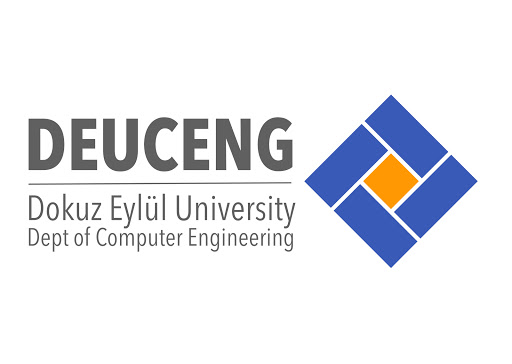
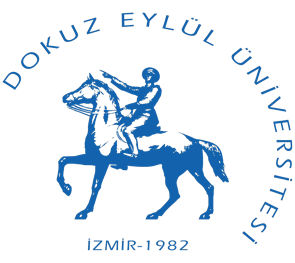
** **

**DOKUZ EYLUL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME 3201 DATABASE MANAGEMENT SYSTEMS**

**Finding Flatmate Web Application**

**Term Project Final Report**

**by**

**Oğuzhan Karakaya-2017510045**

**IZMIR**

**19.01.2021**

**1-Problem Description**

Finding right flat mates is the one of the major issue in the nowadays. The issue is briefly, To match the person looking for a house with the person who is looking for a flat mate.

**2-Specifications of Project**

The user first becomes a member of the system. Members able to find roommates for their own homes. People who want this create an advertisement on the site and mention the house's belongings, wages and compulsory house rules. Those looking for a house that meet these conditions can apply easy via the web system. Also member who try to find home can publish own ad. In addition They can communicate with host the message system.

**3-Solution Domain**

There are two type of user. One has flat, room or something else. And that user requires a flat mate. Let this one called type 1. Other type has no flat or room. And that one searches flat-room and flat mate-roommate. Let this one called type 2. So there should be two table with respect to user type. Type one is will choose flat mate. Type 2 will choose location, flat mate and so on. Type one can reject type two's. Type two apply to type one.

Let make that simple

Type 1 (has flat but has no flatmate or roommate) can do

* register and login the system
* accept/reject offers
* search flatmate
* list people up to requirements (price, empty or with goods etc)
* send offer
* chat

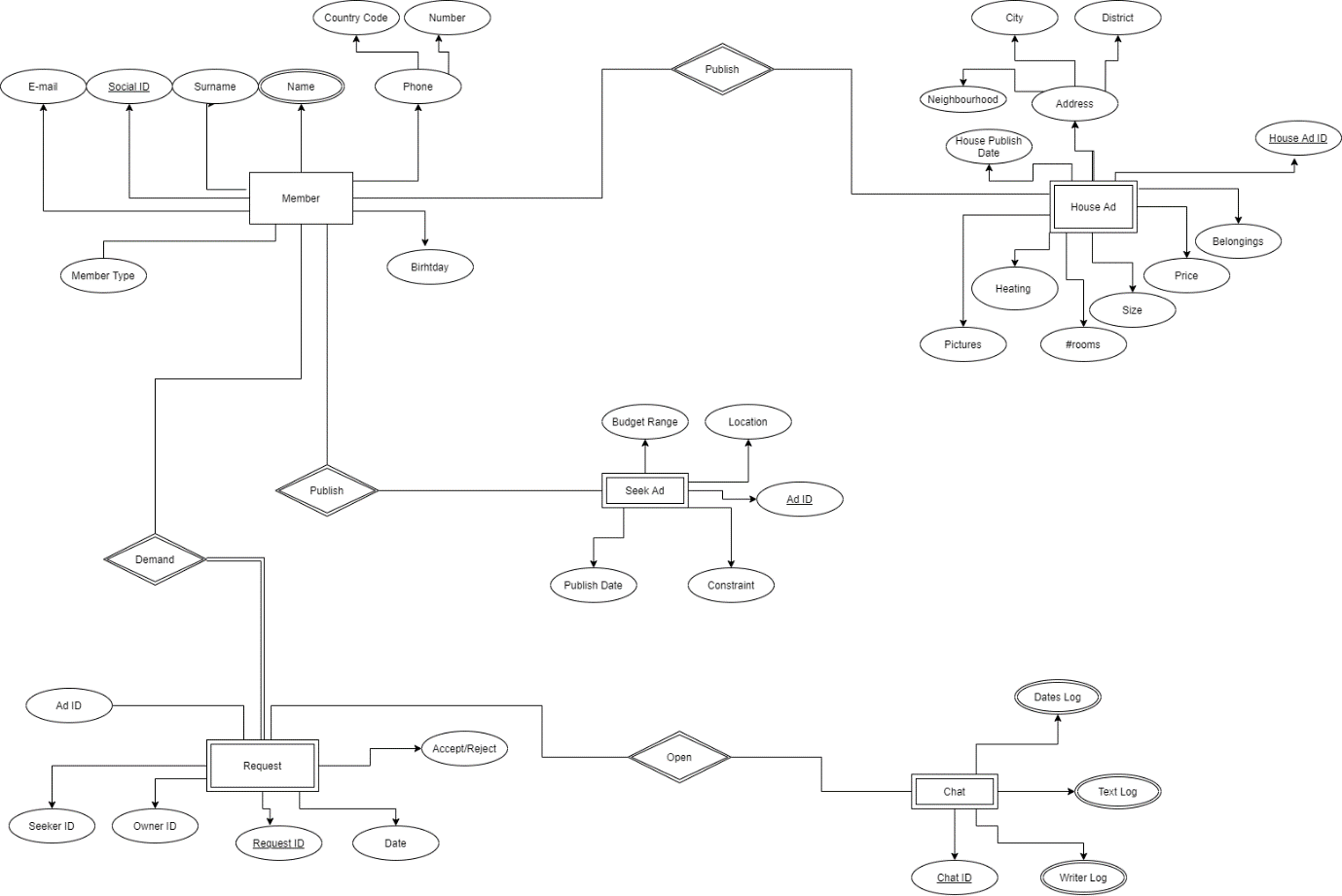
Type 2 (has no flat and friend or has no flat but has friend) can do

* register and login the system
* send offer for room
* chat

**4-System Constraint**

* Members must be over 18 years old
* House owner can reject applicant without giving reason
* Without confirmation applicant can not send message to house owner
* Members must fill the information respect to member type

**5-ER Diagram**

First  ****

Entities:

* Member
* House Ad
* Request
* Seek Ad
* Chat

Last

metin, metal, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Pragnosis**

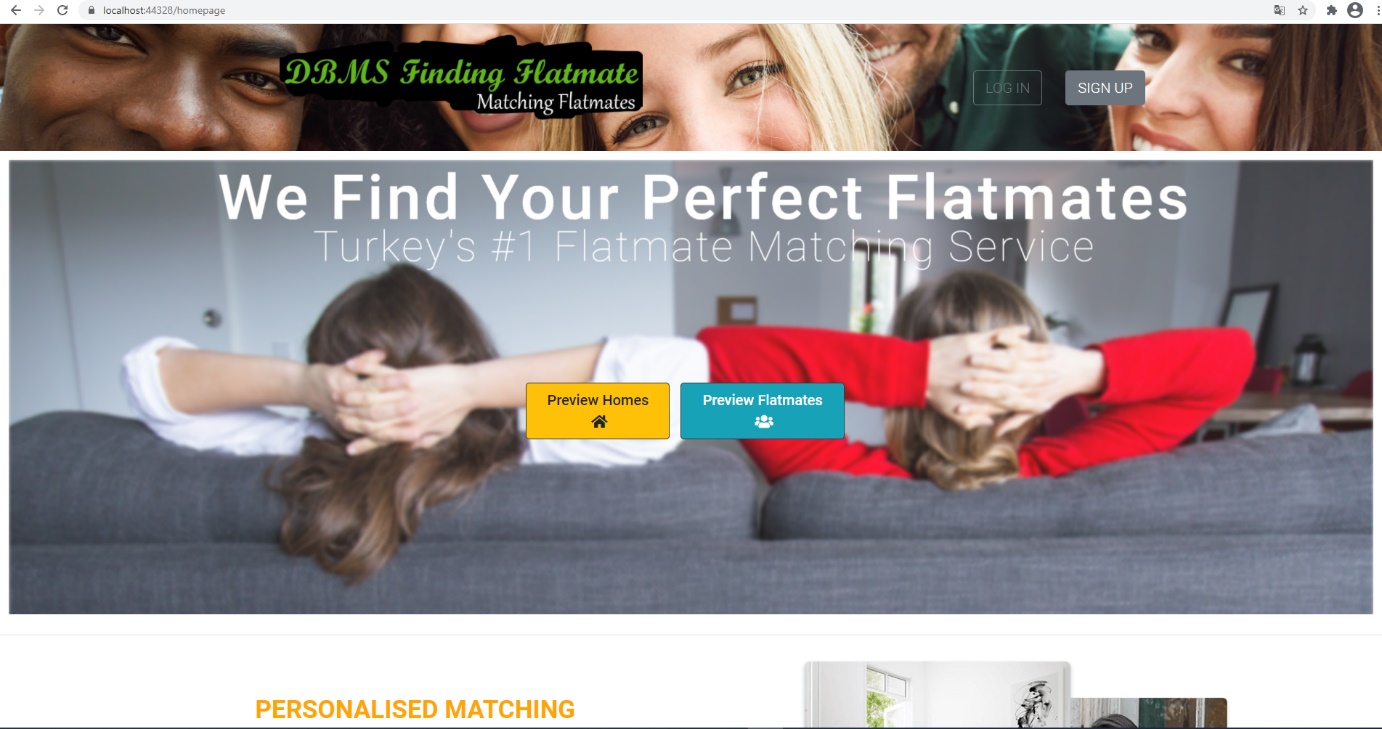
Project members hopes to complete all the task that given by them selves. Communicating between each user of the website could be hard. That’s why maybe chat program of the website will not be completed due to lack of knowledge about web communicating.

Milestones has divided to three in the group. These are front-end, database and database queries, back-end. All the milestones will be completed. But as mentioned just before, there could be some functions that may not be handled.

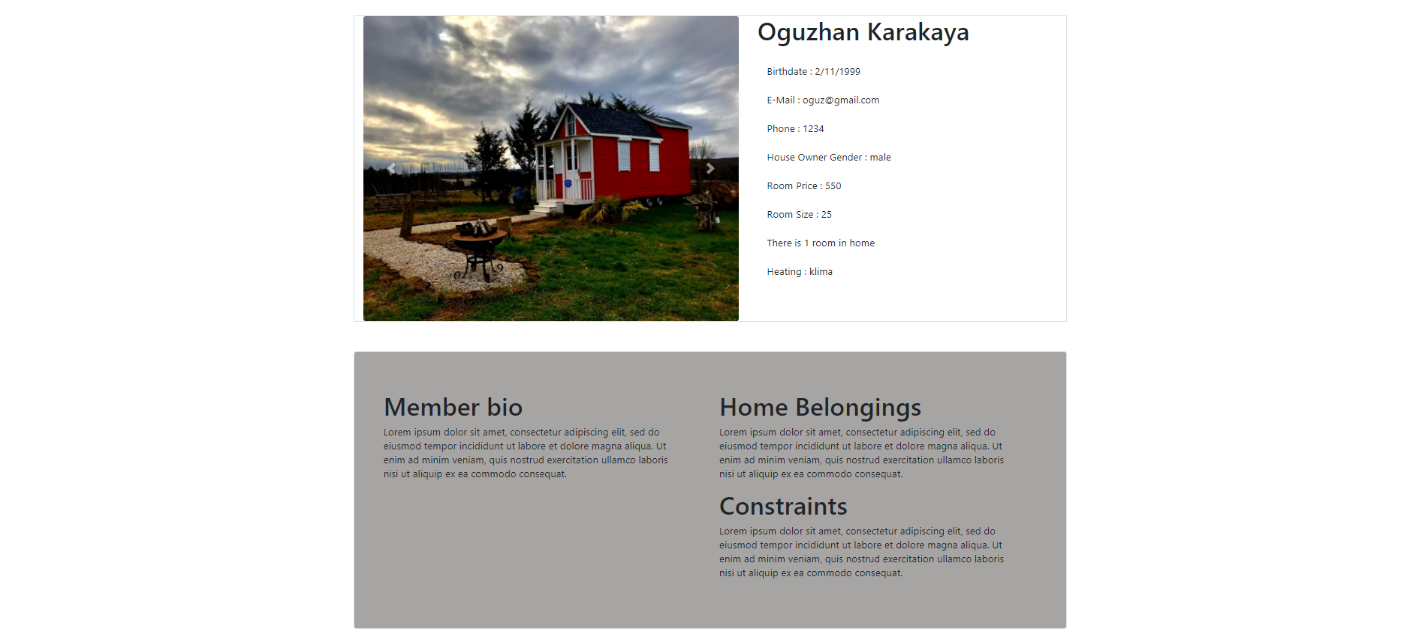
Testing this website will be challenging. Project members have to add abstract users to database and they have to test it in different web browsers such as Chrome, Firefox, Opera and so on. Also Input-Output tests should be handled. These tests will be accomplished at the end of the project timelin

**Layout**

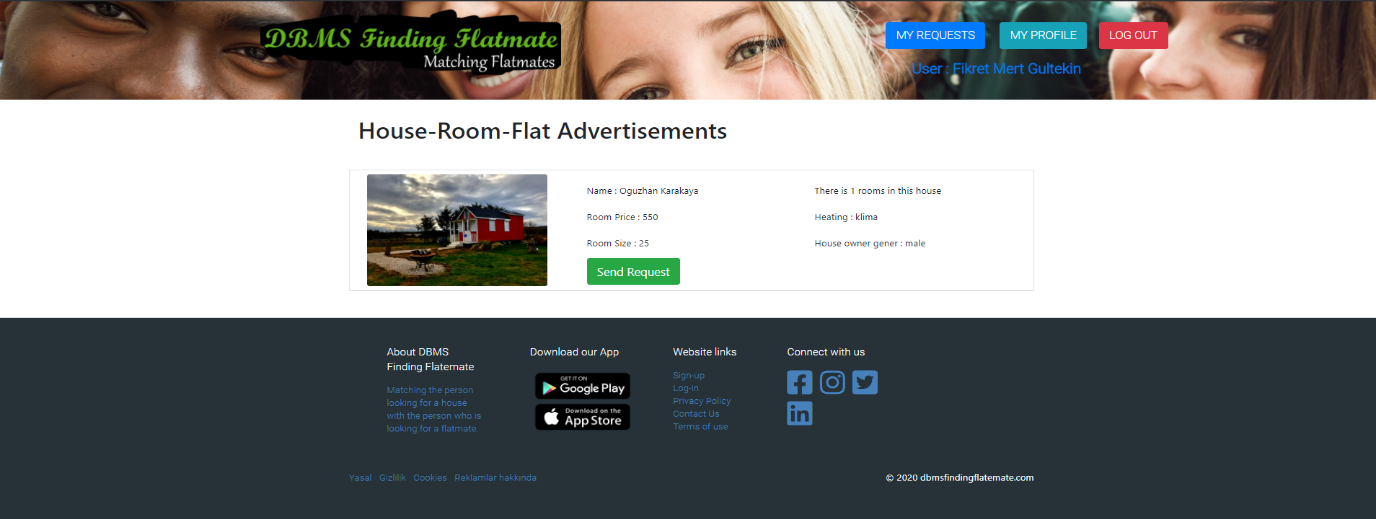
Home Page: Home Page lies on the top of the site hierarchy. User can not reach some features before sign in the system or login the system.

****

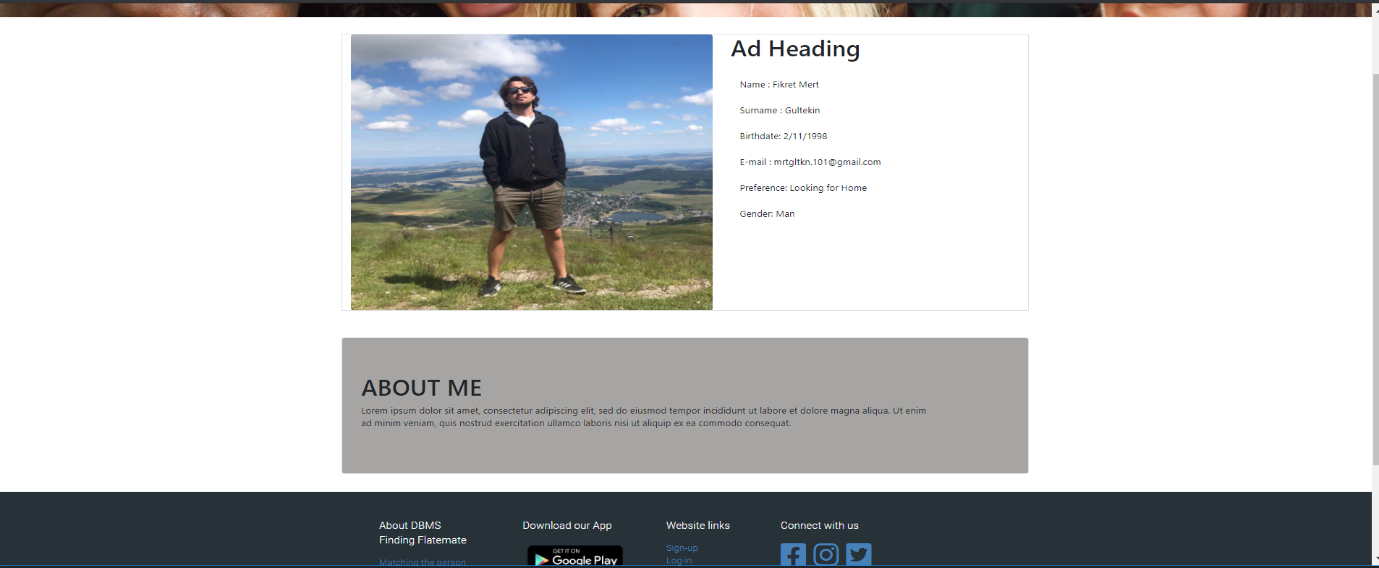
House Details Page: House details page has house owner’s biography, constraints and home belongings. Also features of the house.

****

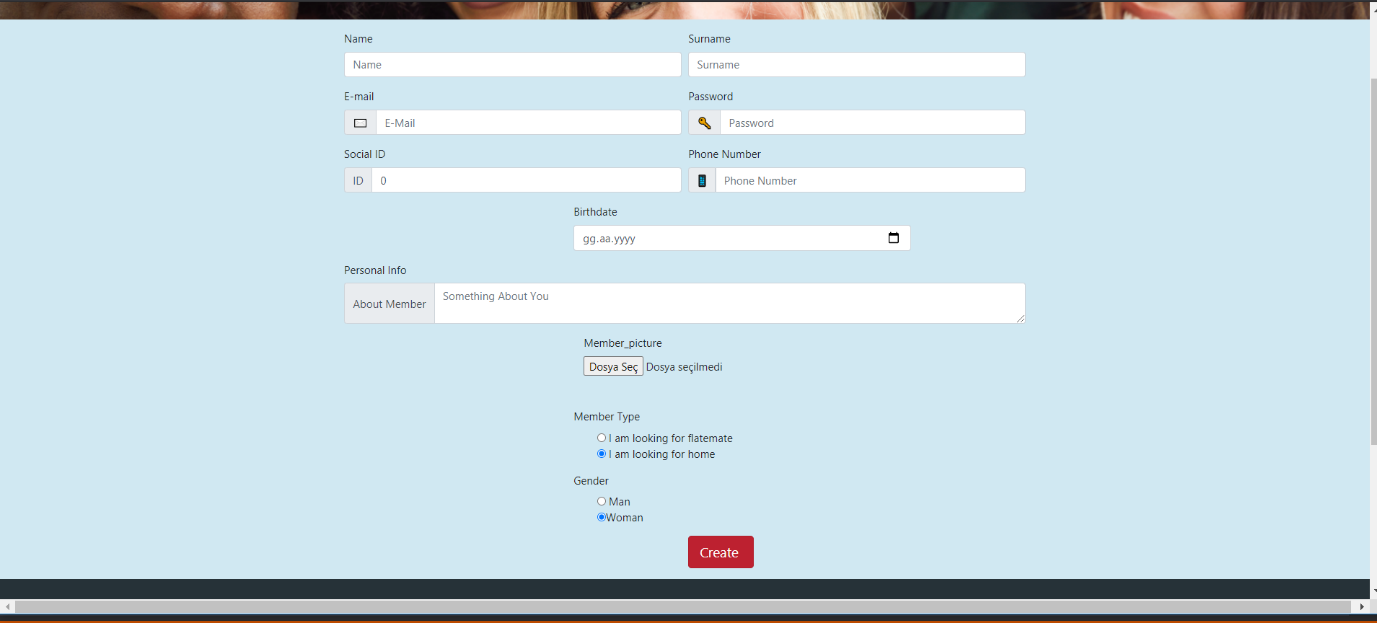
House-Room-Flat Advertisements Page: List of the room-flat advertisements can be shown here.

****

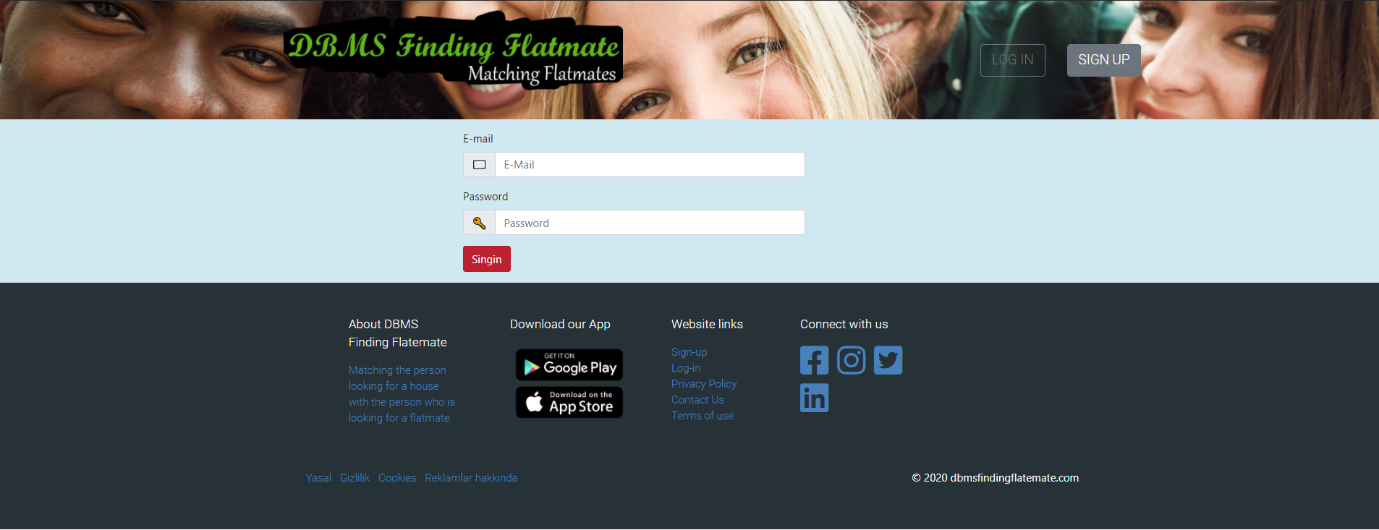
Person Details Page: Person who looking for a room , gives an advertisement, and this users information is shown in this page.

****

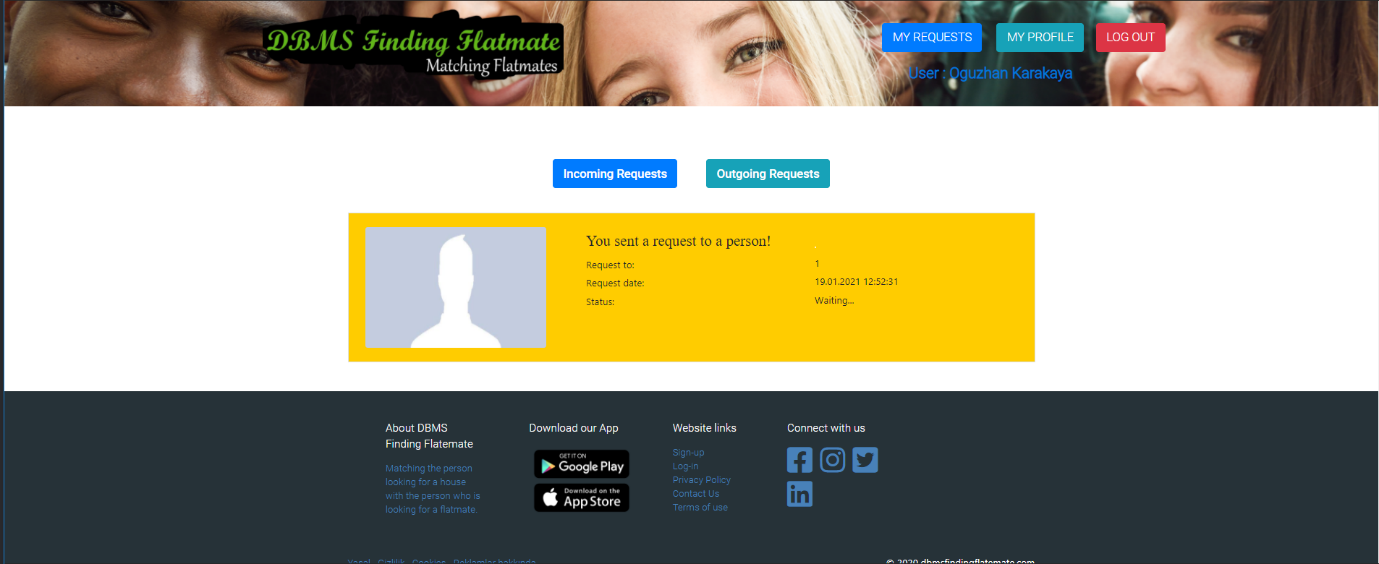
Sign-Up Page: Sign-up page to the system.



Sign in: Sign in page to the system.



Request Page: Users can send request to be flat mate. And user can see outgoing and incoming requests.



**Abstract**

First of all, we learned the basics about the first part we will do, such as how to use mvc. After that we shared the pages neatly on each member and almost all the front end pages were created. At this stage, the bootstrap library has been actively used. After all these steps, all front-end pages have been merged into a single layout. Backend frontend compatibility tested with a simple database connection. Membership feature to the site is completed with backend connections. The database table to be created was revealed and expressed in the er diagram. All connections with the last database created were created properly.

**Completion**

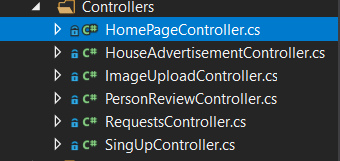
System member registration system has been completed. All information received from the user was placed directly in the profile section. User types were separated. If the priority is a home-facing user, they can send requests to homes and wait in the home-seeking tab. If the user is a user who is looking for a friend to his house, he creates the house after subscribing. The house advertisement appears in the home tab. The application request sent by users to each other appears stably on the notification screen properly. In Addition required button placement is made in navbar section according to membership type. Using some static variables in the code may cause security vulnerability. The chat screen we planned could not be completed as of time. (We tried to write ourselves without getting code)

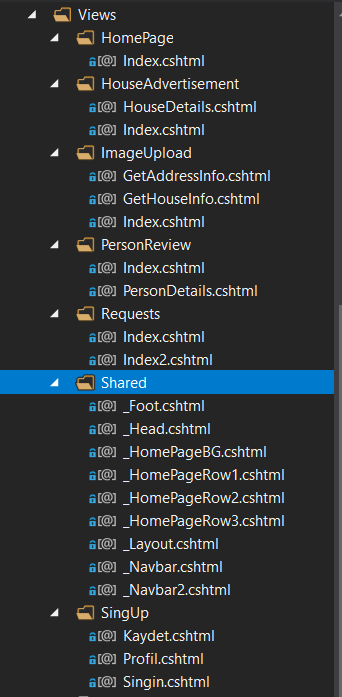
**Functional Decomposition**

House Advertisement: The controller HouseAdvertisement uses select function in database and bring houses and convert them into model then send to related views. If user click the one of the houses image, controller redirect it to houseDetails view and selected house models information published in this page,

Sing Up And Login :Website pull the member table from the model and make member control in the singup control. Here, if the user wants to register on the site, controller direct the registration view to get the information and save it in the database by the controller. If user try to login in this wesite , user login, controller check whether that user is in the member table at the model and log in.

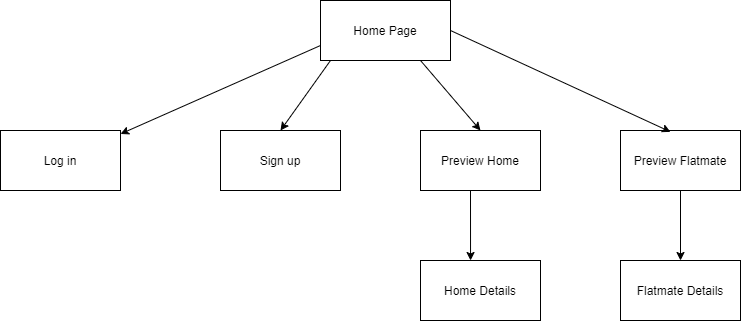
Request: In View of the requests, the requests coming to the user and the requests sent by the user appear. User can accept or reject requests. These operations are performed by the controller. The user activates functions on the controller according to users response.

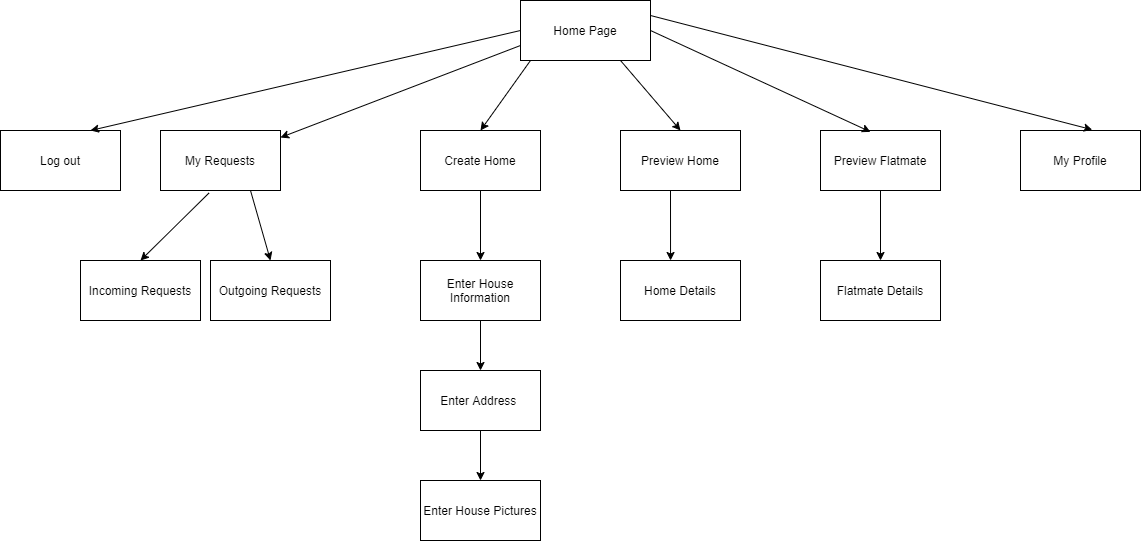


**High Level Organization**

First tree shows the high level organization before log-in the system

****

Second tree shows that the high level organization after log-in the system



**Future Work**

* Chat:

If there was enough time, chat would be added to the application. When the requests between users were approved, a chat would start between them. They would have been able to talk to each other in detail on the application and decide whether to be flat mates.

* Better UI:

The project team has little experience in design. The extra time could be used by improving the UI of the application.