in collecting clients loyal regarding the services and also what is needed to be implemented.

4.0 Evaluation

4.1 Advantages

From a team management perspective, the team created a clear and structured way of working, making the purpose of tasks and activities clearer.

From a design perspective, we borrowed from Apple's product design philosophy and incorporated the Japanese Zen aesthetic of Wabi-Sabi into the site's interface design, making the site simple and easy to use, reducing the time cost of multiple choices. We abandoned traditional PHP in the development in favor of JS. node.js allows the system to handle high concurrency scenarios with higher performance and lower CPU overhead. From a market demand perspective, most websites focus on sharing rental information and neglect the need to find a roommate during the sharing process. RR can post to find a suitable roommate and at the same time learn about the availability of properties, meeting multiple needs with one system [3].

From an architectural point of view, the subject's system is made up of several subsystems. The low coupling and high cohesion make our system easy to maintain. So, the various modules in the system are not dependent on each other. When problems are encountered, they are also isolated in small, independent units of code. At the same time, the elements in a module are directly related to the functionality that the module is intended to provide, reducing the rate of code duplication.

4.2. Limitation

The Node.js used for the development of the system is weakly fault-tolerant and when an exception occurs in an asynchronous callback and the corresponding error's event is not subscribed to, then the whole process hangs. This can lead to program crashes in the long run.

Regarding web layout, web pages do not support small screens, such as mobile phones. As the web page is not responsive it requires some advanced knowledge of CSS. The website is currently only aimed at students studying at the University of Liverpool, which leads to a narrow audience. Further development is required. The same database also needs to be expanded.

The system is relatively rudimentary in terms of encryption for data security and requires the use of some desensitization techniques, and the company should have some regulations in place for data handling and access restrictions.

5.0 Future Developments

In terms of the user interface, the site's interface also needs to be beautified, including the arrangement of several components to improve the site's functionality.

There is a need to design a feature that allows for in-site chat, as well as a supporting notification so that users can better exchange information. A subscription system may need to be incorporated so that users do not miss important posts.

Adding asymmetric encryption technology to the system will help improve the security of the whole system.

Adding intelligent customer service, through AI technology, and a large amount of industry-related data, directly through intelligent chatbots, to reply to common questions for users, improving their service experience and reducing manual customer service costs by over 40%.

For further maintenance and updates, we will make the source code available on GitHub and fix bugs exposed during testing.

Use optimization methods such as B+ trees to make the database more efficient.

Refine the post-sorting recommendation algorithm for a better experience.

5.10 Roommate Application Form

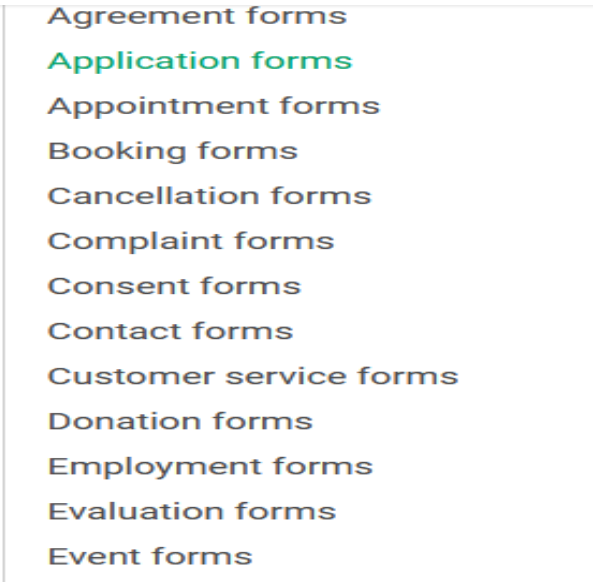


Figure 5: Application forms

This will be included in the site just in case there is an agreement, application, booking, consent deal that needs to be signed by the roommates. Whenever there is an organization of an event of function, the users can just fill the form to check their availability without them attending for a function for signatures and conformations.

6.0 Professional Issues

The BCS Code of Conduct establishes a code of ethics for the IT industry [1]. The code addresses different topics such as the public interest, professional competence and integrity, duties to relevant bodies, and responsibilities to the profession.

Our projects are executed with careful consideration of the ethical issues that will be encountered during the project.

In line with the BCS requirements for the public interest, we offer a variety of options when filling in personal information, such as gender, sexual orientation, marital status, religion, etc., to meet the legitimate claims of everyone to use the site without discrimination (figure 3.7). Every registered user can make legitimate comments in posts, and any discriminatory comments can be made by contacting us via official email and we will delete the post.

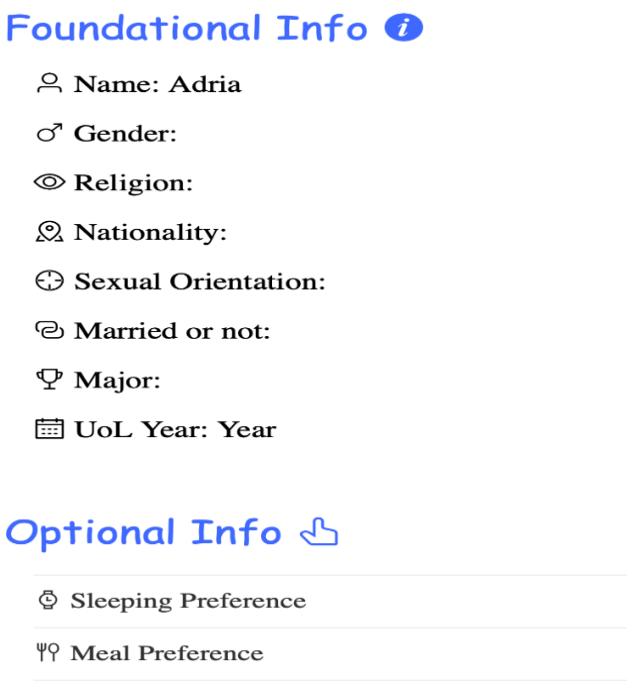


Figure 6: Personal information

Beyond the first time, we respect different points of view and listen to honest criticism of the work of different users and improve it through the official contact channels provided. The main contact channel is email, and the email address is provided on the ABOUT US page (Figure 7).

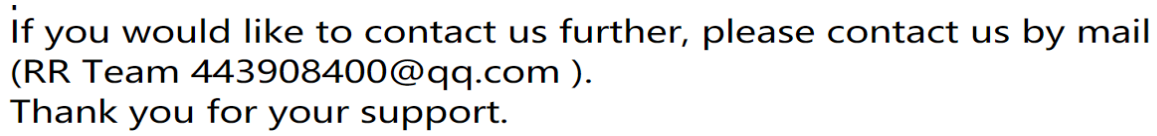


Figure 7： Contact us

Because of the requirements of the General Data Protection Regulation [2], the storage of personal data needs to be handled securely to prevent unauthorized or unlawful processing. The team also tries to ensure the legal rights of third parties using a password registration system and the secure storage of user information.

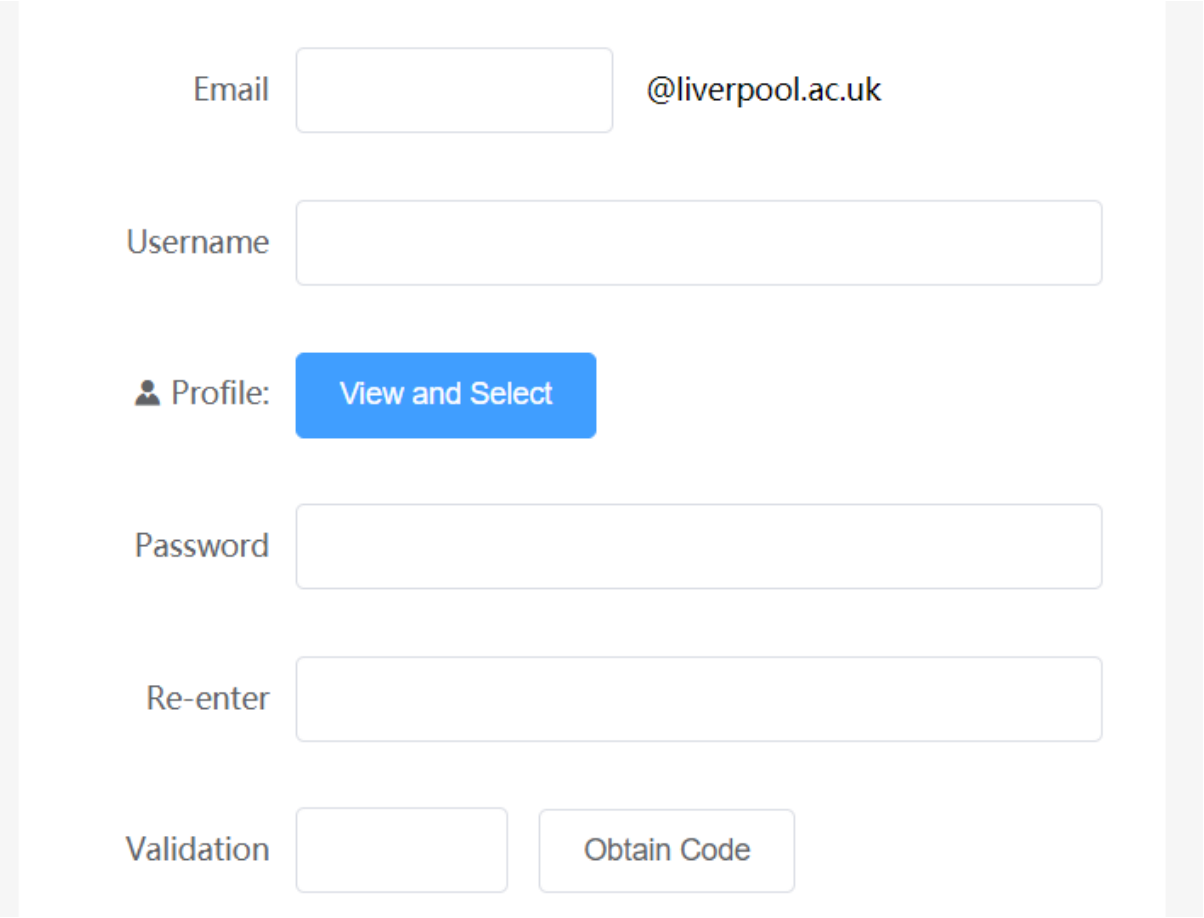


Figure 8： Login and password settings

To secure the account, the users are expected to create the account under a secure domain of @liverpool.ac.uk that will help to authenticate will the university email and if the username was not created by the university then the account cannot be created successfully and you will be requested to visit the admin so for your university mail setup. Upon registration, an OTP pin will be sent to your university mail for verification and authentication. The link has a pin that can be confirmed or a link that will redirect to the Roommate recruitment home page showing a successful login. If the pin or the link is not clicked in 60 seconds it expires and you will be forced to request another pin for verification.

Bibliography

[1]*Bcs.org*. Accessed: May.12,2021. [Online]

Available:
https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/

[2] *General Data Protection Regulation.*(2018).Accessed: May.12,2021. [online]

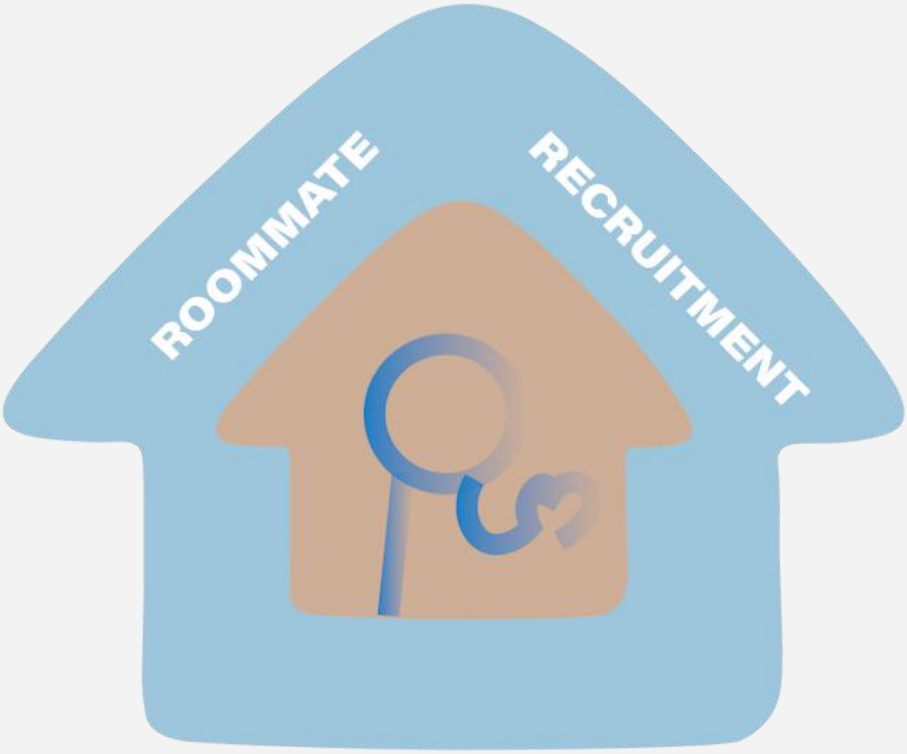
Available: https://gdpr-info.eu/

[3]C. Sasaki, "Effect on the traditional colors of Japan gave Japanese aesthetic study - focusing on the 'Wabi', and 'Sabi' -", *Journal of Korean Traditional Costume*, vol. 16, no. 1, 2013. Available: 10.16885/jktc.2013.04.16.1.91.

[4]K. Rose, "Book Review: Middle Managers in Program & Project Portfolio Management: Practices, Roles & Responsibilities", *Project Management Journal*, vol. 37, no. 1, pp. 82-82, 2021. Available: 10.1177/875697280603700108.

[5]C. Scheitle, "Google's Insights for Search: A Note Evaluating the Use of Search Engine Data in Social Research*", *Social Science Quarterly*, vol. 92, no. 1, pp. 285-295, 2011. Available: 10.1111/j.1540-6237.2011.00768.x.

Design



Team 60 :

Liheyu.Zhang Han.Jiang Zhicheng.Wang

Jiaming.Zhang Yuqi.Xiao Zexun.Lan

Content

1. Introduction

1.1 Purpose

The purpose of this design document is to provide a guidance of the RoommateRecruitment(RR) website development, in order to assist engineers in forming an explicit workflow. RR is the website that allows the target group in need to choose preferred roommates.

1.2 Definition

RR-Roommate Recruitment

1.3 Scope

RR can view the basic information of the user, such as gender, age, professional course, and grade, as well as the description of their personality, religious beliefs, hobbies, and other valuable information, so as to make a choice after understanding.

RR will replace scattered chat groups on the Internet and reduce user time costs by filtering repetitive and inefficient information through a single system.

RR encourages users to communicate via email instead of using the chat room system, thereby reducing unnecessary communication time.

RR enhances the authenticity of the account by restricting email suffixes and other means.

1.4 Document Overview

This document includes class diagram, entity-relationship diagram, algorithm draft(pseudocode), UI prototypes, sequence diagram and relative description, for the sake of accurate implementation of the system according to the requirements. This document specifies validation and evaluation guidance for developers to implement functionality of the website.

This document is aimed at software engineers, module assistants and leaders, offering high-quality guide description.

1.5 System Environment

Development Apps: WebStorm, VSCode

Development Environment: Vue.js, Node.js, Egg.js, Ajax

Diagrams: Visual Paradigm, MySQL Workbench

Database: MySQL

Test App: Postman

1.6 Design Approach

1.6.1 Architecture Design

We divide the architecture into data-service layer, application-service layer, and client layer. This is also called three-tier architecture.

1.6.2 Data Flow Design

The data flow of the SRS is mostly internet-based. Data is sent to and retrieved from a MySQL database via the user's interaction with the application layer.

1.6.3 User Interface Design

We follow the Model-View-Controller (MVC) architecture rules in our user interface design, which means the underlying data in the system and users' operations are separated.

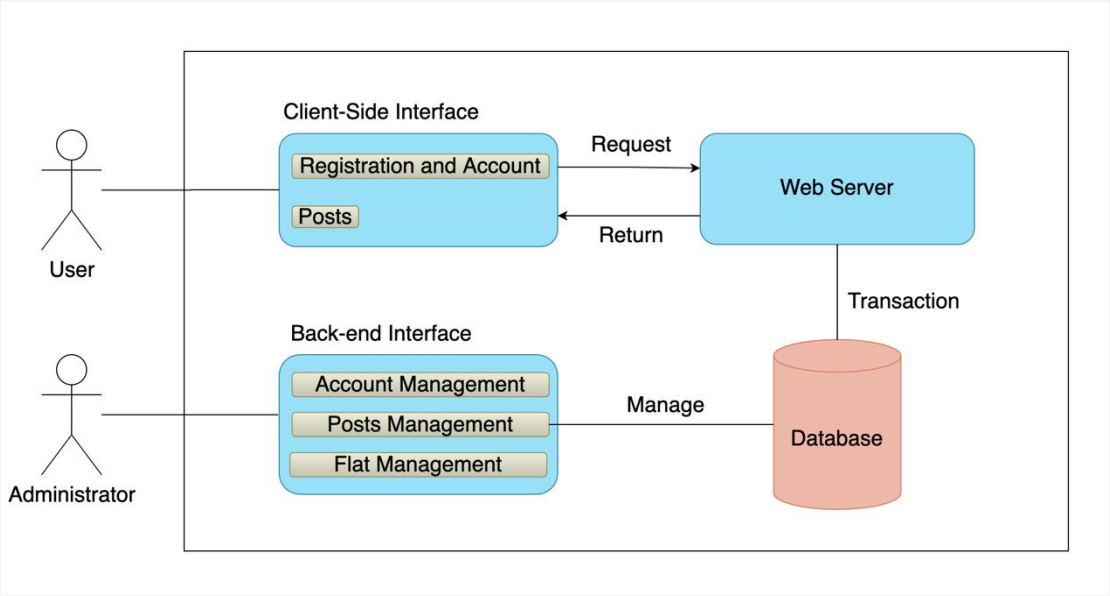
We will provide several user interface prototype drafts, which presents the basic structure of the interface, such as navigation bar and content filter. Meanwhile, they emphasize the operating regulations that users should comply with.

2. Requirements Traceability

Requirement	Description	Design Reference
REQ1, 2, 3, 4	Login & Register	§3.4.1, §3.5, §4.3, §4.4, §6.3.5, §6.5
REQ5, 6, 7, 8, 9, 10	Post & Management	§3.4.2, §3.5, §4.3, §4.4, §4.5, §5, §6.3.1, §6.3.2, §6.3.3, §6.3.4, §6.4
REQ11	Join a Room Group	§3.5, §4.3, §4.4, §6.3.3
REQ12	Edit User Info	§3.5, §4.3, §4.4, §4.5, §6.3.6

3. Architectural Design

3.1 Boundary Diagram



3.2 System Architecture

We follow the Model-View-Controller (MVC) architecture rules in our user interface design, which means the underlying data in the system and users' operations are separated.

The controller handles user actions (such as request results, browse listings), and the view is what the user will see from the Java client or web page. The model retrieves data and functions from the server and is used by the component to display the data.

3.3 Three-tier Architecture

We divide the architecture into data-service layer, application-service layer, and client layer. This is also called three-tier architecture. This architecture improves data security and stability by avoiding direct access to the data-service layer by the client layer. Besides, we can change various parts of the system without reworking the entire application, which improves the reusability of the application.

For example:

Presentation tier: Website pages (MySQL Workbench for Developer)
Logical tier: PHP / Java (or Python)
Data tier: MySQL server
The website can get the post list or post detail through the API by sending the HTTP request to the server: POST/GET.
The server will then return a JSON or XML file back to the client.

3.4 Function Modules

According to different requirements of structure and functions of the website, we split the website into 2 modules. Now the details of both methods are explained as follows.

3.4.1 Registration and Account

This module of the website deals with account details and registration of all users as well as logging in and out of the system.

3.4.2 Posts

This module of the website deals with any post related activities, which includes searching specific post items, releasing a new post, and managing existing posts.

3.5 Modules Interface

3.5.1 Registration and Account

Account Registration

This interface limits the users to register by email address ending with "liv.ac.uk" and provides an area to enter the verification code sent to this email. Once the verification passes, the user should set a password and get the authorization to access this website.

Account Management

This interface presents the current personal details of the users. These details are editable by clicking the “edit” button to update the users’ personal information.

3.5.2 Post

Main User Interface

The main interface of our website consists of various parts. Users can easily proceed to login and post functions in the navigation bar. From the filter area, users can choose preferred conditions to view the satisfying posts. The following area displays the existing posts by our recommendation algorithm.

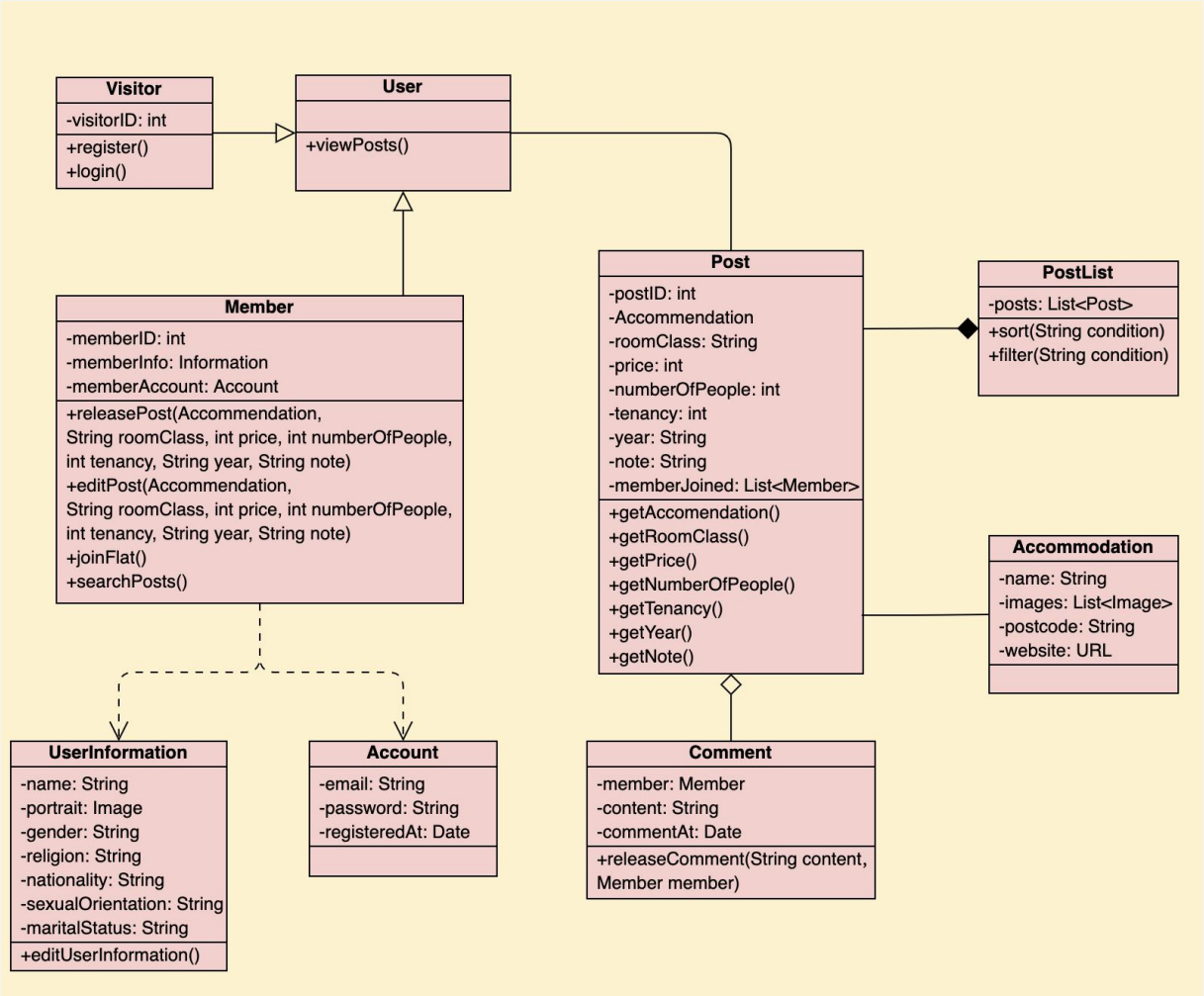
Post List

After filtering, posts that satisfy the user's preference will be presented in a list. On the left side, there is a vertical sidebar containing chosen conditions. Users could update their choices in the sidebar by clicking other conditions.

Post Detail Page

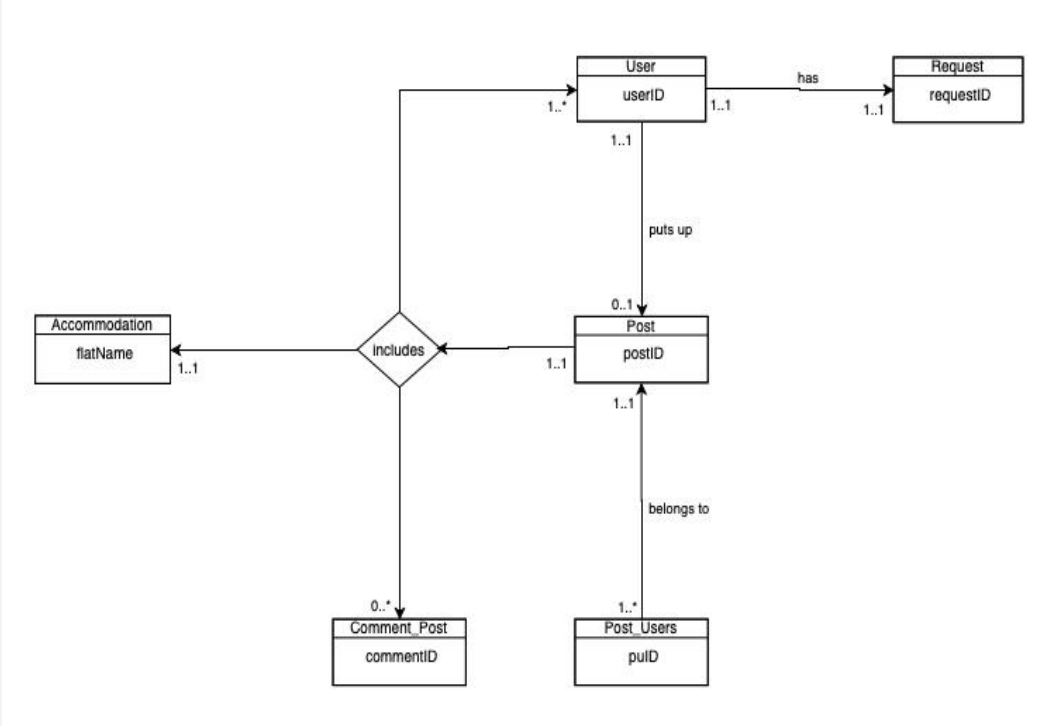
This page displays the details of the post including information of flat and roommate, as well as its comments. Besides, the owner of this post could edit post details by clicking the “edit” button. Users could join the flat by clicking the blank “join” button.

3.6 Class diagram



4.Database Design

4.1 Global Logical Data Model (ER)



4.2 Data Dictionary

Entity	Attributes	Description	Data type and Length	Key	NULL	Multi-Valued
User	userID	The unique ID to identifier the user	int(11)	PK	No	No
	email	The email address of the user	varchar(50)	AK	Yes	No
	username	The user name determine by user	varchar(50)		Yes	No
	password	The password of the user	varchar(225)		Yes	No
	gender	Gender of the user	enum('Male','Female')		Yes	No
	profile	the profile photo of the user	int(11)		Yes	No
	major	the major of the user	varchar(30)		Yes	No
	year	user's grade	int(11)		Yes	No
	religion	the religion of the user	varchar(30)		Yes	No
	nationality	user's nationality	varchar(50)		Yes	No
	sexualOrientation	the user's sexual orientation	varchar(50)		Yes	No
	marriedOrNot	user's marital status	int(11)		Yes	No
	sleepingPreference	the user's sleep preferences	varchar(50)		Yes	No
	mealPreference	the user's meal preferences	varchar(50)		Yes	No
	user_note	A description of the user that is shown on his/her profile	varchar(200)		Yes	No
Post	postID	the unique ID to identifier the user	int(11)	PK	No	No
	ownerID	the unique ID of the post owner	int(11)	FK	No	No
	post_note	the description of the post which the user want to remark	varchar(2000)		No	No
	post_date	time the post was created	varchar(30)		No	No
	vacancy	fill the picture	int(11)		No	No
	capacity	number of people in the room	int(11)		No	No
	flatName	the name of the flat	varchar(30)	FK	No	No
	price	the price of rhe flat	int(11)		No	No
	tenancy	room lease length	int(11)		No	No
	starTime	check-in time	int(50)		No	No

Post	postID	the unique ID to identifier the user	int(11)	PK	No	No
	ownerID	the unique ID of the post owner	int(11)	FK	No	No
	post_note	the description of the post which the user want to remark	varchar(2000)		No	No
	post_date	time the post was created	varchar(30)		No	No
	vacancy	fill the picture	int(11)		No	No
	capacity	number of people in the room	int(11)		No	No
	flatName	the name of the flat	varchar(30)	FK	No	No
	price	the price of rhe flat	int(11)		No	No
	tenancy	room lease length	int(11)		No	No
	starTime	check-in time	int(50)		No	No
Accommodation	flatName	the name of the flat	varchar(30)	PK	No	No
	image1	the image of the accomodation	varchar(100)		No	No
	image2	the image of the accomodation	varchar(100)		No	No
	image3	the image of the accomodation	varchar(100)		No	No
	image4	the image of the accomodation	varchar(100)		No	No
	image5	the image of the accomodation	varchar(100)		No	No
	image6	the image of the accomodation	varchar(100)		No	No
	url	the url of the new web	varchar(2083)		No	No
	postcode	the posrcode of the accomodation	varchar(10)		No	No
	address	the address of the accomodation	varchar(100)		No	No
CommentPost	commentID	the unqiue ID of the coment	int(11)	PK	No	No
	postID	the unqiue ID of the post	int(11)	FK	No	No
	userID	the ID of user who comments this post	int(11)	FK	No	No
	context	the commment on the post	varchar(2000)		No	No
	date	Date when the comment is upload	datetime		No	No
Post_Users	PostID	the unqiue ID of the post	int(11)	PK	No	No
	UserID	the unqiue ID of the user	int(11)	FK	No	No
	puID	the unqiue ID if the post_user	int(11)		No	No
Request	postID	the unique id of the post	int(11)	PK	No	No
	ownerID	the owner's ID	int(11)		No	No
	userID	the user's ID	int(11)		No	No
	requestID	the unique of the group	int(11)		No	No

4.3 Logical table design

Accommadation(flatName,image1, image2, image3, image4, image5, image6, url, postcode, address)
PRIMARY KEY (`flatName`)

Comment_Post(commentID, postID, userID, content, date)
PRIMARY KEY (`commentID`)
CONSTRAINT `fk_postID` FOREIGN KEY (`postID`) REFERENCES `post` (`postID`),
CONSTRAINT `fk_userID` FOREIGN KEY (`userID`) REFERENCES `user` (`userID`)

Post(postID, ownerID, post_note, post_date, vacancy, capacity, flatName, price, tenancy, startTime)
PRIMARY KEY (`postID`)
CONSTRAINT `fk_flatName` FOREIGN KEY (`flatName`) REFERENCES `accommadation` (`flatName`)
CONSTRAINT `fk_ownerID` FOREIGN KEY (`ownerID`) REFERENCES `user` (`userID`)

Post_Users(postID,userID, puID)
PRIMARY KEY (`puID`)

Request(ownerID, postID, userID, requestID)
PRIMARY KEY (`requestID`)

User(userID, email, username, password, gender, profile, major, year, religion, nationality, sexualOrientation, marriedOrNot, sleepingPreference, mealPreference, user_note)
PRIMARY KEY (`userID`)

4.4 Physical Table Structure

4.2.1 Physical Table Structure

DBDL for User Table

Domain UserID	Fixed length integer length 11		
Domain Email	Variable length character string maximum length 50		
Domain Username	Variable length character string maximum length 50		
Domain Password	Variable length character string maximum length 50		
Domain Gender	Variable length character string maximum length 10		
Domain Profile	Fixed length integer length 11		
Domain Major	Variable length character string maximum length 30		
Domain Year	Fixed length integer length 11		
Domain Religion	Variable length character string maximum length 30		
Domain Nationality	Variable length character string maximum length 50		
Domain SexualOrientation	Variable length character string maximum length 50		
Domain MarriedOrNot	Fixed length integer length 11		
Domain SleepingPreference	Variable length character string maximum length 50		
Domain MealPreference	Variable length character string maximum length 50		
Domain User_note	Variable length character string maximum length 2000		
User (	userID	UserID	NOT NULL
	email	Email	
	username	Name	
	password	Password	
	gender	Gender	
	profile	URL	
	major	Major	
	year	Year	
	religion	Religion	
	nationality	Nationality	
	sexualOrientation	SexualOrientation	
	marriedOrNot	MarriedOrNot	
	sleepingPreference	SleepingPreference	
	mealPreference	MealPreference	
	user_note	Note_user	)
	Primary Key	userNo	
	Alternate Key	email	

DBDL for Accommodation Table

```
Domain FlatName          Variable length character string maximum length 30
Domain Image1            Variable length character string maximum length 100
Domain Image2            Variable length character string maximum length 100
Domain Image3            Variable length character string maximum length 100
Domain Image4            Variable length character string maximum length 100
Domain Image5            Variable length character string maximum length 100
Domain Image6            Variable length character string maximum length 100
Domain Url               Variable length character string maximum length 2083
Domain Postcode          Variable length character string maximum length 10
Domain Address           Variable length character string maximum length 100

Accommodation (  flatName     FlatName     NOT NULL
                 image1       Image1
                 image2       Image2
                 image3       Image3
                 image4       Image4
                 image5       Image5
                 image6       Image6
                 url          Url
                 postcode     Postcode
                 address      Address      )
                 Primary Key  flatname
```

DBDL for Request Table

```
Domain OwnerID           Fixed length integer length 11
Domain PostID            Fixed length integer length 11
Domain UserID            Fixed length integer length 11
Domain RequestID         Fixed length integer length 11

Request (  ownerID     OwnerID     NOT NULL
           postID      PostID      NOT NULL
           userID      UserID      NOT NULL
           requestID   RequestID   NOT NULL)
           Primary Key requestID
```

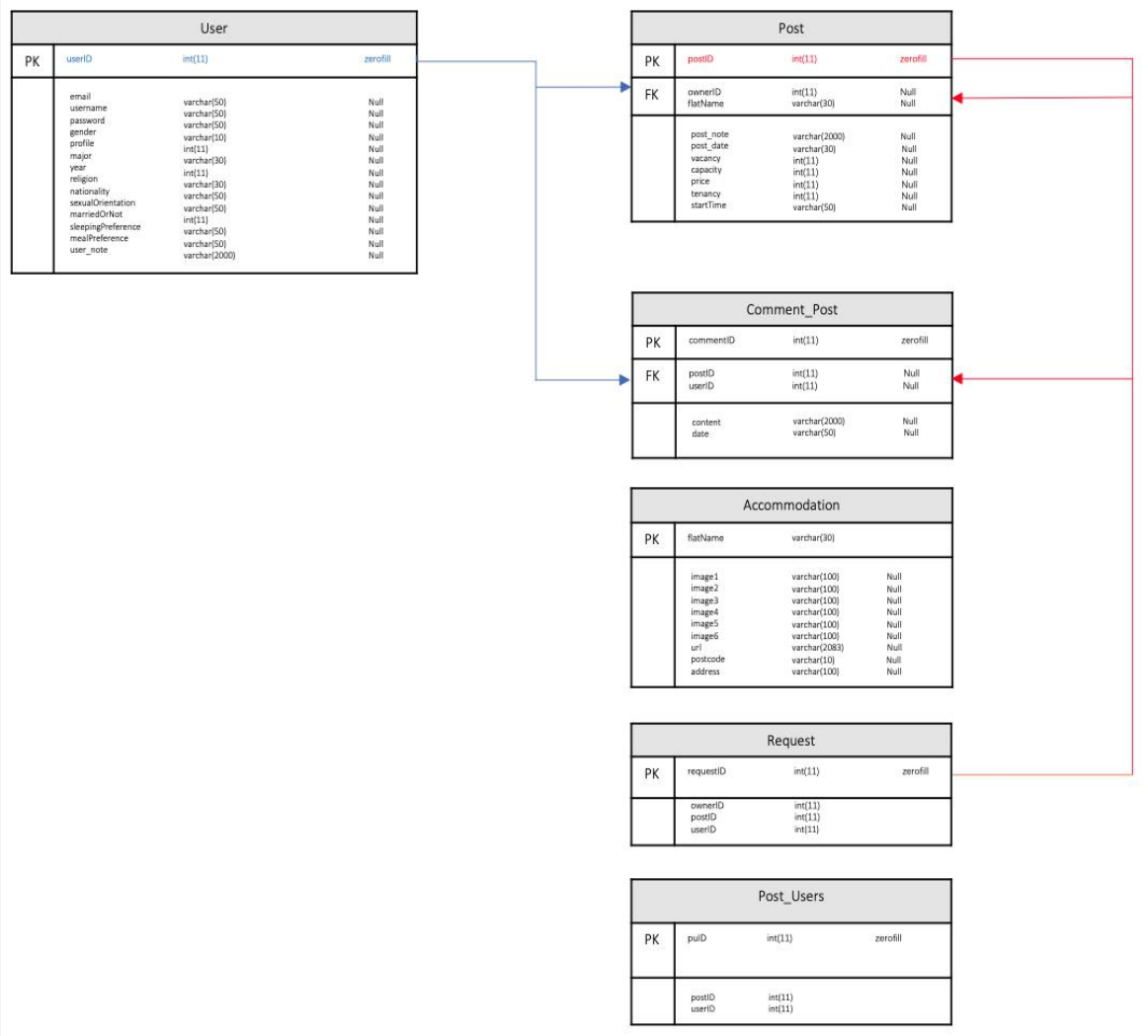
DBDL for Comment_Post Table

```
Domain CommentID        Fixed length integer length 11
Domain PostID           Fixed length integer length 11
Domain UserID           Fixed length integer length 11
Domain Content          Variable length character string maximum length 2000
Domain Date             Variable length character string maximum length 50

Comment_post (  commentID    CommentID                          NOT NULL
                postID       PostID
                userID       UserID
                content      Content
                date         Date                               )
                Primary Key  commentID
                Foreign Key  fk_postID References Post(postID)
                Foreign Key  fk_userID Reference User(userID)
```

DBDL for Post_Users Table

```
Domain PostID           Fixed length integer length 11
Domain UserID           Fixed length integer length 11
Domain PuID             Fixed length integer length 11

Post_Users (  postID       PostID      NOT NULL
              userID       UserID      NOT NULL
              puID         PuID        NOT NULL)
              Primary Key  puID
```

4.4.2 Physical Data Model



4.5 Transaction Matrix

Table/Transaction	(a)				(b)				(c)				(d)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
User																
Post	X	X				X	X					X				
Accommodation	X	X				X	X					X				
CommentPost												X				
Post_Users																
Request													X			
Table/Transaction	(e)				(f)				(g)				(h)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
User	X				X	X	X						X			
Post																
Accommodation																
CommentPost									X	X						
Post_Users									X	X						
Request													X			

I: Insert R: Read U: Update D: Delete Transactions for (a to (h:

a) Create posts

b) Edit posts

c) Delete posts

d) Save posts

e) Create an account

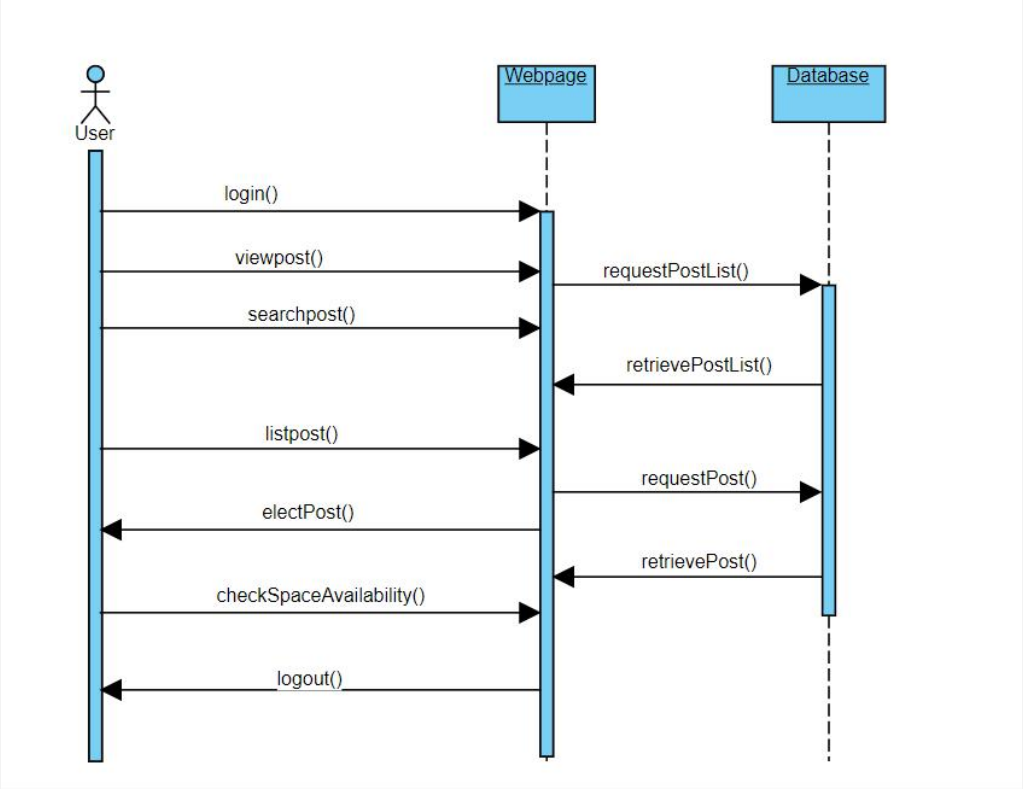
f) Create user profile

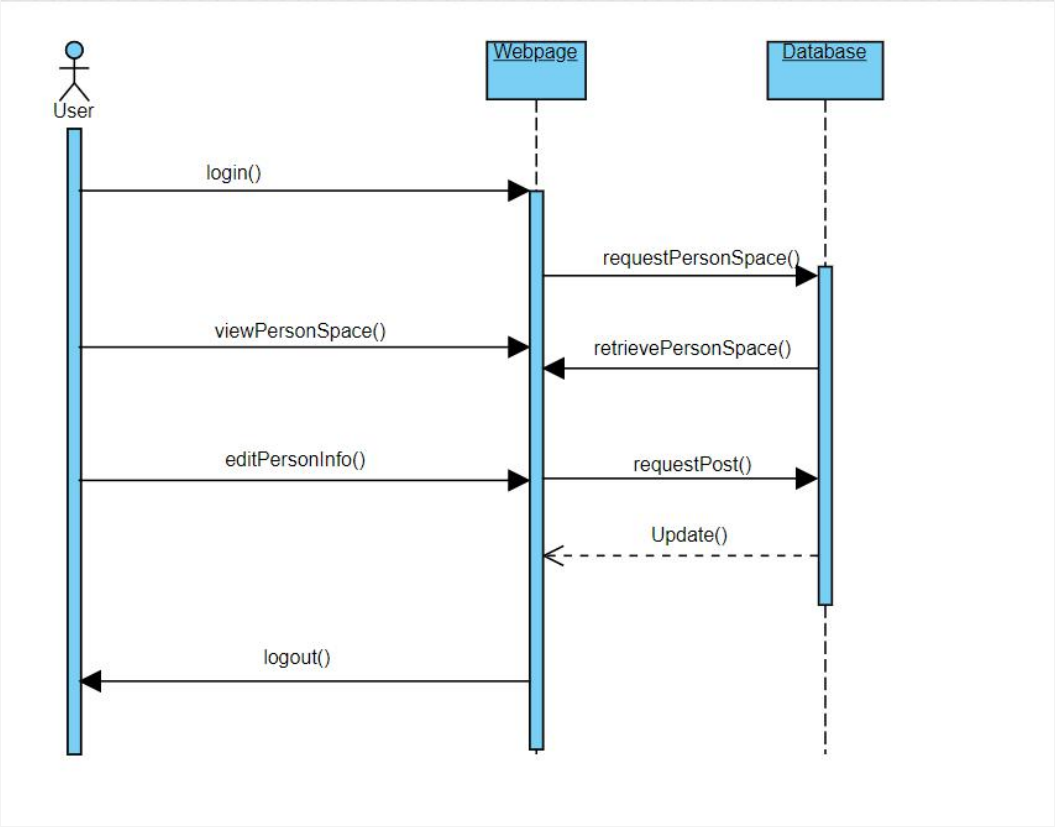
g) Create comments

h) Request

When a post is deleted from the Post table, then it will also be deleted from the CommentPost table and Accomodation table because of the foreign key constraints.

4.6 Sequence diagrams





5. Algorithm Design

5.1 Sort Algorithm

This algorithm is used to provide a default sort regulation for post presentation (recommendation part of home page, post list page). When a user does not choose any sort options, the website will follow this regulation to present post items.
In this algorithm, we decide to use the number of vacant rooms in a flat as the prior sorting condition. For example, if a ten-room flat A has 2 vacant rooms and a six-room flat B has 1 vacant room, then system will calculate:
PA = (10-2) / 10 = 0.8
PB = (6-1) / 6 = 0.83
Obviously, PB is larger than PA. Therefore, the post of flat B takes precedence over flat A.

After that, we consider the price condition when the vacant points are equal. We decide to use the middle position of the price range as a prior sorting condition, then choose a larger one and a smaller one successively. For example, there are 3 flats, flat A is 154 pounds per week, flat B is 140 pounds per week, flat C is 160 pounds per week. After descend sorting, postList is [C, A, B]. Therefore, we choose A first, then A, finally B.

```
'use strict';
const compare = function(prop) {
  return function(obj1, obj2) {
    let val1 = obj1[prop];
    let val2 = obj2[prop];
    if (!isNaN(Number(val1)) && !isNaN(Number(val2))) {
      val1 = Number(val1);
      val2 = Number(val2);
    }
    if (val1 < val2) {
      return -1;
    } else if (val1 > val2) {
      return 1;
    }
    return 0;

  };
};


const Service = require('egg').Service;

class SortService extends Service {

  async defaultSortPostList() {
    try {
      const postList = await this.ctx.model.query('select * from Post', {
        type: 'SELECT',
      });
      postList.sort(await compare( prop: 'vacancy'));
      const finalList = [];
      const tempList = [];
      tempList.push(postList[0]);
      for (let i = 1; i < postList.length; i++) {
        if (postList[i].vacancy === postList[i - 1].vacancy) {
          tempList.push(postList[i]);
        }
        if (postList[i].vacancy !== postList[i - 1].vacancy) {
          if (tempList.length === 1) {
            finalList.push(tempList[0]);
          } else {
            tempList.sort(await compare( prop: 'price'));
            const pos = tempList.length / 2;
            const mid = tempList[tempList.length / 2];
            finalList.push(mid);
            for (let j = 0; j < tempList.length / 2; j++) {
              const left = pos - 1;
              const right = pos + 1;
              if (left >= 0) {
                finalList.push(tempList[left]);
              }
              if (right < tempList.length) {
                finalList.push(tempList[right]);
              }
            }
          }
          tempList.length = 0;
          tempList.push(postList[i]);
          if (i === postList.length - 1) {
            finalList.push(tempList[0]);
          }
        }
      }
      return postList;
    } catch (e) {
      return null;
    }


  }
}
module.exports = SortService;
```

```
'use strict';

const Controller = require('egg').Controller;

class recommendationController extends Controller {
  async index() {
    const list = await this.ctx.service.sort.defaultSortPostList();
    for (let i = 0; i < 6; i++) {
      const image = await this.ctx.model.query('select image1 from Accommadation where flatName= :flatName', {
        replacements: { flatName: list[i].flatName },
        type: 'SELECT',
      });
      list[i].img = image[0].image1;
    }
    this.ctx.body = [
      list[0], list[1], list[2], list[3], list[4], list[5],
    ];
  }
}
module.exports = recommendationController;
```

5.2 Filtering algorithms

First create a hash table as the blank root node of the structure, then iterate through the sensitive word lexicon to get a sensitive word string, and finally iterate through the sensitive word string to get a currently traversed character. Finally, find whether the tree structure already contains the current traversal character, if it does, go directly to this node that already exists in the tree structure, and then continue traversing down the character.

```
const Controller = require('egg').Controller;

class loadListController extends Controller {
  async create() {
    const select = await this.ctx.request.body.select;
    var posts = await this.ctx.model.query('select postID, flatName, price, vacancy, capacity from Post', {
      type: 'SELECT',
    });

    posts.forEach(key1 => {
      flats.forEach(key2 => {
        if (key2.flatName === key1.flatName) {
          key1.img = key2.image1;
          key1.postcode = key2.postcode;
        }
      });
    });
    posts.forEach(postKey => {
      const postCode = postKey.postcode;
      if (postCode) {
        const areaCode = postCode.split(' ');
        postKey.areaCode = areaCode[0];
      }
    });

    var temp = [];
    if (select.name[0]) {
      const names = select.name;
      names.forEach(key1 => {
        posts.forEach(key2 => {
          // eslint-disable-next-line eqeqeq
          if (key2.flatName == key1) {
            temp.push(key2);
          }
        });
      });
      posts = temp;
      temp = [];
    }
    if (select.price[0]) {
      const prices = select.price;
      prices.forEach(key1 => {
        posts.forEach(key2 => {
          // eslint-disable-next-line eqeqeq
          if (key1 == 'Under £100') {
            if (key2.price <= 100) {
              temp.push(key2);
            }
          }
          // eslint-disable-next-line eqeqeq
          if (key1 == '£100-150') {
            if (key2.price > 100 && key2.price <= 150) {
              temp.push(key2);
            }
          }
          // eslint-disable-next-line eqeqeq
          if (key2.price > 150 && key2.price <= 180) {
            if (key2.price <= 100) {
              temp.push(key2);
            }
          }
          // eslint-disable-next-line eqeqeq
          if (key2.price > 180 && key2.price <= 210) {
            if (key2.price <= 100) {
              temp.push(key2);
            }
          }
          // eslint-disable-next-line eqeqeq
          if (key1 == 'Beyond £210') {
            if (key2.price > 210) {
              temp.push(key2);
            }
          }
        });
      });
      posts = temp;
      temp = [];
    }
    this.ctx.body = posts;
  }
}
module.exports = loadListController;
```

The dictionary library here is：

```
with open("./sensitive_words", 'w') as f:
f.write("\n".join( [ "keyword" + str(i) for i in range(0,10000)]))
```

5.3 Password validation algorithms

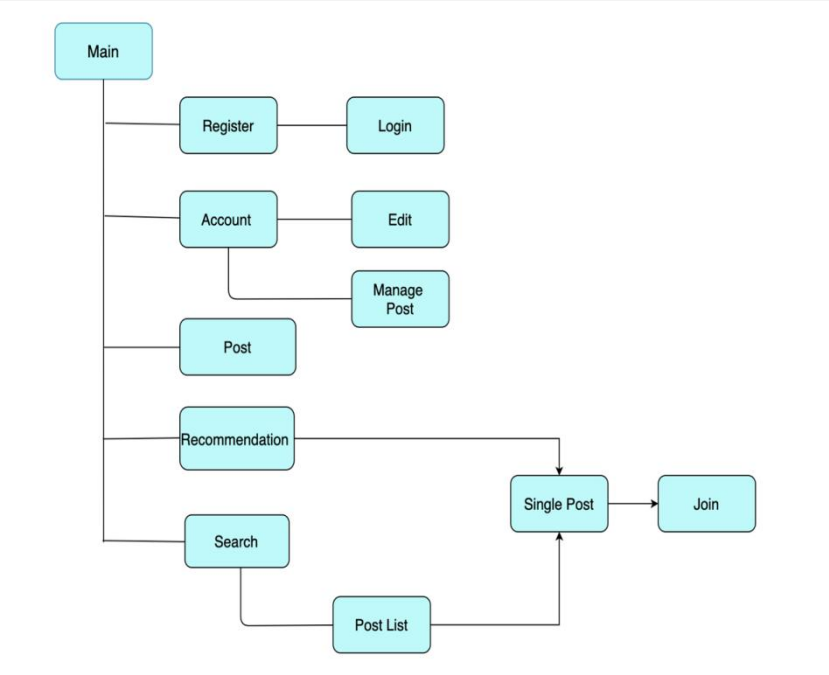
This algorithm is used to verify whether the password set by the users meets the requirements. The regular expression is used to determine whether the password is composed of "a-z", "A-Z" and "0-9" and the length is 8-20. If the above conditions are not met, the error information will be fed back to the users. After the users enter the password for the second time, verify whether it is the same as the first time. If they are different, the system returns error information.

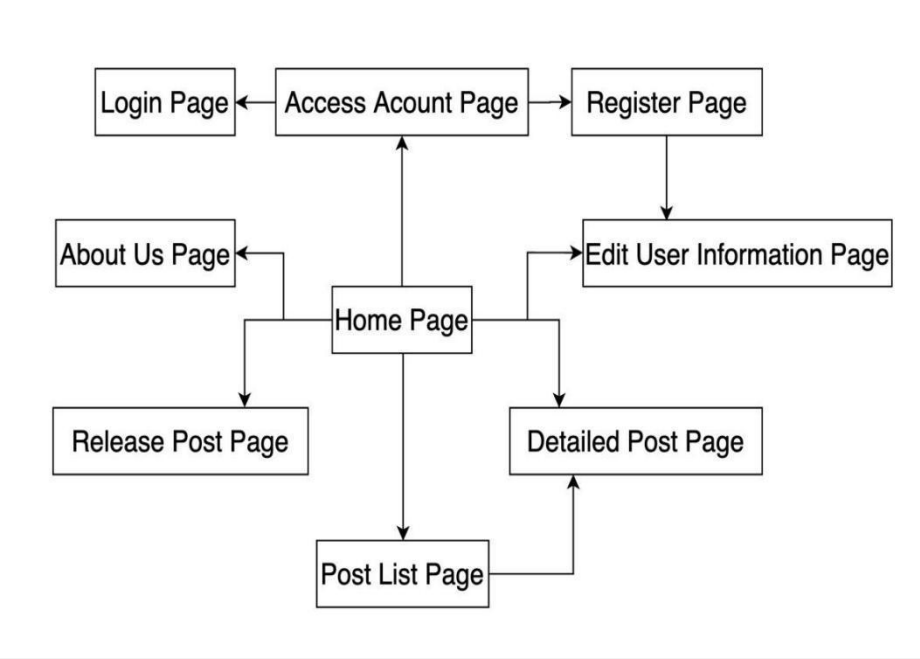
```
var validatePass = (rule, value, callback) => {
  if (value === '') {
    callback(new Error('Please enter password.'));
  } else if(!/^[a-zA-Z0-9-_+@]{8,20}$/.test(value)){
    callback(new Error("Password only contains 8-20 characters, numbers and '_', '-', '+', '@' notations"));
  }
  else {
    if (this.ruleForm.checkPass !== '') {
      this.$refs.ruleForm.validateField( props: 'checkPass');
    }
    callback();
  }
};
var validatePass2 = (rule, value, callback) => {
  if (value === '') {
    callback(new Error('Re-enter password again, please.'));
  } else if (value !== this.ruleForm.pass) {
    callback(new Error('Not consistent.'));
  } else {
    callback();
  }
};
```

6. User Interface Design

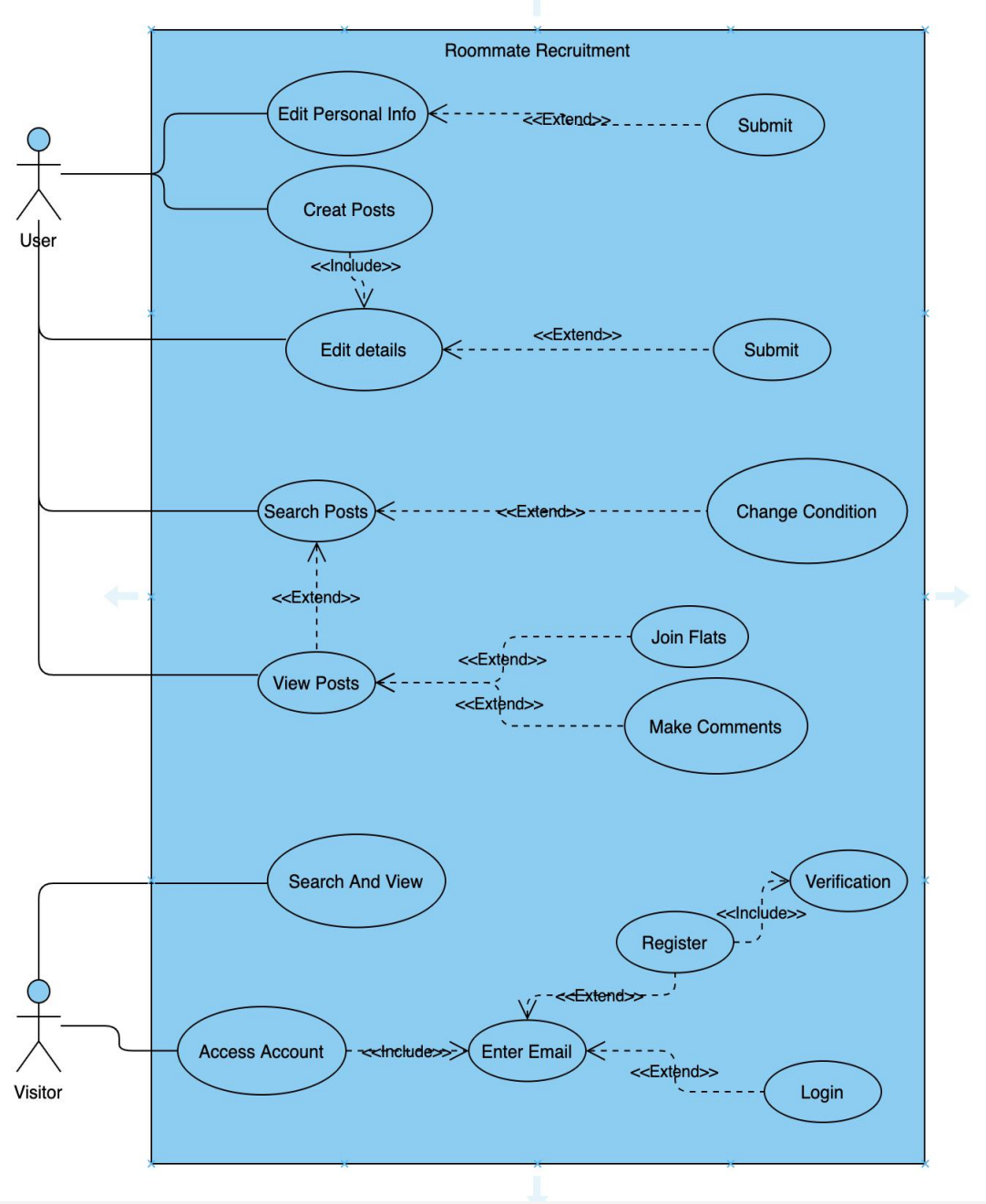
6.1 Structure and Navigation

From the following diagram below, there is an outline of the website. It emphasizes the hierarchical relationships between different pages and logic during the web jumping.





6.2 Use Case Diagram



6.3 User Interface for Use Cases (Specific Pages)

In this phase of the document, we will present several user interfaces prototypes that cover three specific pages: *initial view, post list view, single detailed post view*. We will present the fundamental structure for each page, including the content of the navigation bar, the positions of columns and so on.

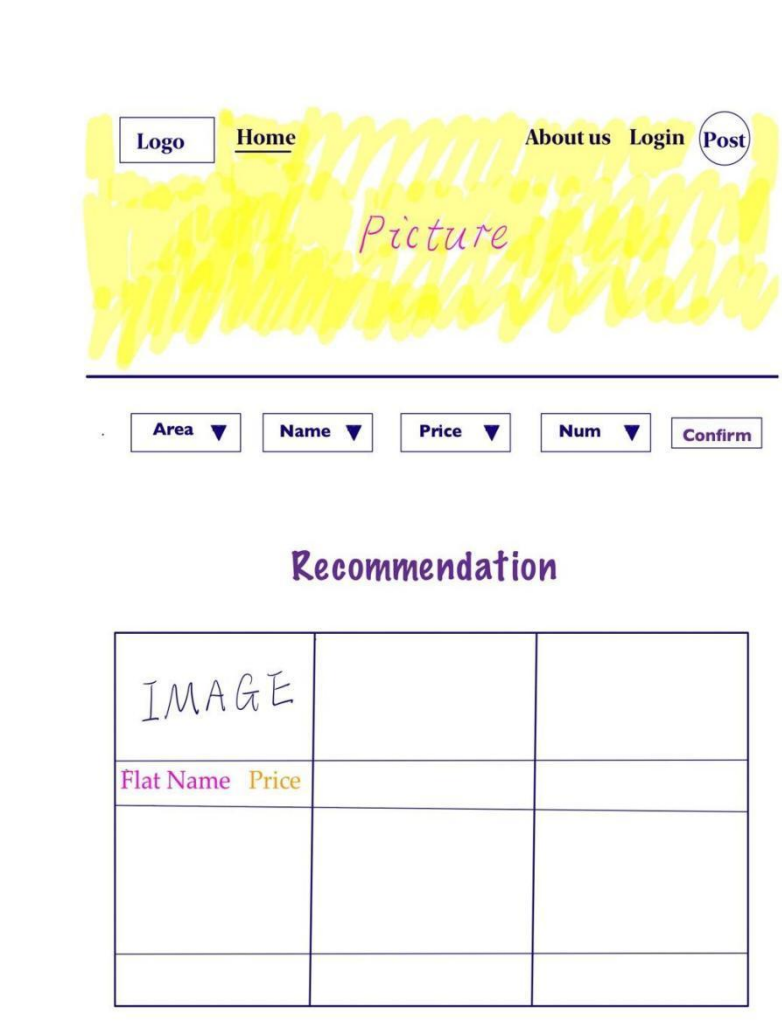
For each user interface, we will provide a particular description and corresponding navigation details, as well as some methods of how to implement certain components.

6.3.1 Page: Initial View

6.3.1.1 Description

Below draft shows the home page of our website which consists of four parts: navigation bar, main cover, filter area and recommended flats. Users can post a new flat recruitment by clicking the “Post” button or search existing flats using filters.

6.3.1.2 User Interface Prototype



6.3.1.3 Navigation

Navigation bar around our website is simple and consistent throughout most screens and pages. Users will be able to use the navigation bar at the top of each page. Functions including logging, registering account, and posting could be applied by clicking corresponding different buttons in the top right corner. The logo of the website is laid in the top left corner which could jump to the home page after clicking.

Under the navigation bar is a high-resolution picture (as the yellow part shown above). Each time the user opens the home page, a random picture which is chosen from a picture set will show on. And after several seconds, the next picture will slide into the page horizontally.

Several filters, at the middle of the home page, are used for navigation to users' preference posts. After choosing different criteria and confirming, the page jumps to the post list view.

At the bottom of the page are six recommended flats by system which include three components: flat name, picture, and price. Users can click the preferred one and then be navigated to a single detailed post view.

6.3.1.4 Issues

The main expected issue in this view is that we are still trying to figure out how to implement the "slide in" effect. If this problem cannot be solved, we will replace it with a static picture.

6.3.1.5 Component Implementation

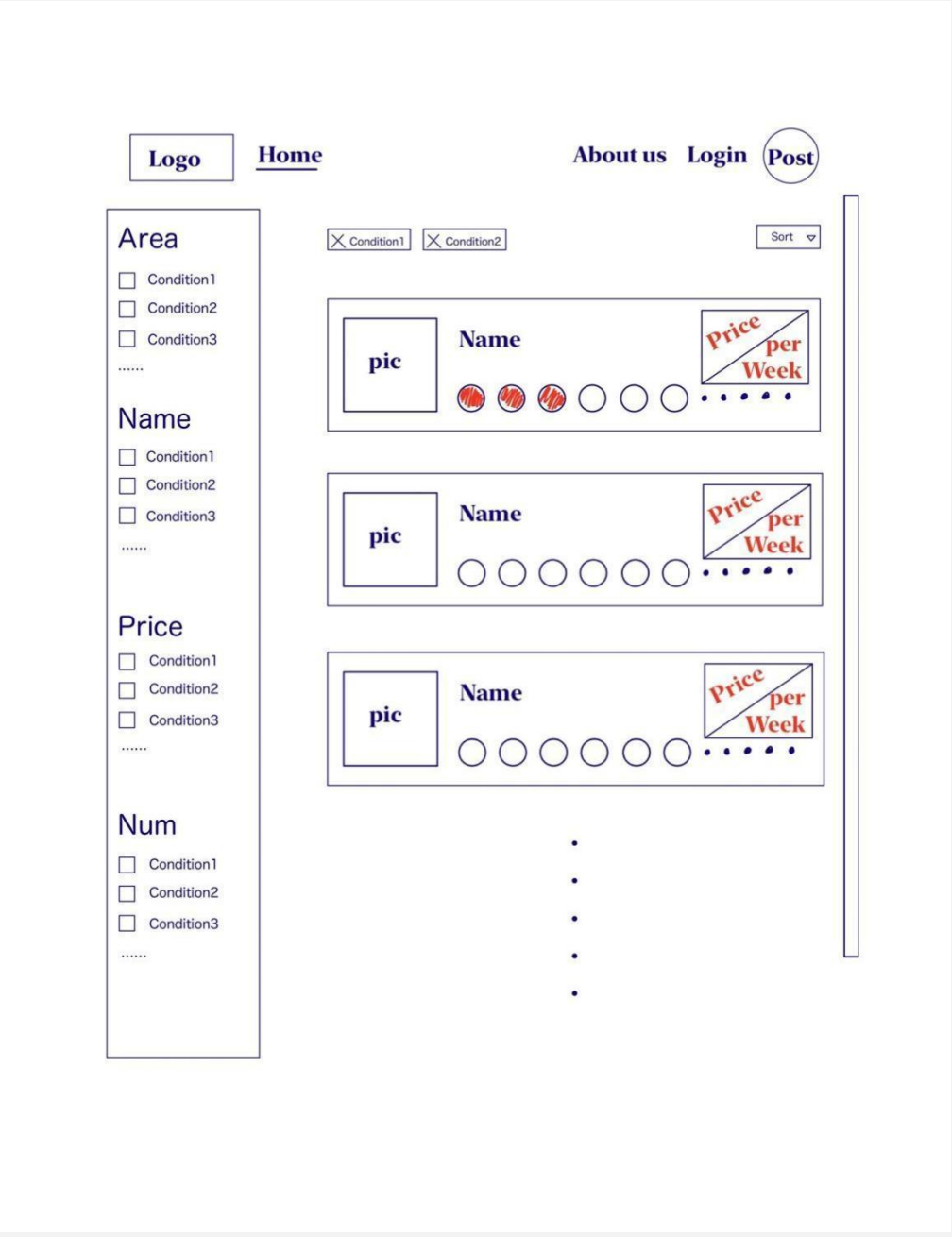
Navigation Bar	---	(box model)
Logo	---	<a><img>
Home/About us/Login	---	<a>
Post	---	<a>
Filter	---	<select></select>
Recommendation	---	<table><tr><td></td></tr></table>

6.3.2 Page: Post List View

6.3.2.1 Description

The main part of the post list page displays all existing flats that match the user's search criteria in the order that the user chooses, such as from low price to high or from low to high the number of people left. The sidebar to the left lists the several types of filters and the values they contain. The values currently selected by the user are highlighted when the page loads.

6.3.2.2 User Interface Prototype



6.3.2.3 Navigation

Navigation bar around our website is simple and consistent throughout most screens and pages. Users will be able to use the navigation bar at the top of each page. Functions including logging, registering account, and posting could be applied by clicking corresponding different buttons in the top right corner. The logo of the website is laid in the top left corner which could jump to the home page after clicking.

Navigation to this page is done by clicking the “confirm” button after the user selects the value corresponding to each filter criteria at the initial page.

From this page, the user can click on each single post to view the details of this flat. The user can also reset the filter at any time by clicking the values of each filter in the left side bar and refreshing this page.

6.3.2.4 Issues

During the development of the Post list page, we may encounter the problem that there are multiple posts with identical conditions after the user selects the filter criteria, we have not found an appropriate algorithm to perform sorting work for the user so that the user can find the most suitable flat the fastest.

6.3.2.5 Component Implementation

List --- <a>

Sidebar --- <input type= “checkbox”>

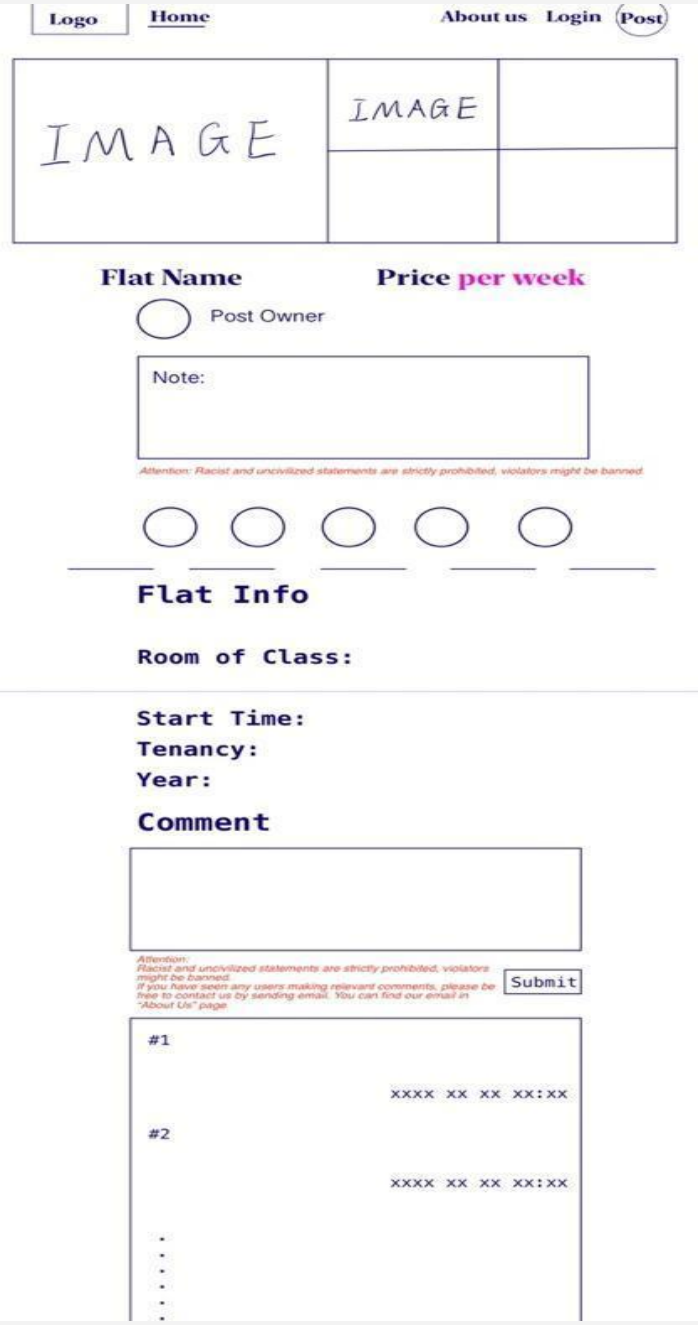
Sort --- <select></select>

6.3.3 Page: Single Detailed Post View

6.3.3.1 Description

The single detailed post page allows authorized users to view specific information about the flat, such as high-resolution pictures, location, post code, price per week, number, and personal details of current roommates like gender, major, living habits and requirements for other roommates. Meanwhile, the post owner’s note is available to view. Start Time and year determine the year and month in which the user will start to live in the apartment. The Tenancy shows the legal period for the user to live in the apartment after the Start Date. The comments of this flat are listed below the roommate information. A warning against racial discrimination and false information is written at the bottom of the comment input box, next to the submit button. It also instructs users how to contact us when they encounter bad comments. The user then determines whether this flat and these roommates are suitable. The user can choose to join this flat by clicking the empty portrait or directly back to the previous page to select a more appropriate flat.

6.3.3.2 User Interface Prototype



6.3.3.3 Navigation

Navigation bar around our website is simple and consistent throughout most screens and pages. Users will be able to use the navigation bar at the top of each page. Functions including logging, registering account, and posting could be applied by clicking corresponding different buttons in the top right corner. The logo of the website is laid in the top left corner which could jump to the home page after clicking.

Navigation to this page is done by clicking a single post from the list of existing posts after a user filters through the preferred criteria. The

information of each roommate currently in this flat can be listed by clicking the corresponding icon.

6.3.3.4 Issues

On this page we may be faced with a picture display area that cannot achieve the best use experience due to the size, clarity, resolution, compatibility, and other factors of the pictures.

6.3.3.5 Component Implementation

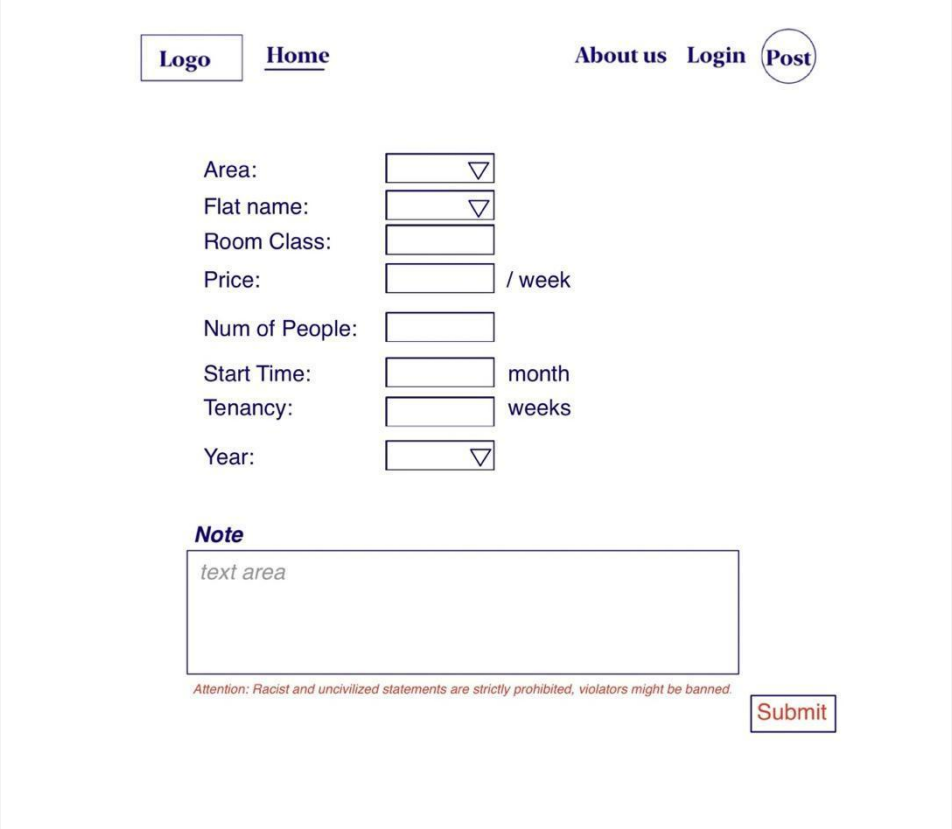
Image	---	<img>
Note/Comment	---	<textarea></textarea>
Join	---	<a>

6.3.4 Page: Post & Edit View

6.3.4.1 Description

This page allows users to fill in information about preferred accommodation to post or edit to find suitable roommates. The user selects the “area” indicates the first part of postcode, “Flat name” to determine a specific accommodation starting with the postcode, “year” which indicate the year of occupancy and the “number of people” that can be accommodated in this flat from our database through a drop-down menu. Other information such as “room class,” “price,” "Start Time" for the month of residence and “tendency” to indicate the number of weeks of rental will be manually entered by the user in the corresponding area. Finally, the user writes down the expected criteria for roommates within 100 words in the “note” area, such as preferred bedtime and daily schedule, eating habits and other information that we did not list. In addition, at the bottom of the text box and on the left side of the submit button, users can see a warning about the prohibition of racial discrimination information and bad information. We use this method to ensure that users get a good experience.

6.3.4.2 User Interface Prototype



6.3.4.3 Navigation

Navigation bar around our website is simple and consistent throughout most screens and pages. Users will be able to use the navigation bar at the top of each page. Functions including logging, registering account, and posting could be applied by clicking corresponding different buttons in the top right corner. The logo of the website is laid in the top left corner which could jump to the home page after clicking.

Navigation to this page is achieved by the user clicking the “post” button on the home page. After all the information is filled in, the user submits this post by clicking the “Submit” button and jumps to the “Single Detailed Post View” page. This roommate recruitment post is logged in the database and can be found on the “Post List View” page.

6.3.4.4 Issues

We tried to gain a balance between personalization and standardization. It is difficult for us to decide which attributes and preferences allow users to edit in person, and which ones are listed by the developer in advance and then given to users to choose.

6.3.4.5 Component Implementation

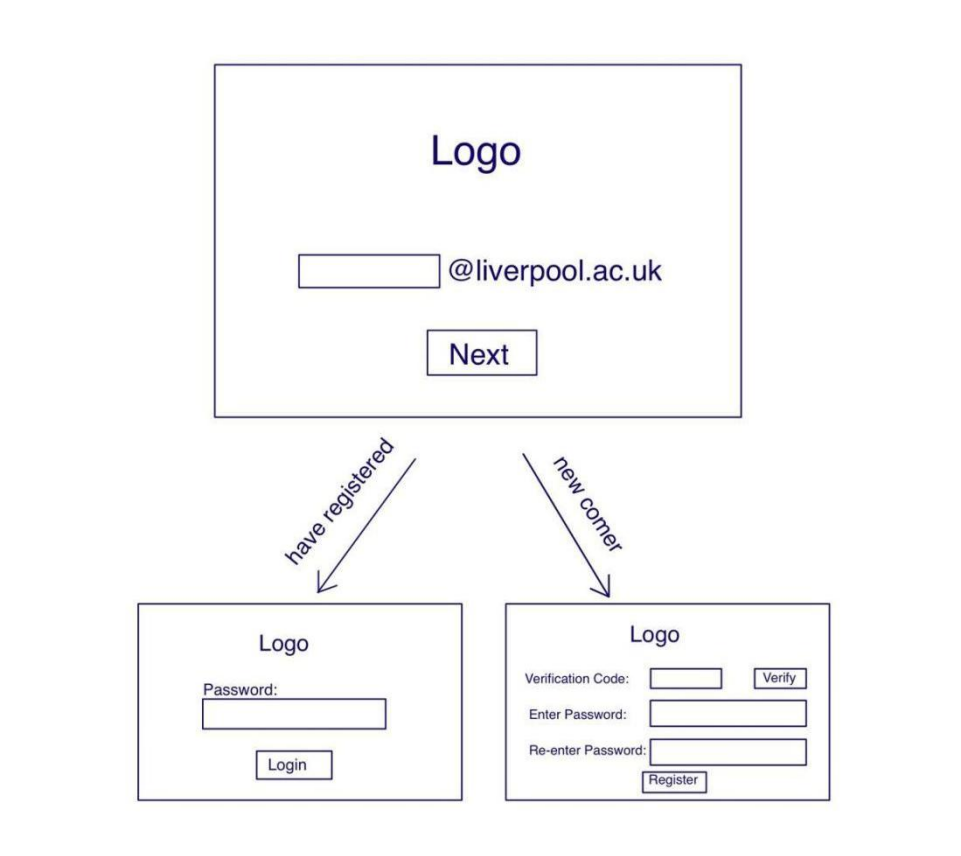
Area/Flat Name/Year	---	<select></select>
RoomClass/Price/Num/Tenancy	---	<input type= “text”>
Note	---	<textarea></textarea>
Submit	---	<input type= “submit”>

6.3.5 Page: Register & Login View

6.3.5.1 Description

This page is used to realize the user's login and registration function to carry out the subsequent post and flat search operation. This page is divided into three sub-pages. On the first page, users are asked to fill in a valid University of Liverpool email address. Users on the registration page are required to fill in the verification code issued by the system to the email address and set a password for their own accounts and confirm. In the login page, the users should enter the password set by themself to log in this website.

6.3.5.2 User Interface Prototype



6.3.5.3 Navigation

Navigation bar around our website is simple and consistent throughout most screens and pages. Users will be able to use the navigation bar at the top of each page. Functions including logging, registering account, and posting could be applied by clicking corresponding different buttons in the top right corner. The logo of the website is laid in the top left corner which could jump to the home page after clicking.

Navigation to this page is done by clicking the “login” button at the top of the home page. The user enters an email address with the suffix "@liverpool.ac.uk" and clicks "Next" to jump. If the email address is not registered, it will be redirected to the registration page and a verification code will be sent to the email address. After the user fills in the verification code, sets and confirms the password and clicks “Next,” the user is redirected to the “login” page. If the mailbox entered in the first page has been registered before, it will also be redirected to the “login” page directly. Users can log in to

their account after entering their password and clicking "Login". At this time, they will be redirected to the home page of the system.

6.3.5.4 Issues

The issue we may encounter in this page is that the user entered the wrong username in the first half of the mailbox. We may also encounter the wrong verification code entered by the users during registration, which will increase the workload of the server when users get the verification code again through the mailbox.

6.3.5.5 Component Implementation

Email/Verification Code	---	<input type= “text”>
Password	---	<input type= “password”>
Next/Login/Register	---	<input type= “submit”>

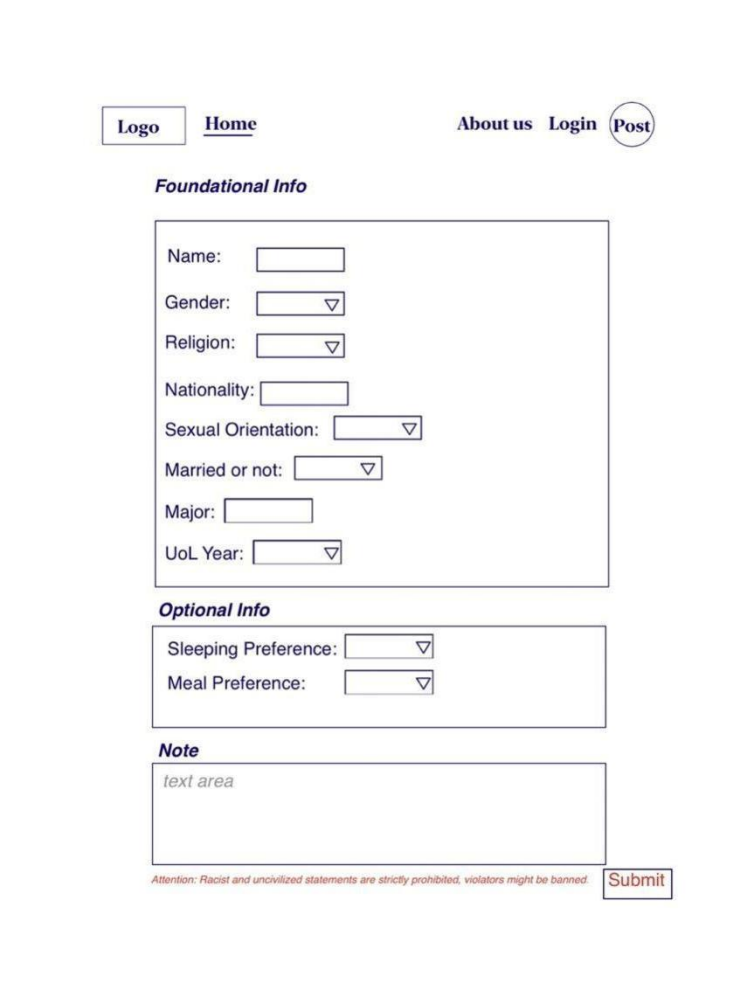
6.3.6 Page: User Information & Edit View

6.3.6.1 Description

In this page, Users are required to fill their detailed information which is used to let others know them. The page is separated into three parts, fundamental information, optional information, and notes.

Fundamental information is located at the upper part of the page. Users need to fill their basic personal details including “name,” “gender,” “major,” “year,” “religion,” “nationality,” “sexual orientation” and “marital status.” In the optional section, there are several options about personal living habits to choose which is aimed to assist users to recruit a more appropriate roommate. Notes section allows users to write more details about themselves which is not listed above within 100 words.

6.3.6.2 User Interface Prototype



6.3.6.3 Navigation

Navigation bar around our website is simple and consistent throughout most screens and pages. Users will be able to use the navigation bar at the top of each page. Functions including logging, registering account, and posting could be applied by clicking corresponding different buttons in the top right corner. The logo of the website is laid in the top left corner which could jump to the home page after clicking.

After filling above information, users can submit these to the system. Then the page refreshes automatically and switches to information display mode.

6.3.6.4 Issues

We tried to gain a balance between personalization and standardization. It is difficult for us to decide which attributes and preferences allow users to edit in person, and which ones are listed by the developer in advance and then given to users to choose.

6.3.6.5 Component Implementation

Name/Nationality/Major --- <input type= “text”>

Gender/Religion/SexualOrientation/Married or not/ UoL Year/Preference --- <select></select>

Note --- <textarea></textarea>

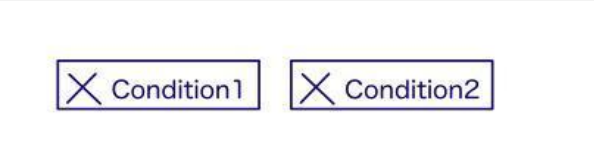
Submit --- <input type= “submit”>

6.4 Search Filter Bar & Sidebar

6.4.1 Search Filter Bar

As the draft above, the main searching functions are implemented by such a filter group. User could click any one of them, then the drop-down filter will present several relevant options that the user can choose, under this condition (e.g., under Area condition, there exists “L1”, “L2”, “L3”, …). After finishing choosing, the user can just tap the “Confirm” button, as a consequence, the web will jump to the “*post list*” page.

6.4.2 Sidebar



Sidebar is more like an auxiliary tool for users when they have already entered the “post list” view but want to change some filter conditions.

Like the example on the left side, users can click the blank boxes to change or add conditions. Any changes will be shown in the post list on the right immediately. Moreover, users can get to know what conditions are chosen currently and can click “x” to delete that condition, just like the picture below.

6.5 User Validation methods

In order to avoid errors or malicious registration, which caused a large amount of invalid data to enter the server to crash, and to ensure that the user group is University of Liverpool students to ensure the authenticity of the posts, we use existing methods to verify user input. Several methods and principles are described below.

Example 1 – "E-mail address suffix are limited to '@liverpool.ac.uk'"

Validation Rule: Users must enter an E-mail address which suffix is '@liverpool.ac.uk.'
Validation Method: Regular expression

Using a regular expression method to match characters after "@" is "liverpool.ac.uk" or not. This method is aimed for identifying registering users as a UoL student. Only passing this verification, the user can go to the next step to verify Email availability.

Example 2 – "Verification code must match the email address."

Validation Rule: Users must correctly enter the validation code sent to the email address.

Validation Method: Php Method mail ()

By using the Php method "mail ()," the verification code is generated in the background and stored in the data table(auth). The email address and the verification code correspond one to one, and the verification code will be sent to the user's email. The user fills in the verification code received, and the background checks whether the verification code matches the email address.

7. Business Rules and Risk Assessment

7.1 Risk Assessment

7.1.2 Major Challenges

The front and back ends do not know each other's content.

Some features of the site are not available.

The database is difficult to process big data, and each form needs further processing.

7.1.3 New Skills Required

Members will be more familiar with HTML, CSS, JavaScript, PHP, and MySQL.

Communication between front-end and back-end members will become more proficient.

7.2 Business Rules

7.2.1 Privacy Policy and Terms

Privacy policies and related provisions shall be in place to protect the interests of users. Users have the right to know how their personal information will be handled using the community system. More importantly, no data should be provided to third parties selling user information.

7.2.2 General Rules

Comment should be scrutinized.

Only students from UoL can make accounts on the website.

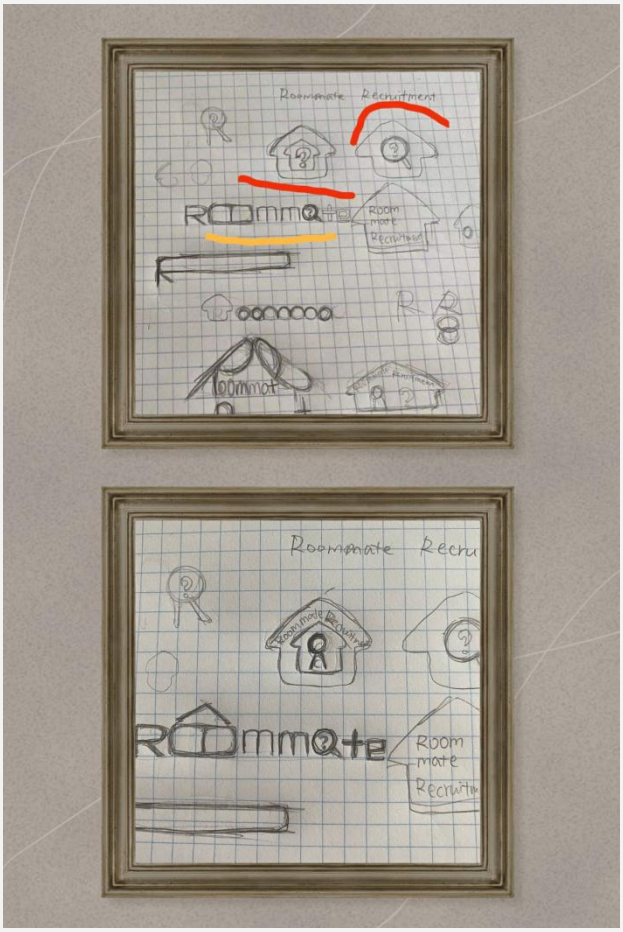
In terms of cookies, the RR should adhere to EU requirements, and site components should be subjected to a cookie audit.

The RR must comply with the UK Equality Act (2010) with respect to accessibility issues.

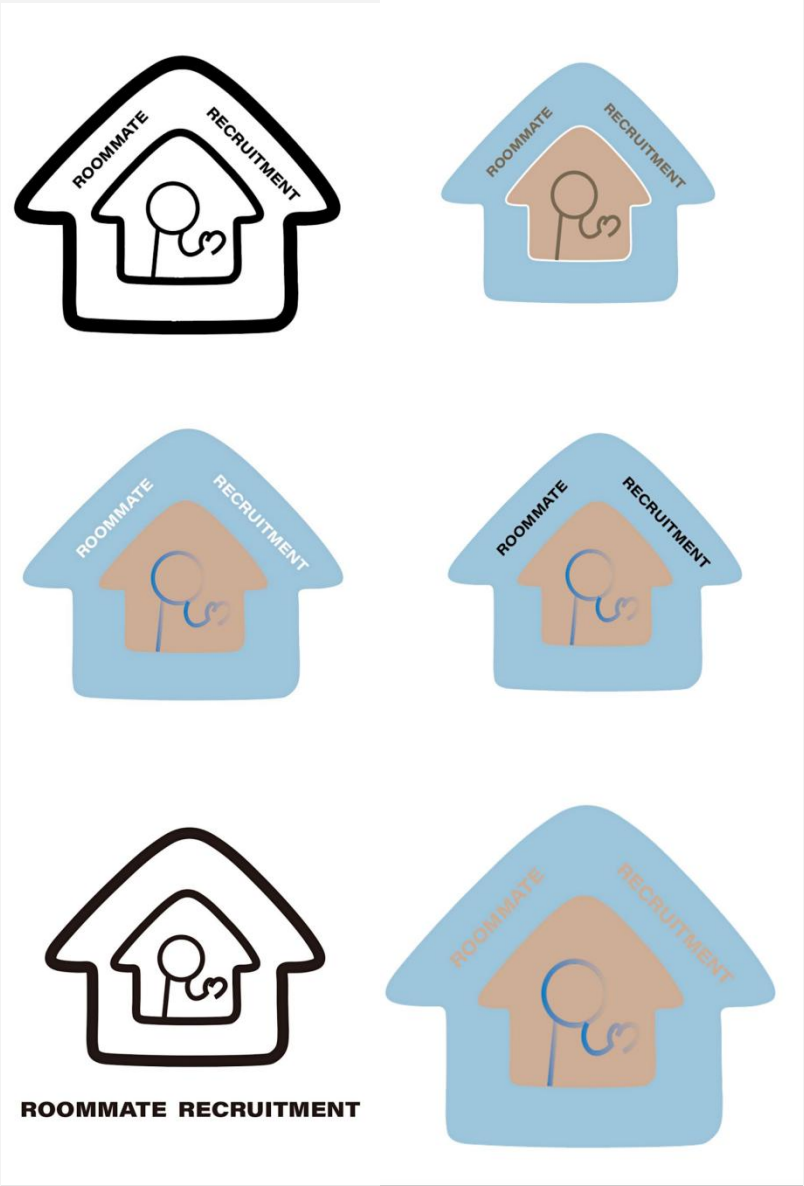
Owing to accessibility, the RR must adhere to the UK Equality Act (2010).

8. Logo design ideas

As we were working on a website to find a roommate, we associated it with a cottage, and then there was a magnifying glass icon with a question mark to indicate a search (the idea was not used in the end), and as the initial letter of the word roommate recruitment is R, the little man inside the house is in the shape of an R with his hand raised to wave to his new roommate.

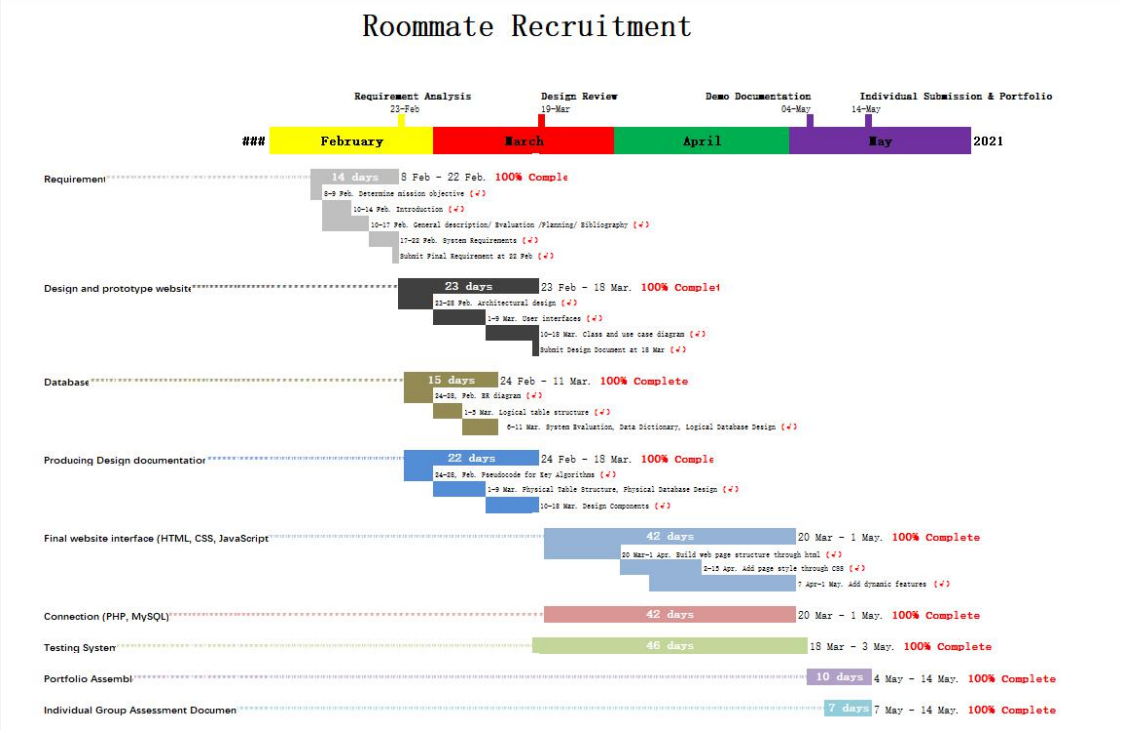


this is draft.



These six images are the finished logo and we ended up choosing the three logos on the left as our website logos to be used in distinct parts of the site.

9.Gantt Table



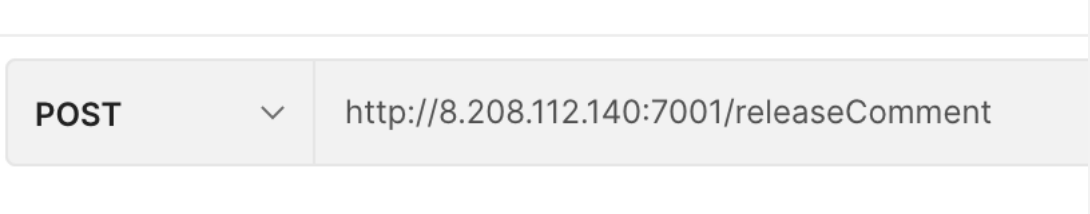
Test Documentation

1. Strategy

1.1 Data used

Data in this RR system mainly includes two different forms, interaction data for interface and long-term storage data for MYSQL Database.

For interface data, the application called Postman is required, which is extensively used for interface testing. It is used to check whether data is valid and accurate when transmission between server-side and user-side. In detail, the choice of data-passing methods is needed first. Normally, POST or GET is the usual method for data interaction. Then, the IP address and port of the system and the router path for certain functions are required to be entered. After that, the JSON form of the data is configured, and then the body data passed through the interface in the next step is input. When the format and data structure are feasible, the required interface will be activated if the code is valid. If there is an error in the code, the request will be rejected.
For example:



The comments-releasing function needs to be tested, and POST method is chosen to transmit data.

The data format is JSON, as the image above. According to the server-side codes, when releasing a comment, "postID", "usertID" and "content" need to be transformed.

If codes are effective and data structures are valid, a successful message is returned (defined by the coder).

```
{
    "code": "000",
    "msg": "Comment Unsuccessful, try later."
}
```

And if codes contain bugs or data structures are invalid, an unsuccessful message will be returned (also defined by the coder). Next, the code will be adjusted or debugged according to the Postman feedback.

For long-term storage data (MYSQL database) testing, SQL queries are required to test "adding", "deleting", "altering", and "querying" functions of the database. And some "meaningless" data that follow the database format can be added to test the presence of the web pages, as the image below shows.

postID	ownerID	post_note	post_date	vacancy	capacity	flatName	price	tenancy	startTime
1619912463	1619912463	Test note for post2	2021-05-01	3	6	Calico	145	44	2021-09
1619912464	1619912464	Test note for post3	2021-05-01	4	6	Host Hope Street	115	51	2021-08
1619912465	1619912465	Test note for post4	2021-05-01	5	6	The Address	175	51	2021-09
1619912466	1619912466	Test note for post5	2021-05-01	2	6	Ablett House	145	44	2021-09
1619912467	1619912467	Test note for post6	2021-05-01	3	6	Prospect Point	165	51	2021-09

Combining interface data and long-term storage data, the whole data system is tested and the performances of their cooperation are verified.

1.2 System testing

This website can work normally in most cases. There are four main functions of the website which consist of login and register, posting, searching and joining, and personal information management.

Users need to register for an account when they first enter the website. After that, users could enter their personal information such as gender, religion, nationality, and major. This information management system helps website users know each other better. For finding new roommates, users can release a new post or search existing posts through several conditions. Also, they are given six recommendation flats on the home page below the selection boxes, which can be selected. After choosing their preference, users can see several posts that satisfy their requirements. Details of a particular post will be displayed on a new page by clicking it. On this page, users can view detailed information such as how many people are currently in this flat, the price, and the tenancy of the flat to achieve a better understanding of this flat. The users can return to the previous page if they are not interested in it, or click the '+' button to apply for joining in. After that, the post owner will receive a private message and decide whether to allow him/her to join or not.

1.3 Acceptance testing

When we first started this project, we were not confident enough to design the cover images of the "Initial View" page (homepage) into a "slide-in" display format. Therefore, in the original design document, we stated that we might replace the cover images with a static image. But through learning and discussion, we solved this problem, and on the "Initial View" of the final web page, we displayed the cover images in the form of a carousel.

In the initial design document, we mentioned that we will display the specific information of the apartment on the "Post List View" page such as High-Definition pictures, location, postcode, number, and personal details of current roommates like gender, major, living habits and requirements for other roommates. However, in the design process of the final web page, we did not find a reasonable method to allow users to upload their pictures as profiles, so we gave up this idea and replaced it by choosing one from our pre-stored profile library as their profile.

2 Results

Although we have met all requirements on our website, there are still two bugs.

Firstly, on the detail page, it will show how many people are already in it, and there is a "plus" button for joining next to those profiles. The bug is if the room is full, the plus button will still be there.

Secondly, on the post list page, there is a function where users can change their preferences in a box on the left-hand side of the page. When the user first clicks the check box, the page will refresh normally but nothing will change. Users need to click the checkbox they just click again; this time results will show with the page refresh.

ScreenShots

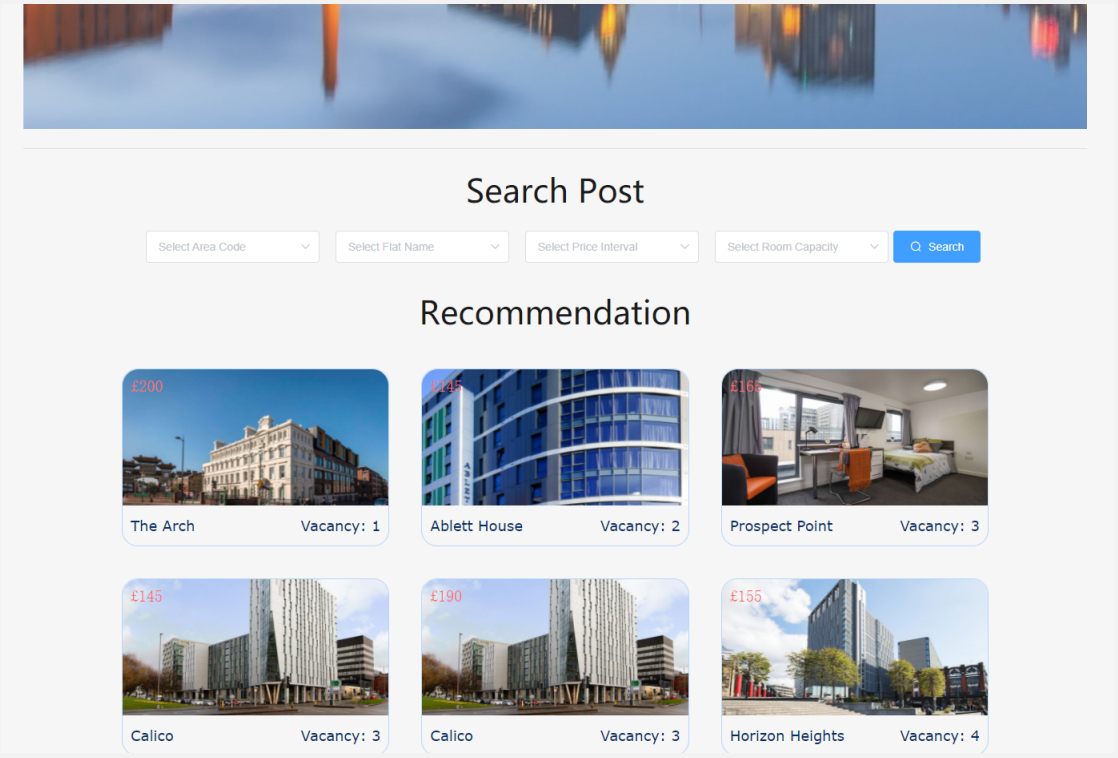


Figure 1: Initial View page (homepage, including a, can, the with cover carousel, filters box, recommended area)

Figure 2: Registration and login page

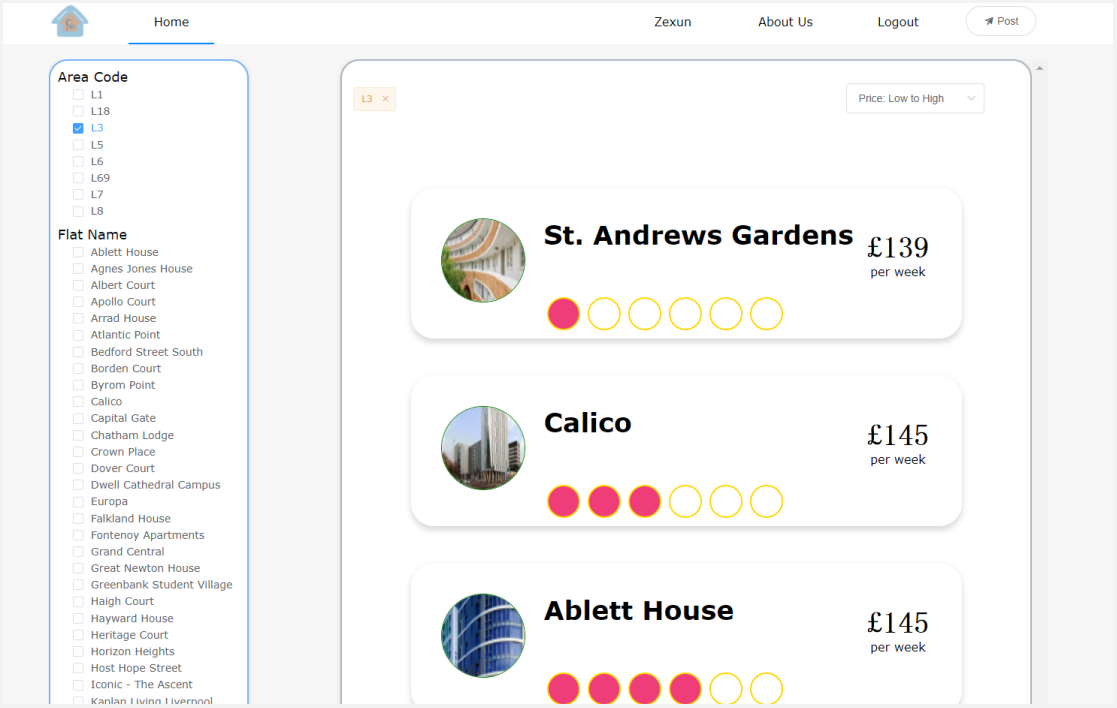


Figure 3: Post List View page (including existing posts that meet the filter criteria, displaysthe price and existing number of roommates)

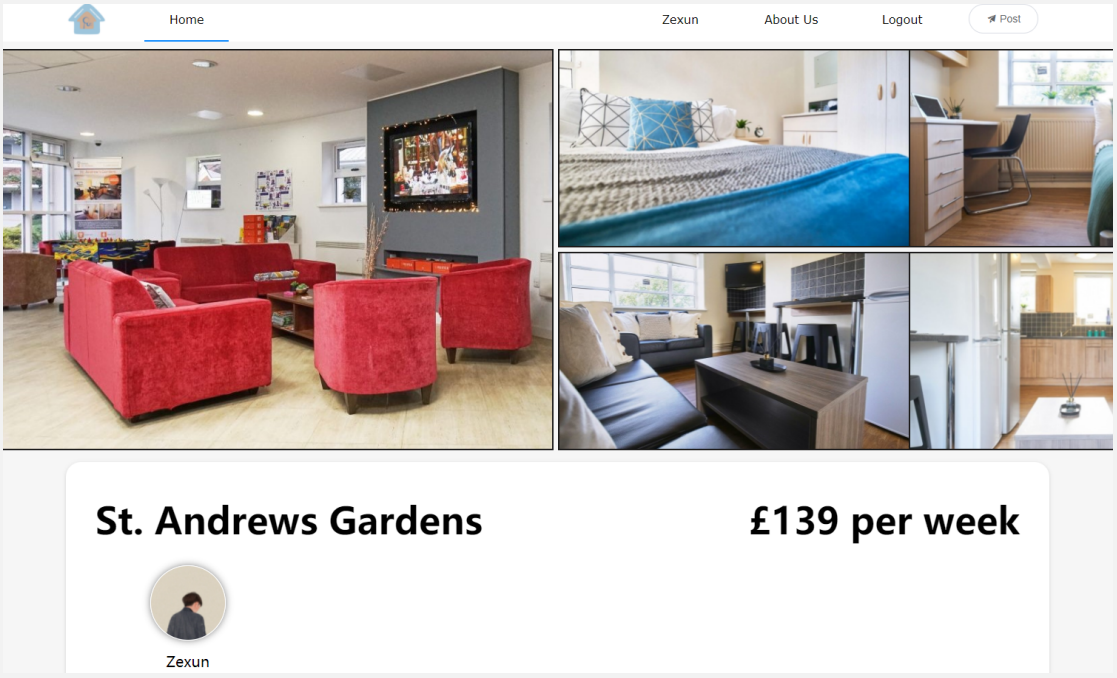


Figure 4: Single Detailed Post View page (including apartment information and publisher information)

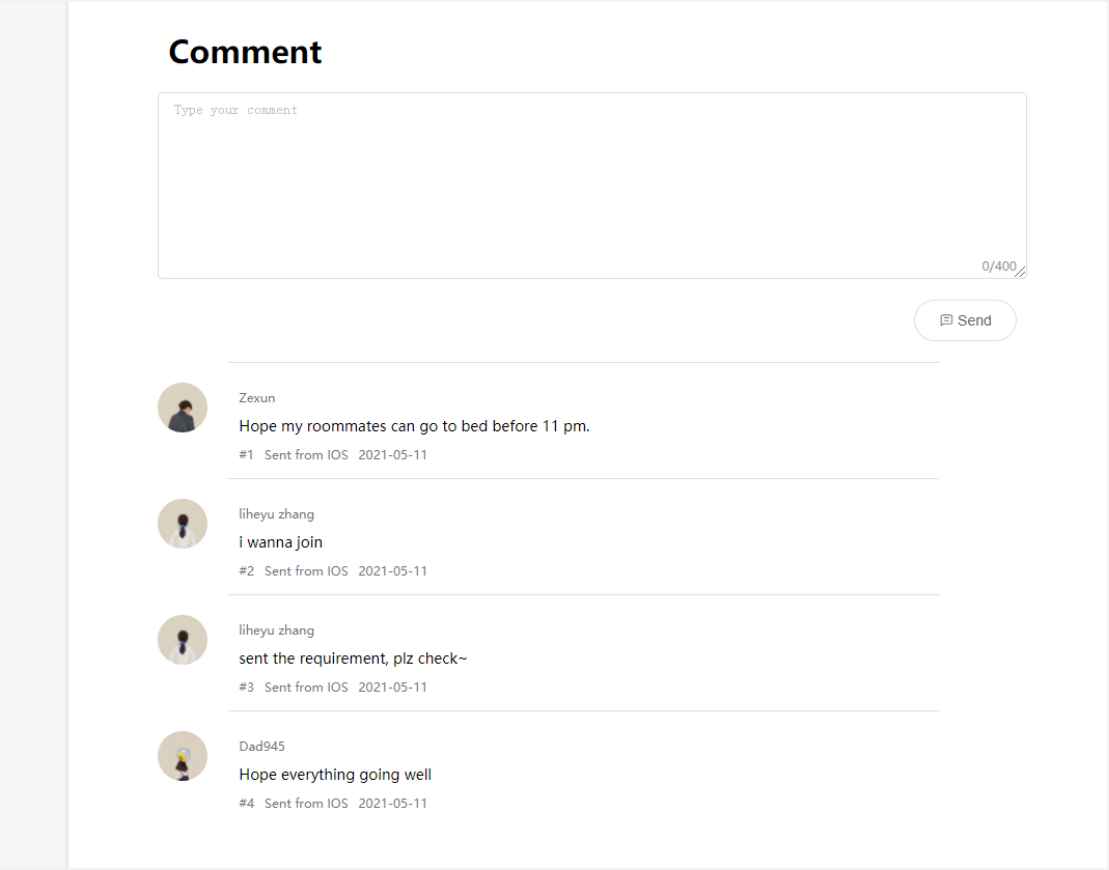


Figure 5: Single Detailed Post View page (including information of existing roommates and comments)

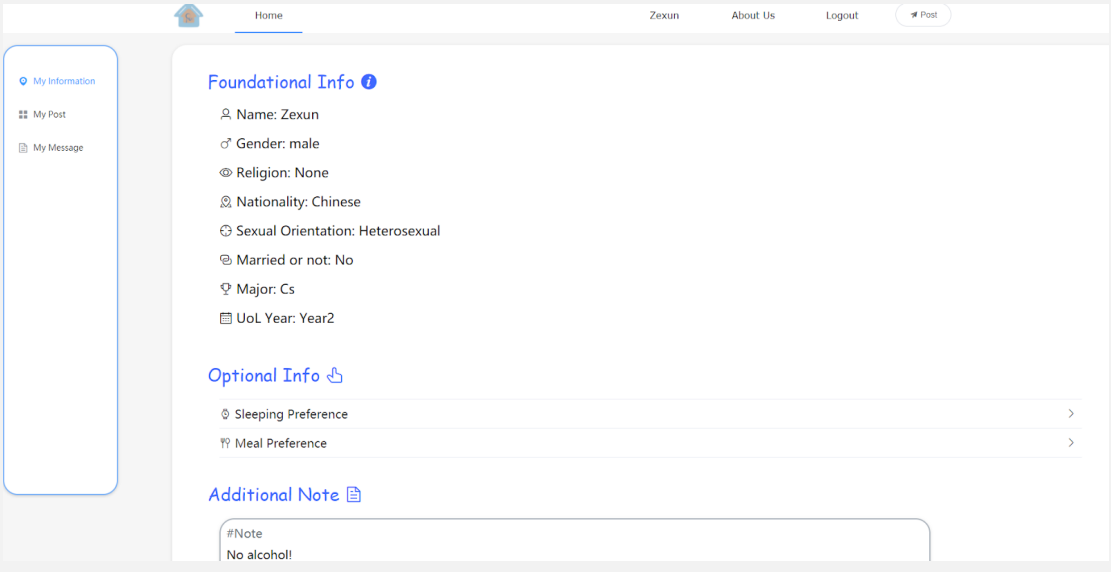


Figure 6: Personal information page (display and modify personal information)

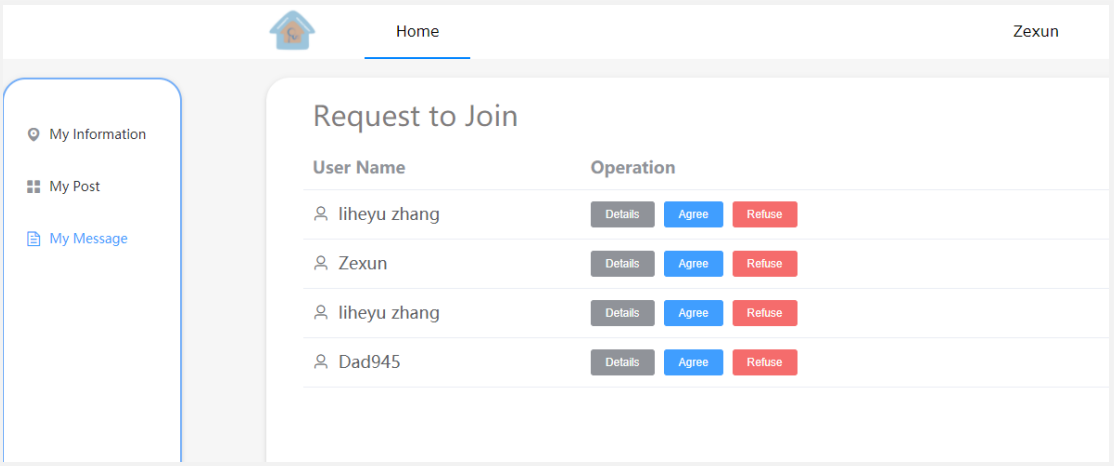


Figure 7: Personal private message page (for publishers, processing users' applications for joining in)