SOFTWARE PROJECT

CSE-408

Section B2

Group No-02

1505092-DIBYENDU BRINTO BOSE
1505094-Raian Latif Nabil
1505109-Ucchwas Talukder

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)
DHAKA 1000

September 19, 2019

Contents

1 Vision Statement

Bachelor bari is an Online Platform that creates a **Mutual Platform** between Users and Owners in a **Secured,Compact,Combined**. It is an online platform that will create a friendly atmosphere and also provide users to give rent there apartments and also take rent from other users.

1.1 Project Goals

- Our system mainly solve the the problem of bachelor and students housing problem.
- It has created a platform to give rent there free apartments and also take rents
- Our primary targeted city is dhaka and to solve its housing problem
- This system will also create a platform for the house owners by which they can give rent to there hostels
- So students and bachelors and travellers wont face any problem for finding there accommodation.

1.2 Project Scopes

- Our system is a concrete and mutual platform for users and owners
- Customer will act both as owner and users and they can give and rent apartments according to there choice
- Users and Owner can both rate about the system .
- Overall system including users,apartments and rooms will managed by the system admin
- Also users and admin can provide reports according to there choices

1.3 features or requirements

Essential Features:-

Importance	Feature Name
Essential	User Registration
Essential	Admin Conformation of Users
Essential	Apartments add and manage
Essential	Individual Room add and manage
Essential	Apartment Renting
Essential	Basic Search
Essential	Manage booking
Essential	Admin Verification

Importance	Feature Name
Desirable	Apartment Recommendation System
Desirable	Apartment Rating
Desirable	User Rating
Desirable	Individual User Report
Desirable	Charts Generation
Desirable	Advance Search Option
Desirable	Advertise Feature
Desirable	User Block System
Desirable	Verification of Apartments

Desirable Features

1.4 Project Milestones

- This is a mainly web-based system that provides a platform for renting apartments
- Users can provide Ratings and reviews to apartments and owners
- Recommendation System is also added according to search of users
- Advance search system is also added in the system
- Overall system will be managed by system admin

2 Software Requirement Analysis

- For the initial steps we went to owners and students
- we made some questionnaire and collect about student and data
- all those data graphs are given below

3 Architectural Description

For this system the over all architectural model is given below-

- The system is developed by using **Model View Controller** pattern
- For backend-
  - MySQL Database is been used
  - PHP language is use to manage the backend
  - LARAVEL framework is used for the development
- For frontened-HTML,CSS,JAVASCRIPT,BOOTSTRAP is used.

The **USECASE** diagram and **DATA-FLOW** diagram and of the system architecture is given below-

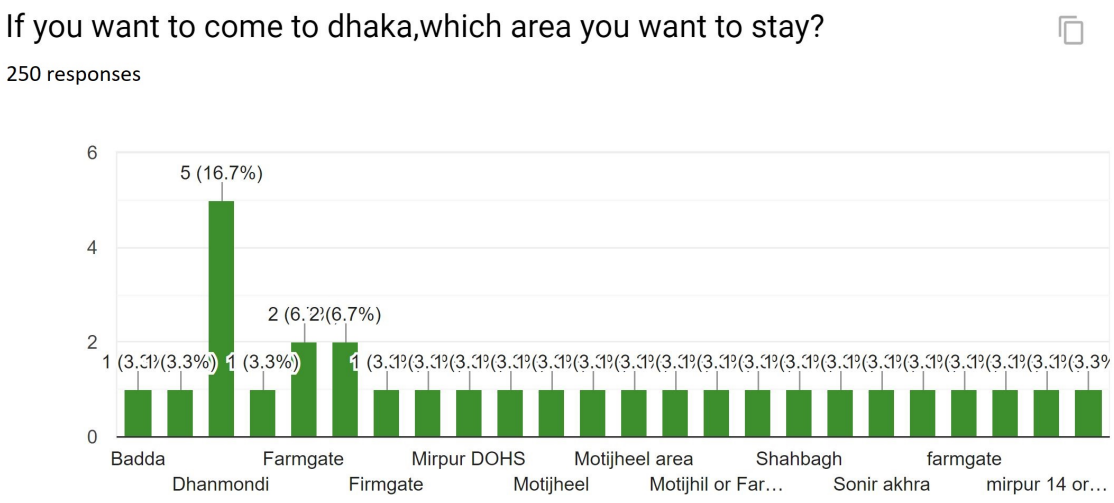

Figure 1: AREA OF APARTMENTS

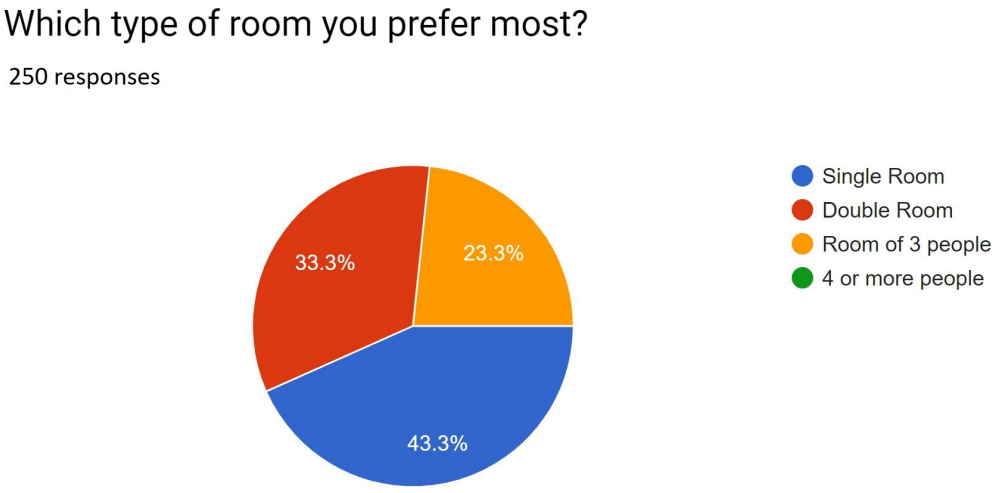

Figure 2: TYPES OF ROOMS

4 User Guideline

These is mainly 2 module for the system.

1. USER MODULE

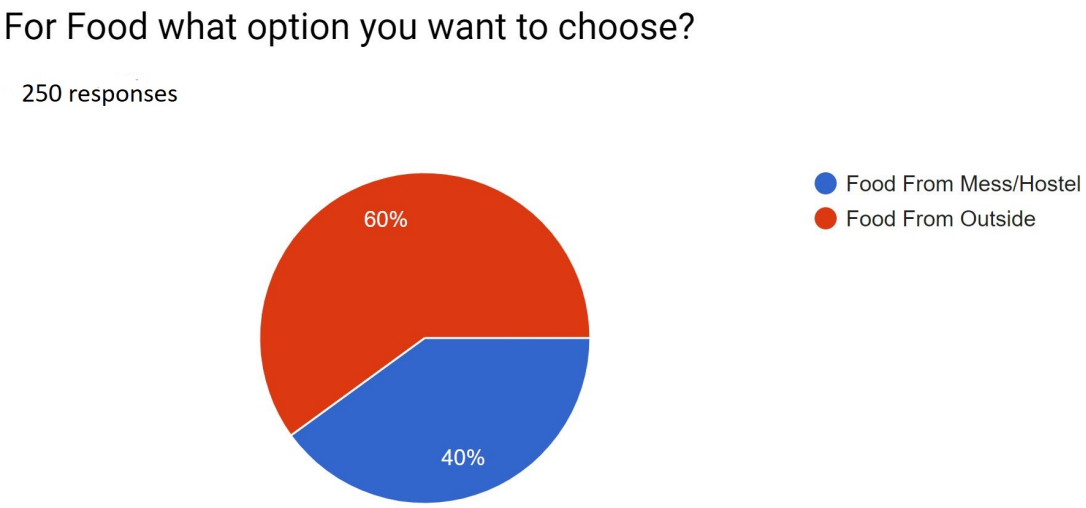

Figure 3: FOOD WANT

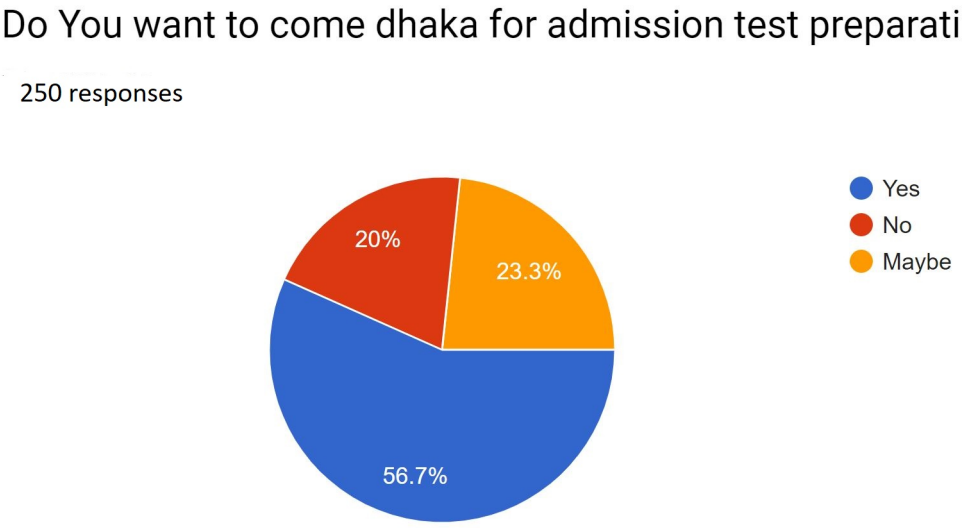

Figure 4: NO OF PEOPLE WILL COME TO DHAKA

2. ADMIN MODULE

The overall functional description of the system with flow for each functionality is given below-

- **Advance Search for Selecting Apartment**

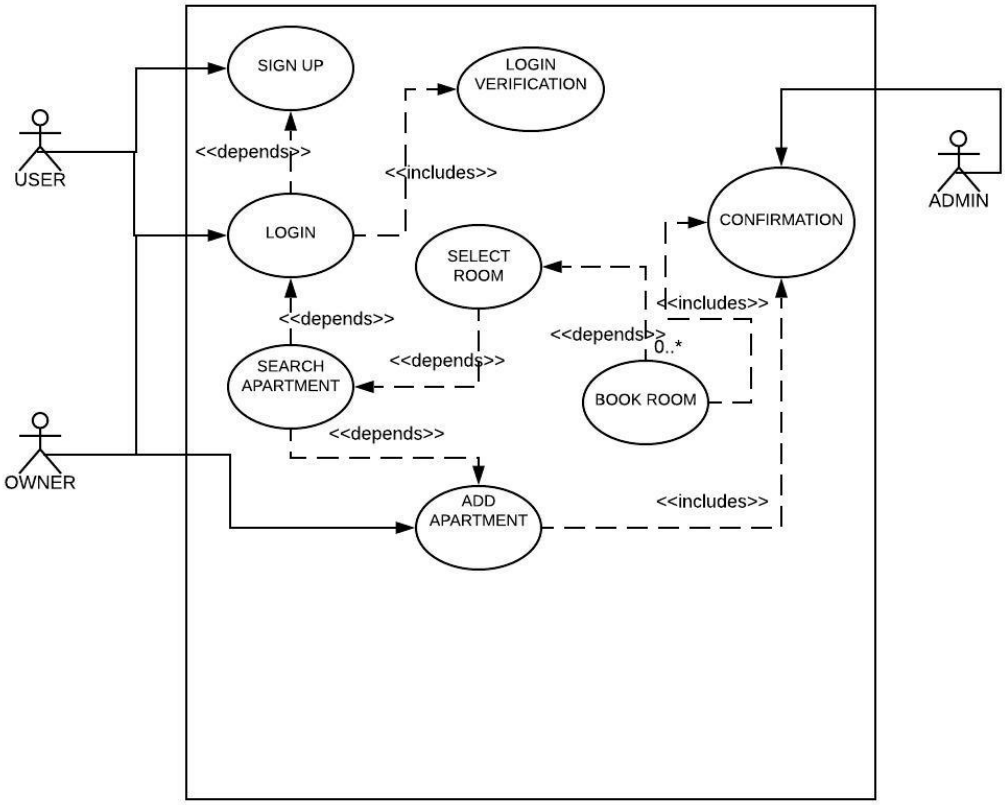

Figure 5: USE CASE FOR THE APARTMENT ADDITION AND CONFIRMATION

- First of all user will come to the homepage and will click the button **SEARCH FOR APARTMENT**
- There is many fields for advance search. User can search by Apartment name,Location,and keywords
- There is seven keywords like **Family,Friend,Pet allowed,late night**. Users can select appropriate things for there needs and then search.
- According to users desire the list of suitable apartments will be shown in a **tabular form**.

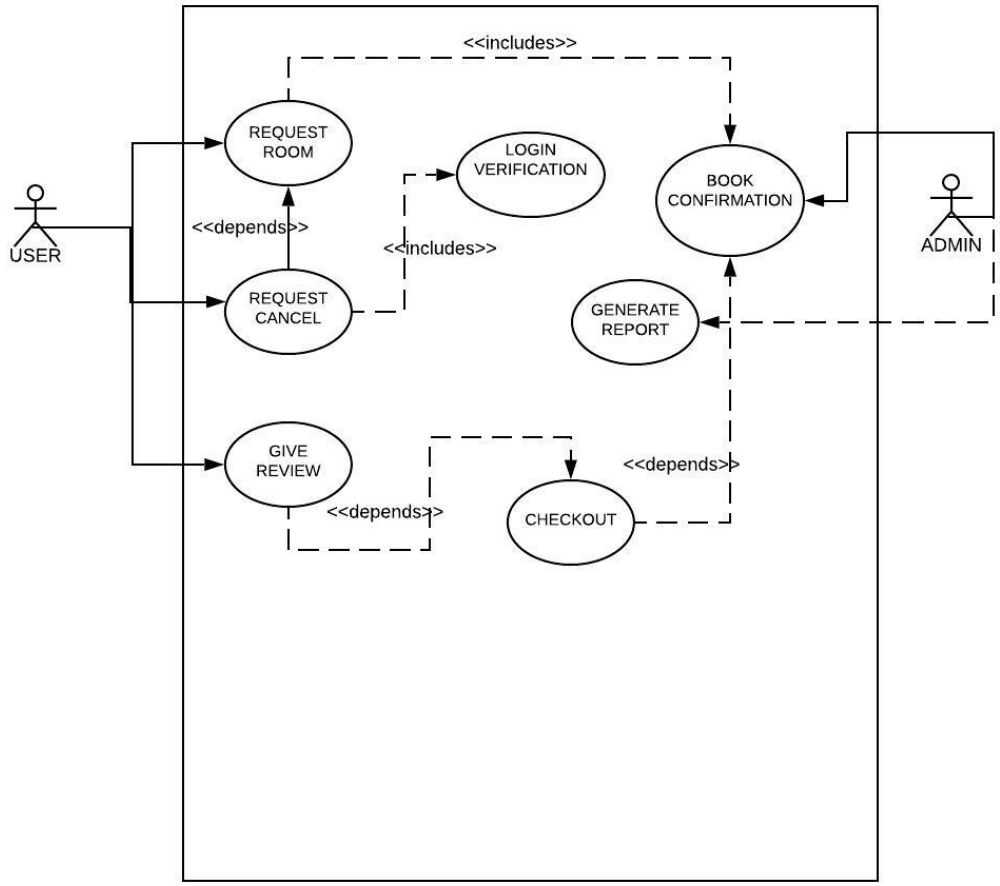

Figure 6: USE CASE FOR THE ROOM BOOK

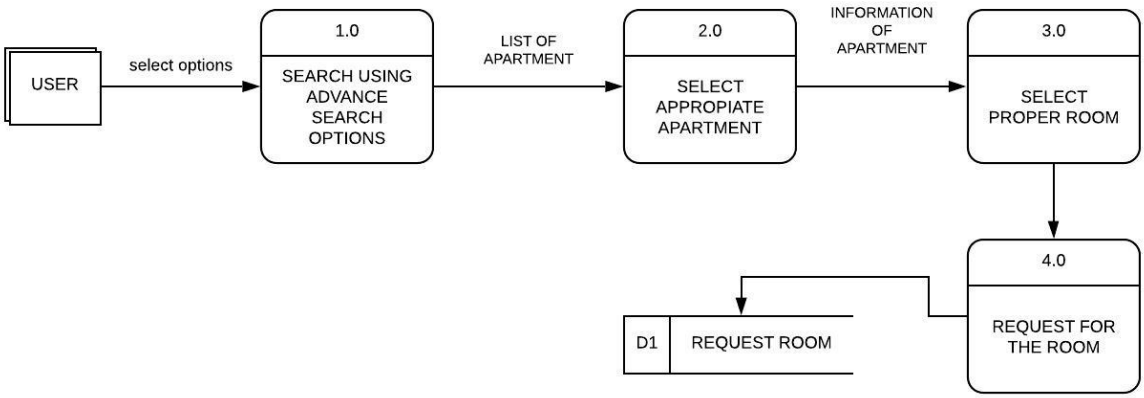

Figure 7: DATA FLOW FOR THE ROOM BOOK

- **Search Appropriate Room in Apartment and BOOK ROOM**
  - after searching for the appropriate apartment,click on the **show room** button

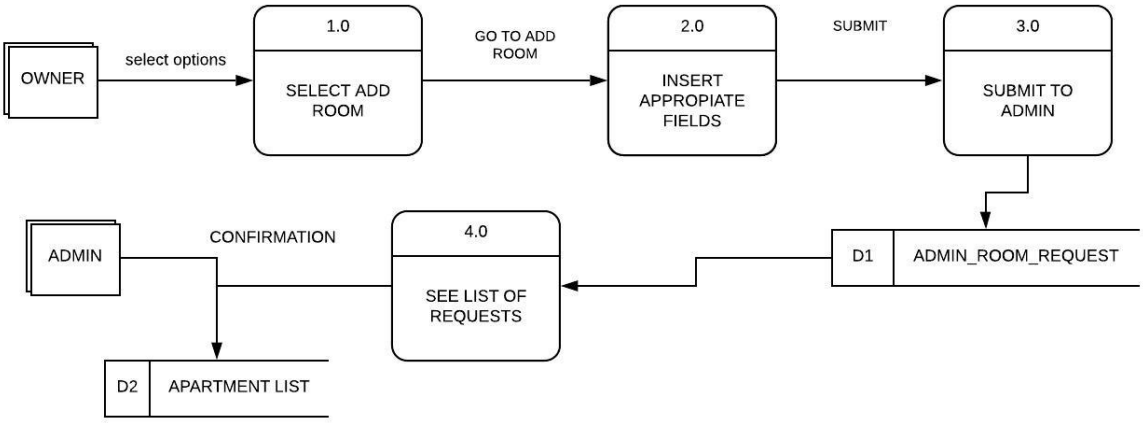

Figure 8: USE CASE FOR THE APARTMENT ADD

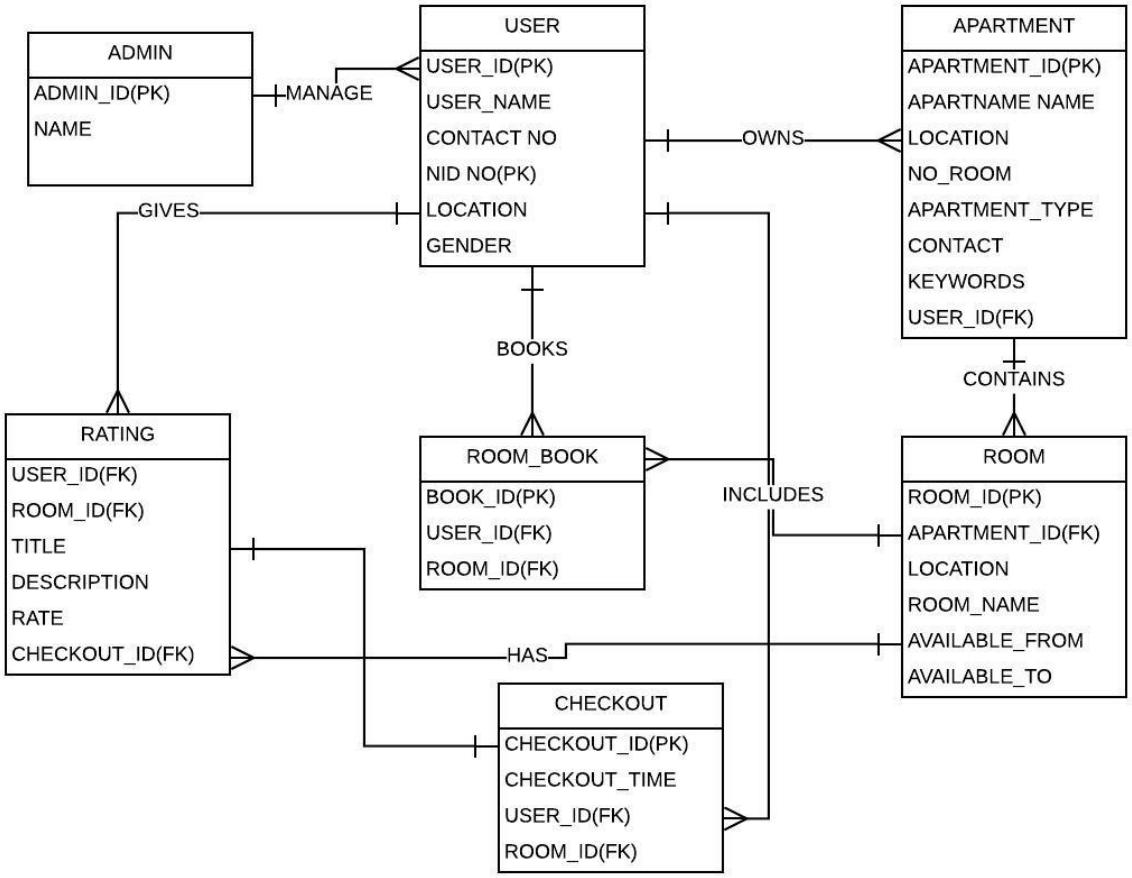

Figure 9: ERD DIAGRAM OF THE SYSTEM

to see the list of rooms of that apartment

– Then the apartment details with **picture,description,location** and individual room description ,price will be shown

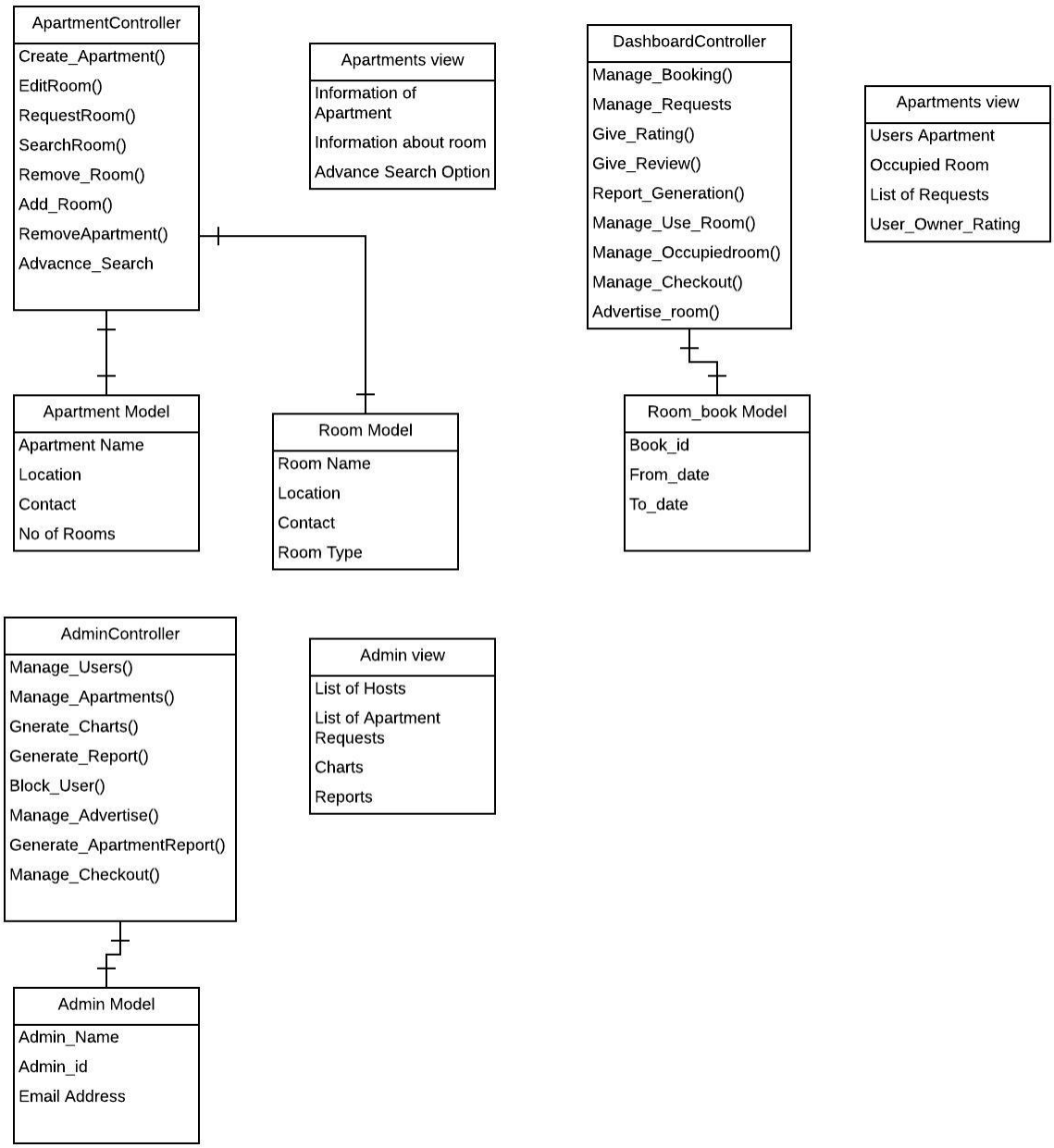

Figure 10: MVC PATTERN OF THE SYSTEM

- By selecting radio button like **AC/NON-AC** or **Family/Share** user can search for the room
- After selecting the room user can also see the **availability** of the room
- Then user can book for certain Days.
- Also in the right side of the window depending on the result of the users search , **recommended apartments will be displayed**
- BY clicking those apartments user can see appropriate apartment information.

- **Creating New Apartment**

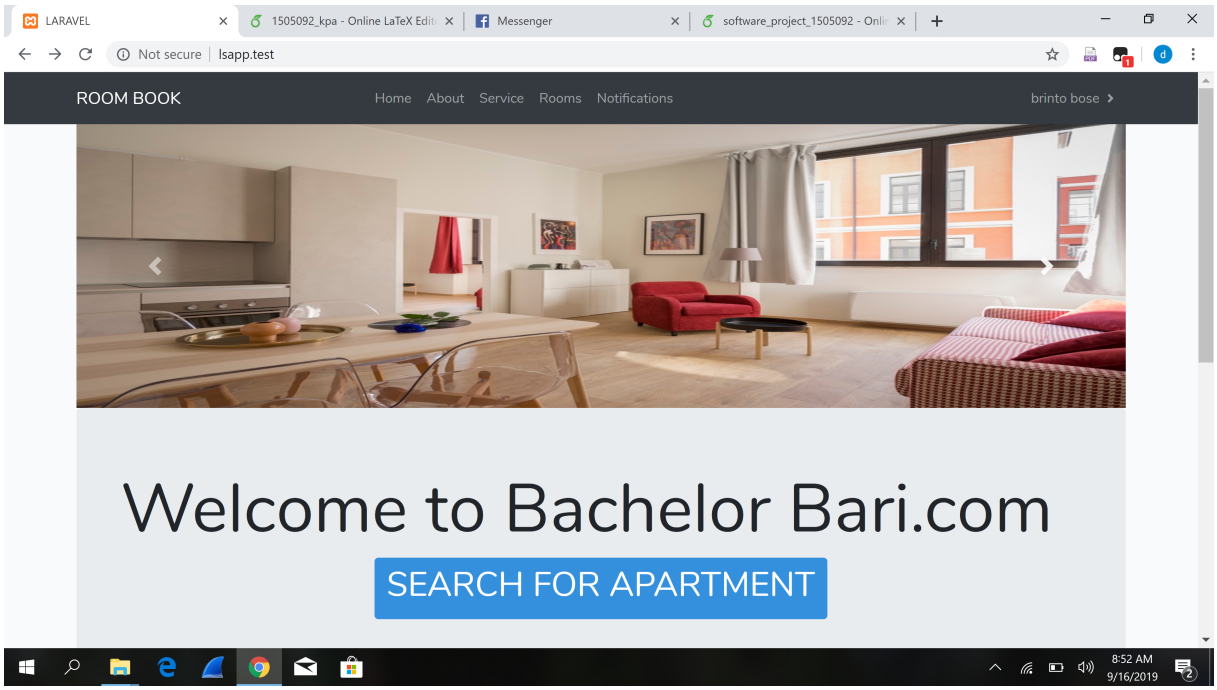

Figure 11: Home Page

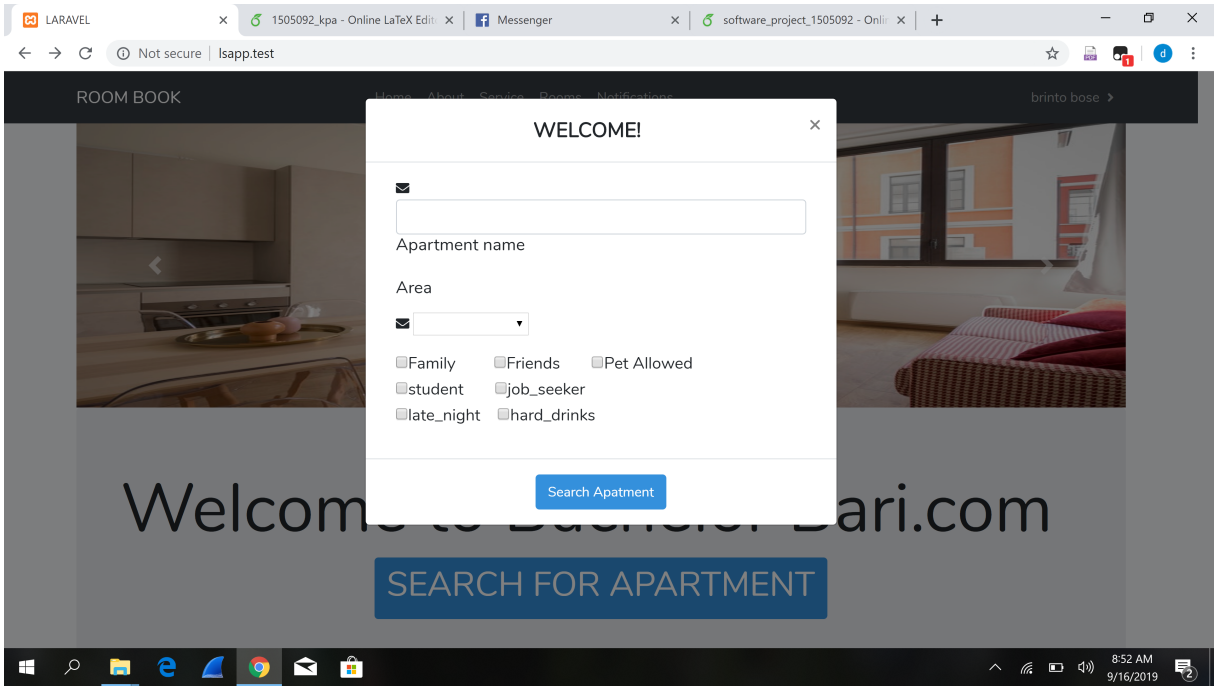

Figure 12: Advance search popup

- After clicking the button in the right side of **user name** user will directed to **User Dashboard**
- This is the window where user can **create,manage,edit,checkout,report generation,advertise there apartments and bookings.**
- By clicking the **Create new Apartment** button user can give information of

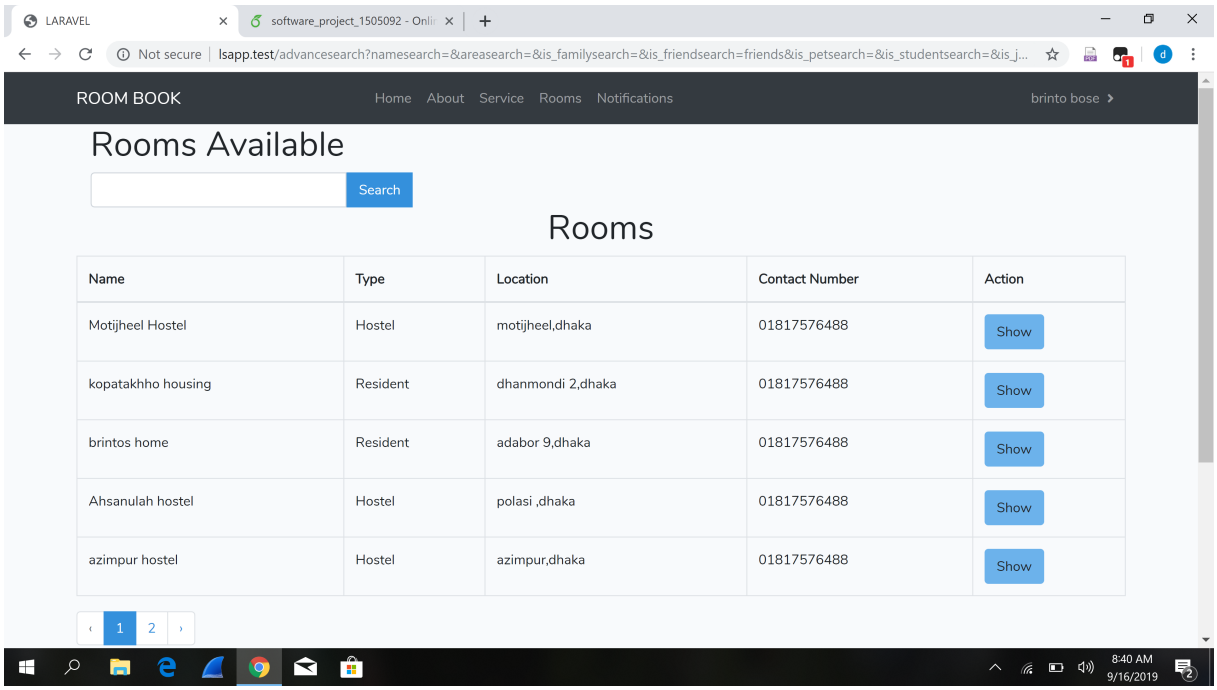

Figure 13: List of Apartments

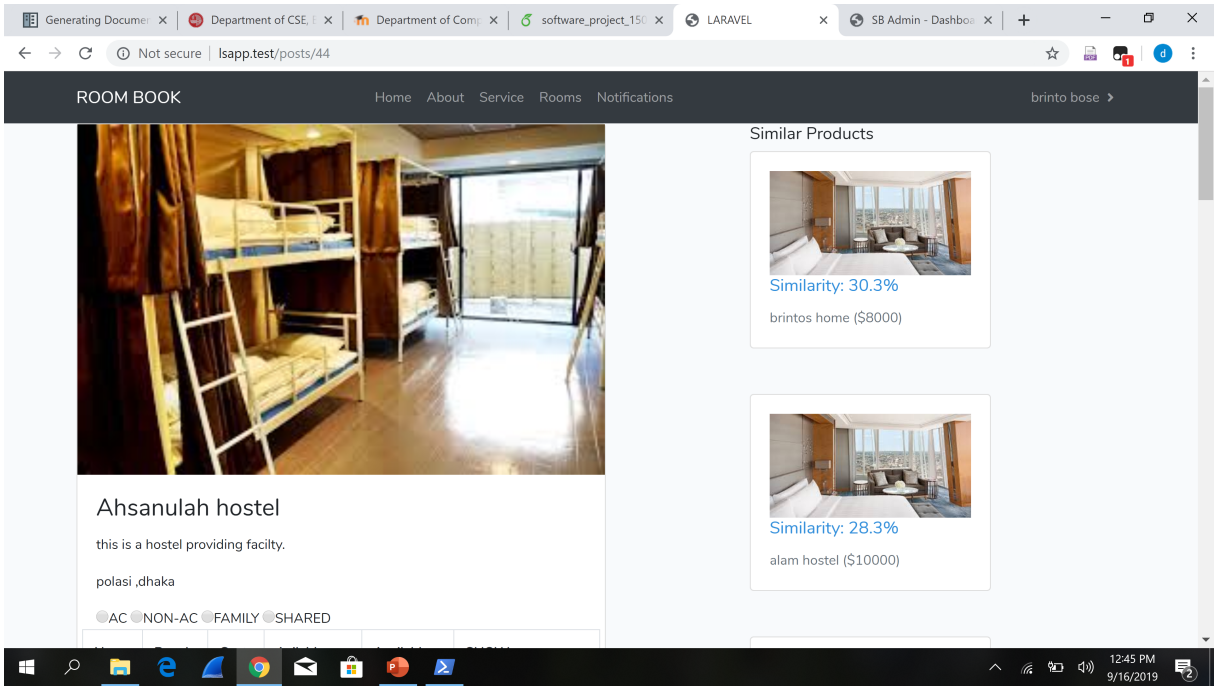

Figure 14: Apartment Description

the apartment that he want to add to the system.

– User have to give information about the apartment like **No of rooms, Each room price,Location,Payment Method,No of allowed people,Contact No,and services(Family,Friend,Pet allowed,student etc)** that apartment gives.

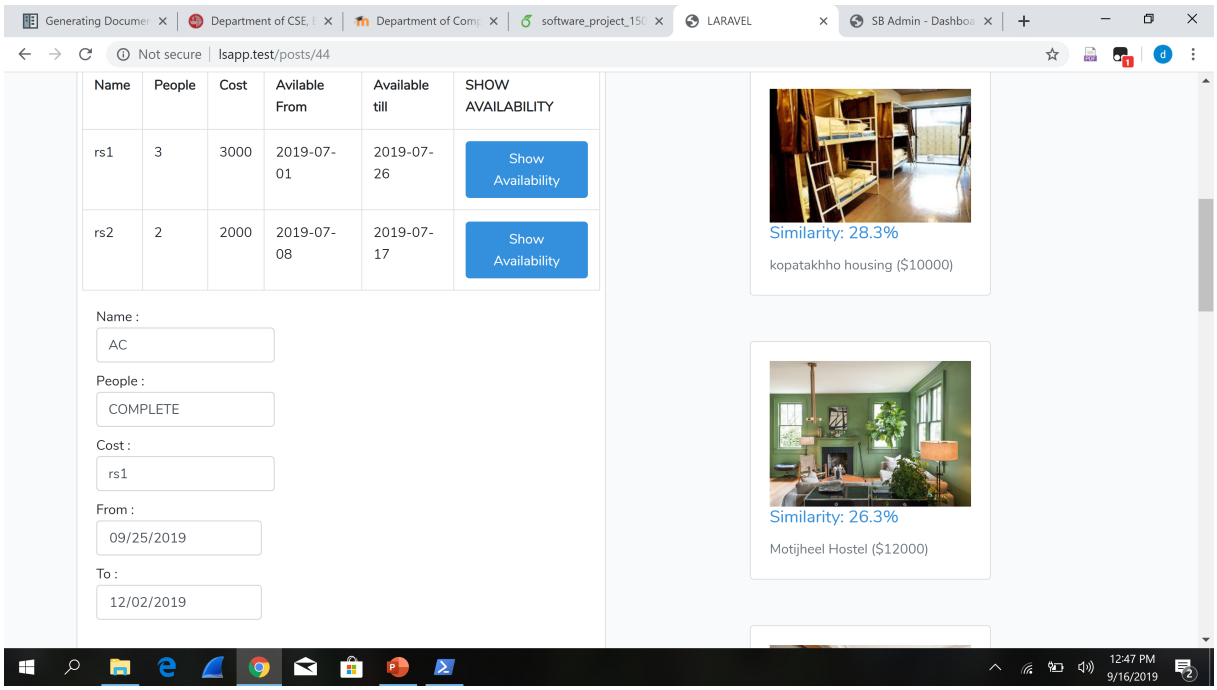

Figure 15: Select room for booking

– After submitting the information properly,the apartment request will send to admin and user have to wait for **admin approval**

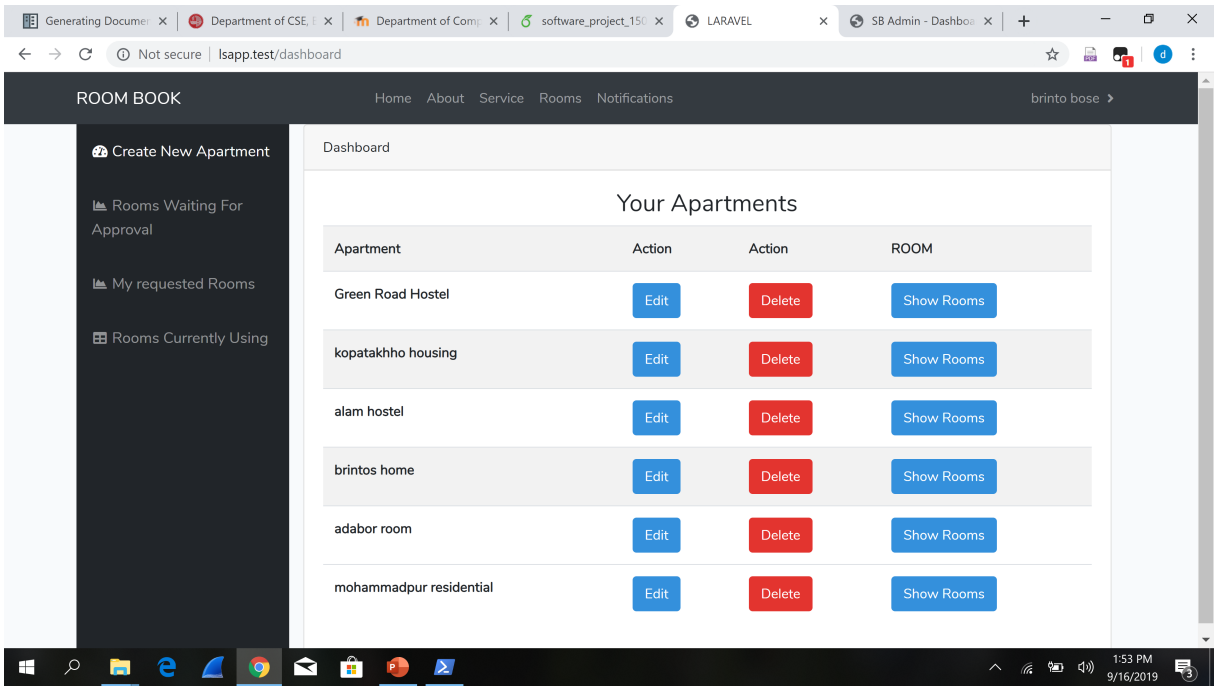

Figure 16: User Dashboard

- **Manage Apartment**

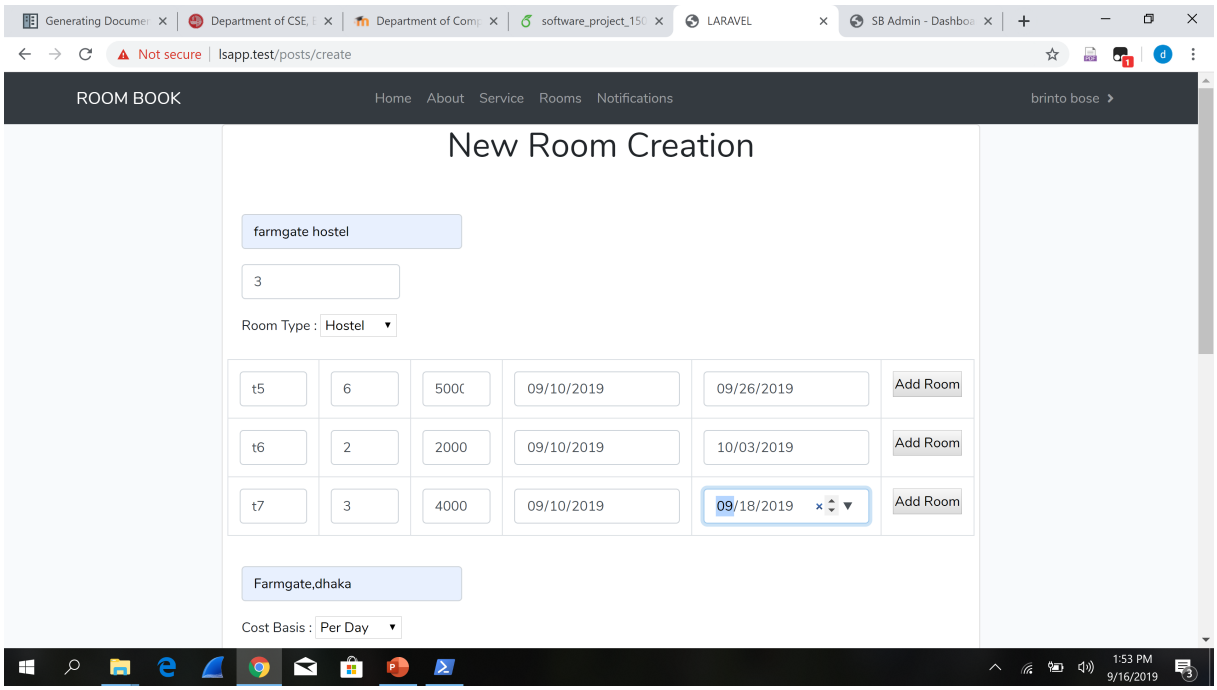

Figure 17: Create New Apartment

- In the **User Dashboard** all the apartments of that user will be listed as Table View.
- User can Edit and delete each apartment information
- Also room information of each apartment of that user are also editable
- User can show the booking status of his apartments from here

- **Confirmation of Booking**
  - Confirmation of Room booking will be done by the owner of that room.
  - Owner can manage the requests in the dashboard .
  - List of **Requested room** will be displayed in a tabular view in the dashboard
  - Owner can confirm or cancel the booking.

- **Maintain and Reporting**
  - Owner can also manage the rooms which are used by different users in **Rooms Currently Using** window
  - Owner can checkout of there rooms after user uses it.
  - Owner can also give rating to the user
  - Owner can advertise there shared rooms after certain user checkout
  - Owner can also generate report about which rooms used by which user and the dates
  - By Selecting the **Convert to PDF** button that report will be in **pdf** format

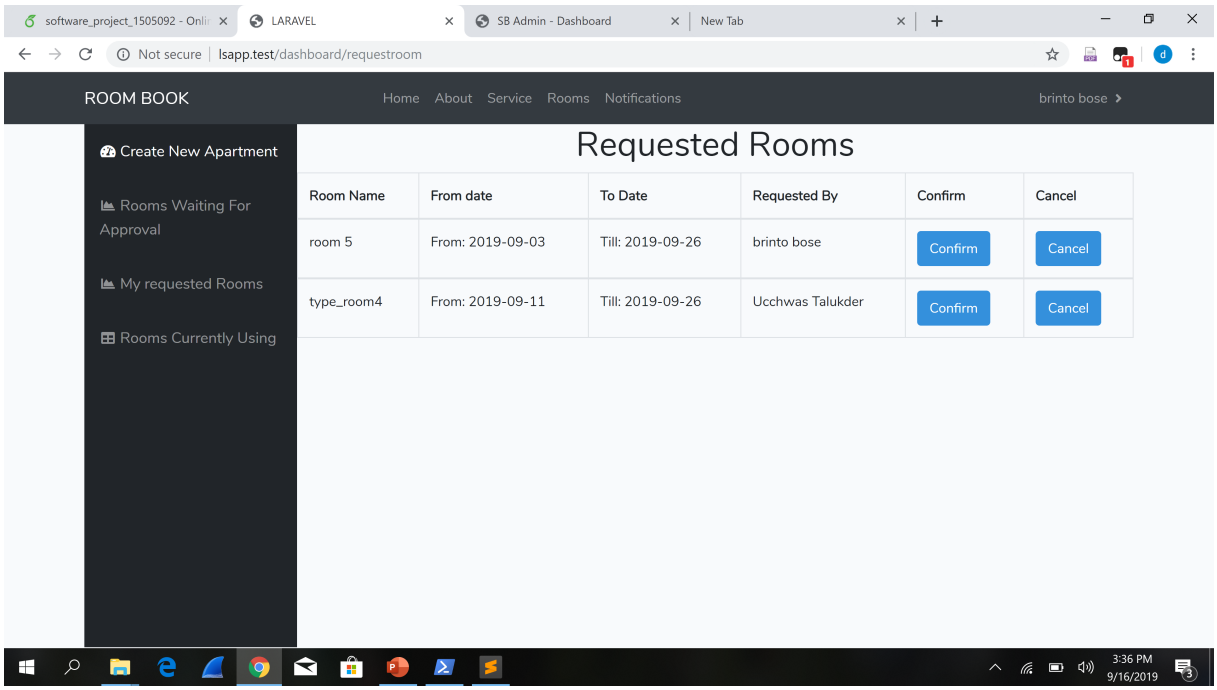

Figure 18: Manage booking

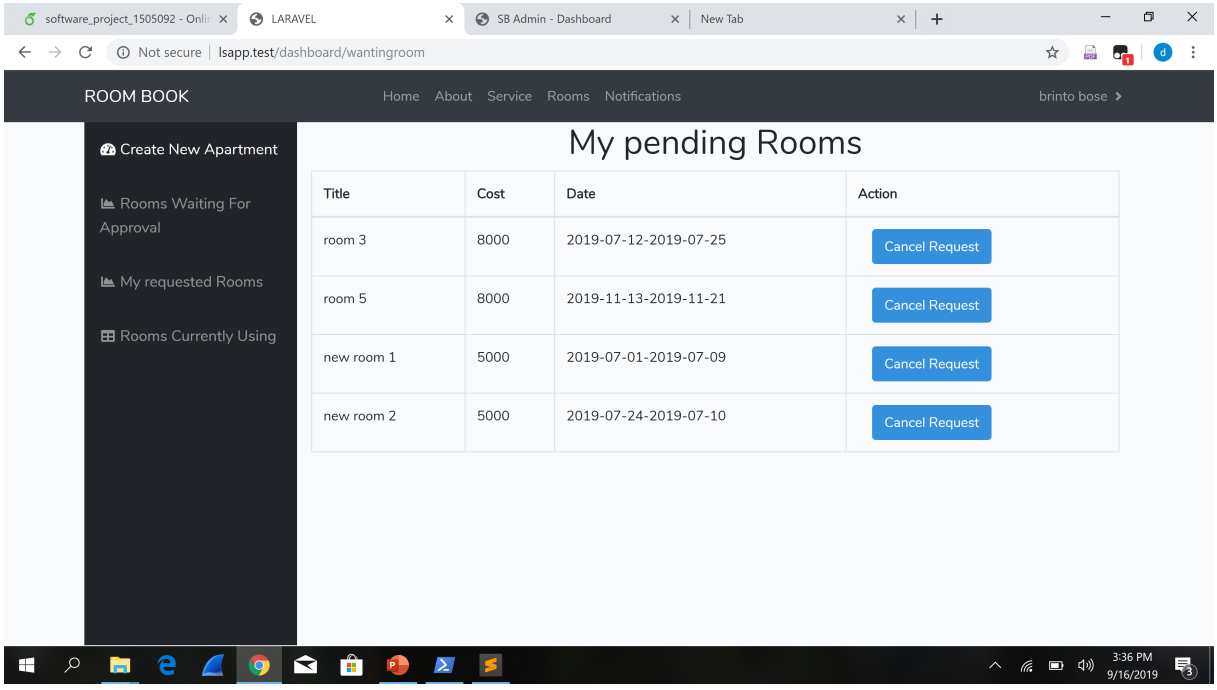

Figure 19: Cancel user request room

- **Admin side User Confirmation**
  - After user signup, the request will be forwarded to admin
  - Admin can manage the **User Requests** in the **Admin Dashboard**
  - The user requests will be displayed in the **Admin Dashboard** in a tabular form

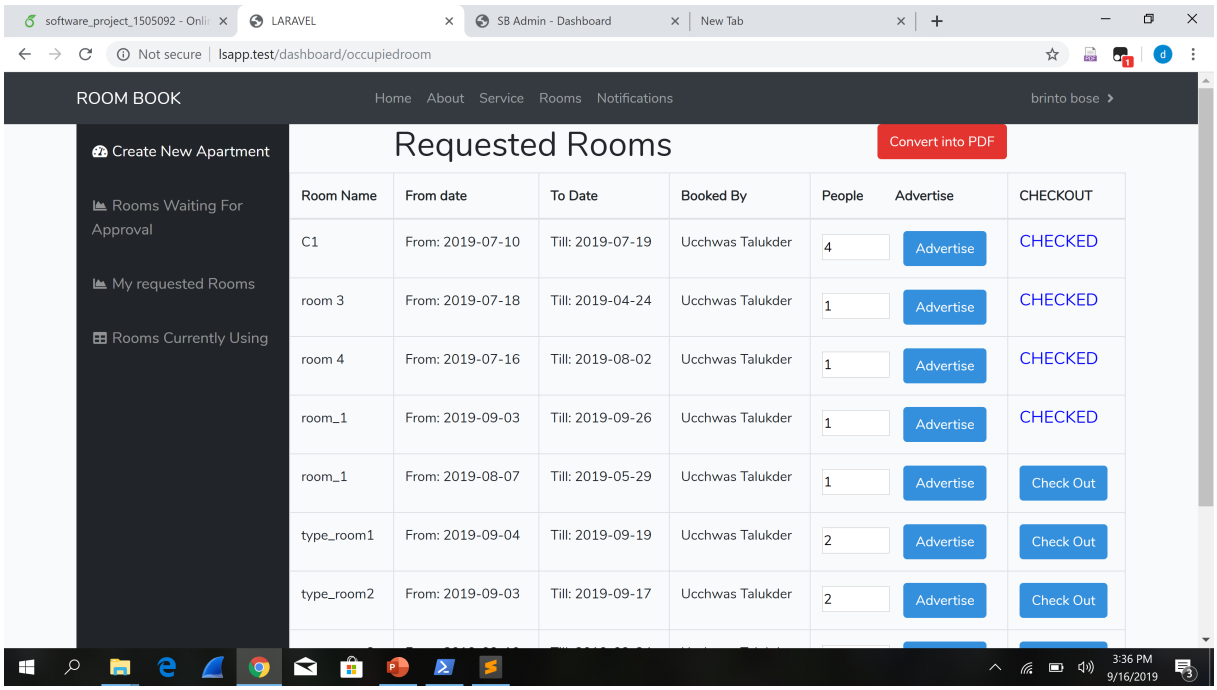

Figure 20: Manage Booked Rooms and Checkout

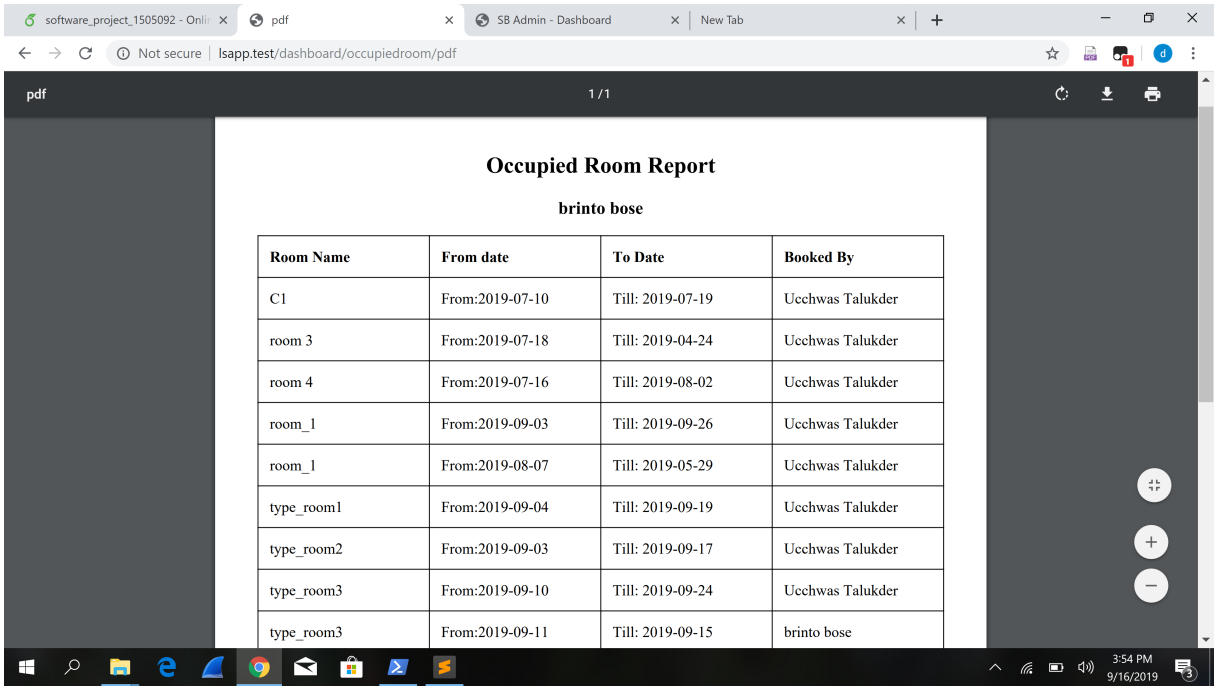

Figure 21: PDF report of Used Rooms

- User can Confirm or Cancel the Request of Users
- If the user is confirmed by admin the **E-MAIL** will send to his account about the confirmation

- **User Individual Report By admin and User Blocking**

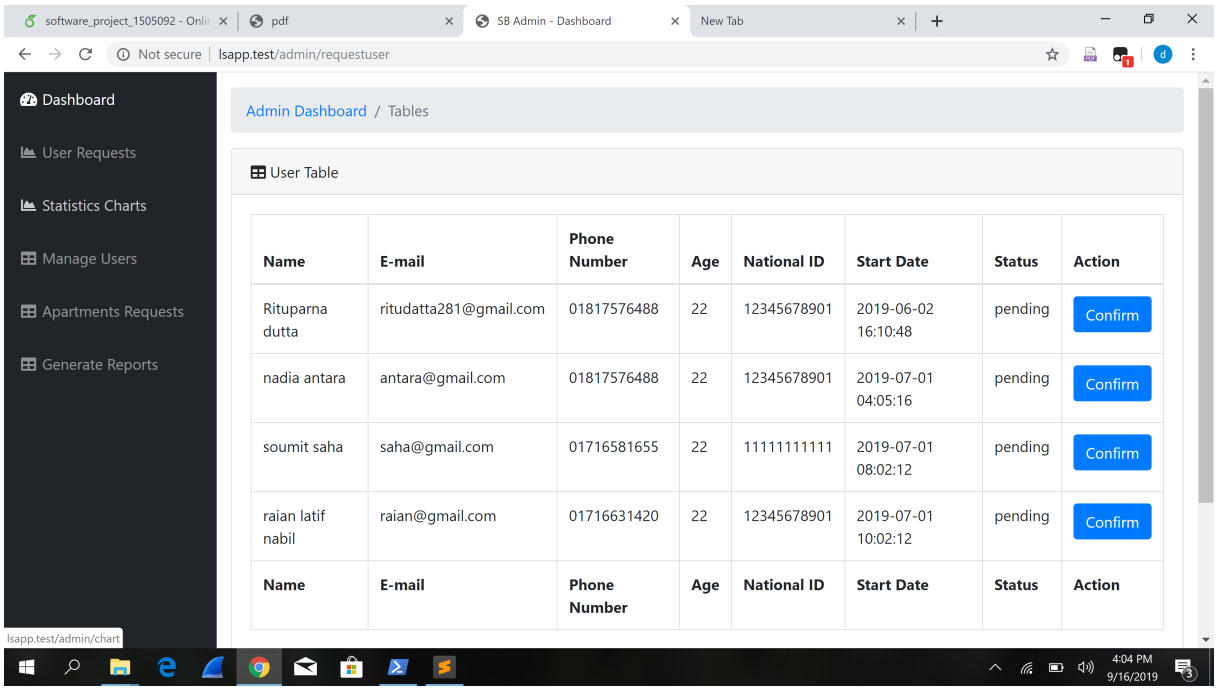

Figure 22: User Confirmation By admin

- Admin can manage the registered users in the **Manage User** window
- Admin can **Block/Unblock** a particular user here
- After blocking that user cant add any apartments
- User can also see individual user apartments information by clicking the **SHOW** button
- There admin can also **pdf report of individual users apartment**

- **Show Statistics Using Charts**
  - Admin can see the statistics of the **Room Type** by a **pie chart**
  - Admin can see the area or location of the apartments **area chart**
  - Admin can see the joining of users by a **Label Chart**

- **Rating Review System** It has a rating review system. The system has 3 way review system.
  - User to Room review
  - User to Owner review
  - Owner to User review

  When a user presses checkout button, the review button comes. If user clicks on the review button, then slides down to ratings and review fields for user and he/she can give this input. When he clicks on submit button, then the field will be disappeared, and the room rating and review are added to the particular flat page. Owner rating

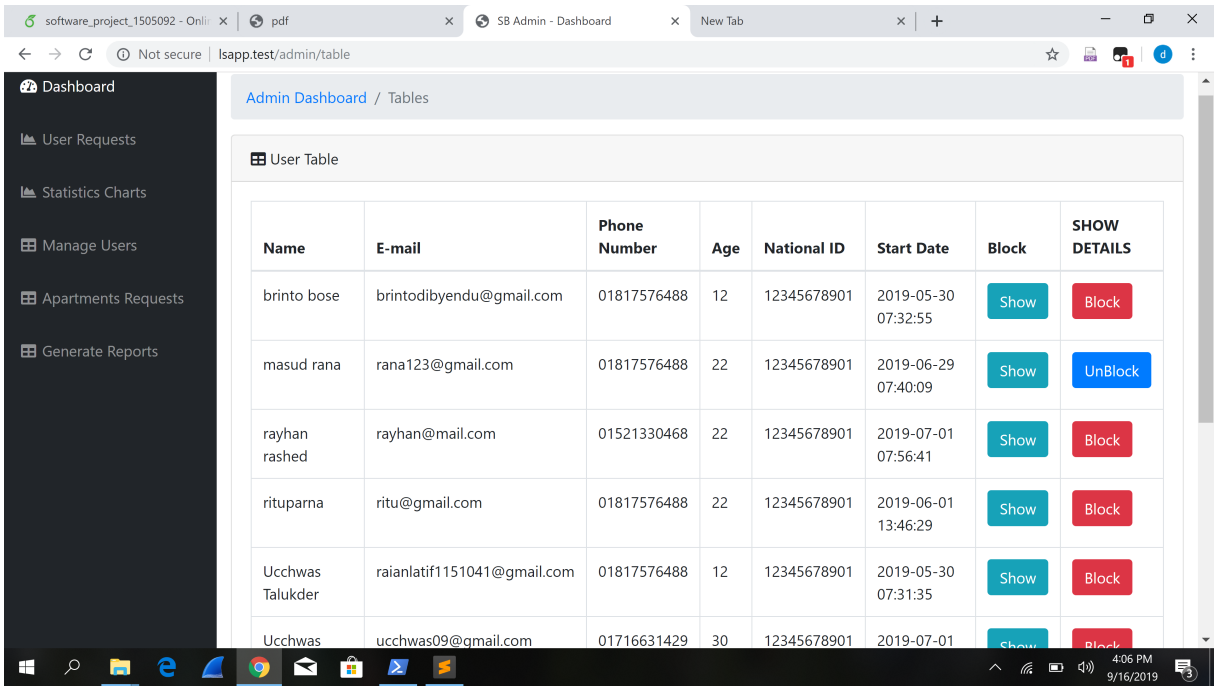

Figure 23: Manage User by Admin

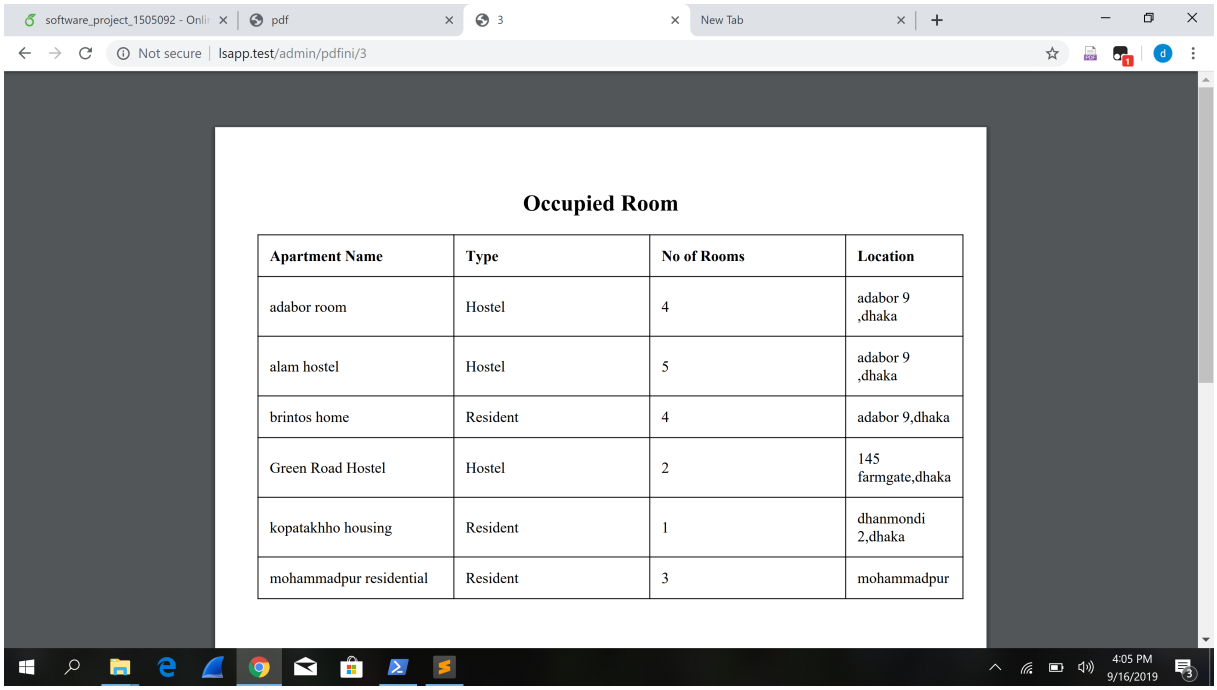

Figure 24: Individual Users Apartment Report

and review are added to the owner profile page. On the other hand, Owner can give review as well. After user checking out from the room, the window will arrive at owner rating page and owner can review and rate the user. the review and rating will be added to the user profile.

- **Notifications** When a host confirms or cancels the booking, user will get a noti-

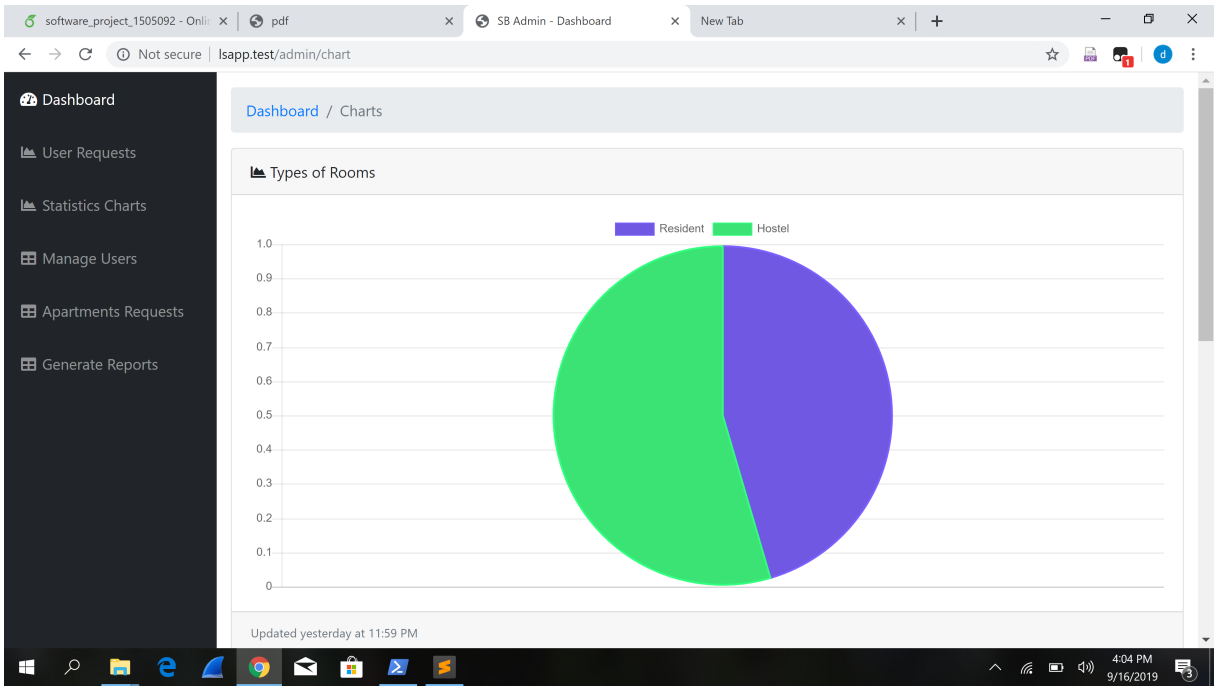

Figure 25: Room Type using PieChart

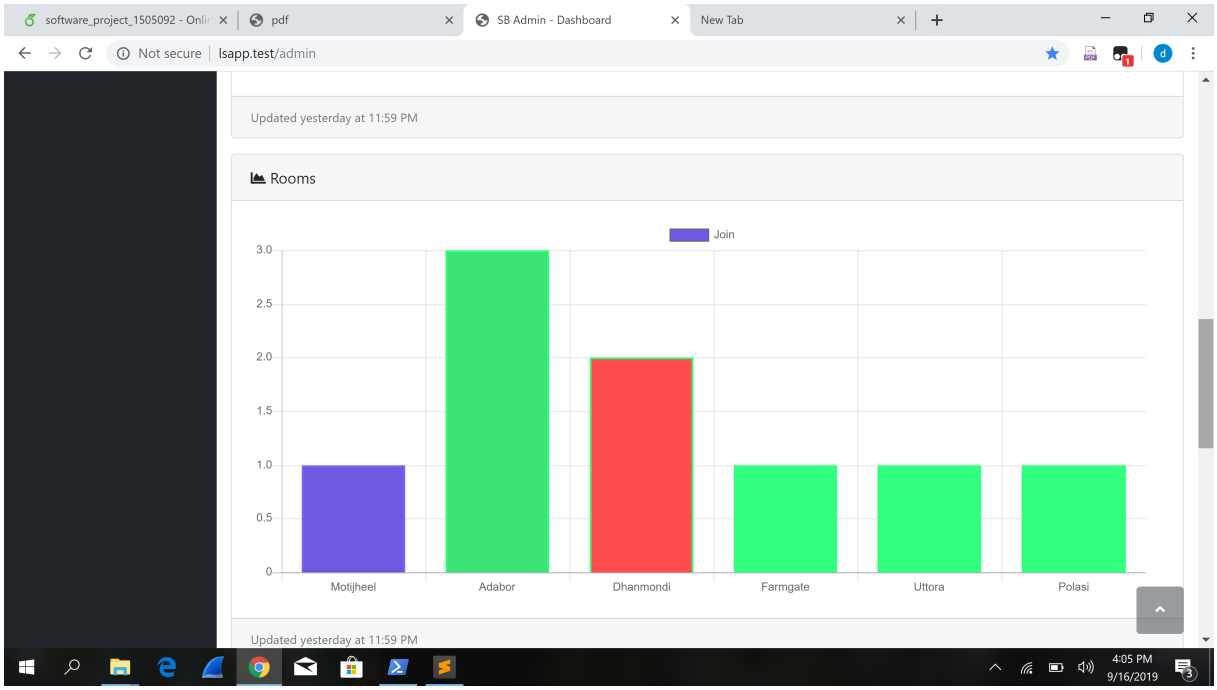

Figure 26: Apartment Location using Label Chart

fication of confirming or cancelling his request to the rooms. When User checkout from the room, host will get a notification as well so that he can advertise the room.

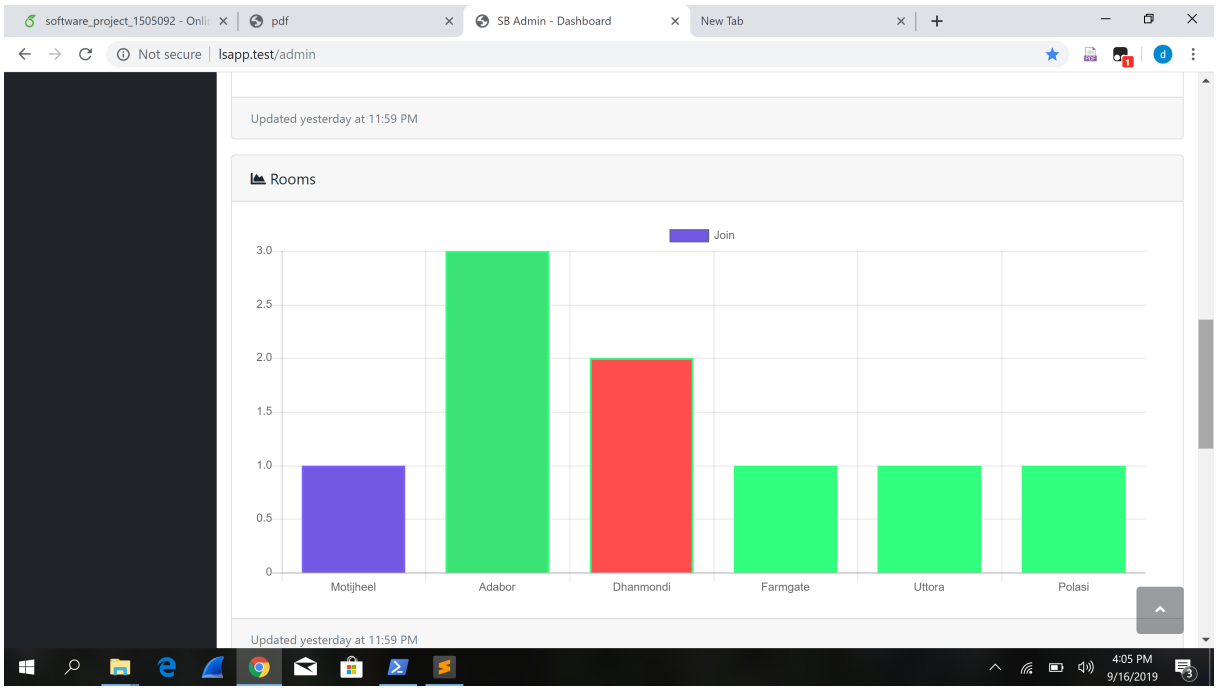

Figure 27: User Joining Date using Area chart

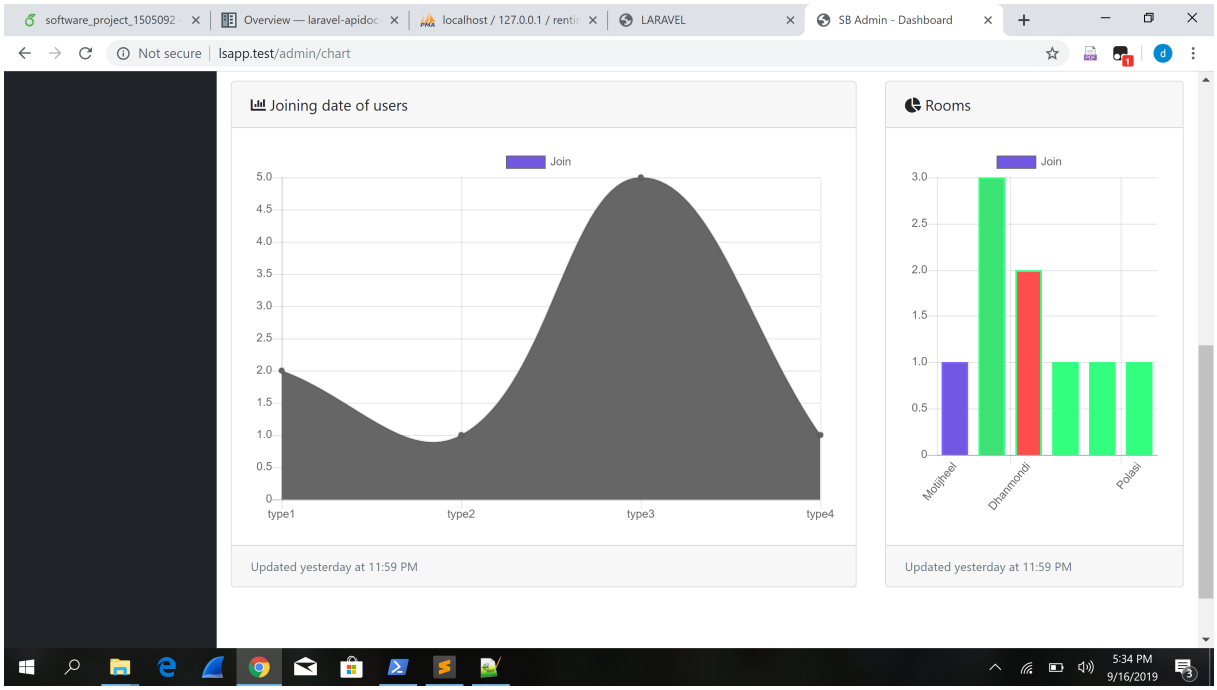

Figure 28: User Joining Date using Area chart

5 Iteration Plan

- **Iteration #1**
  - Objectives The purpose of this iteration is to ensure basic authentication and registering into the system and creating apartments of hosts. after this itera-

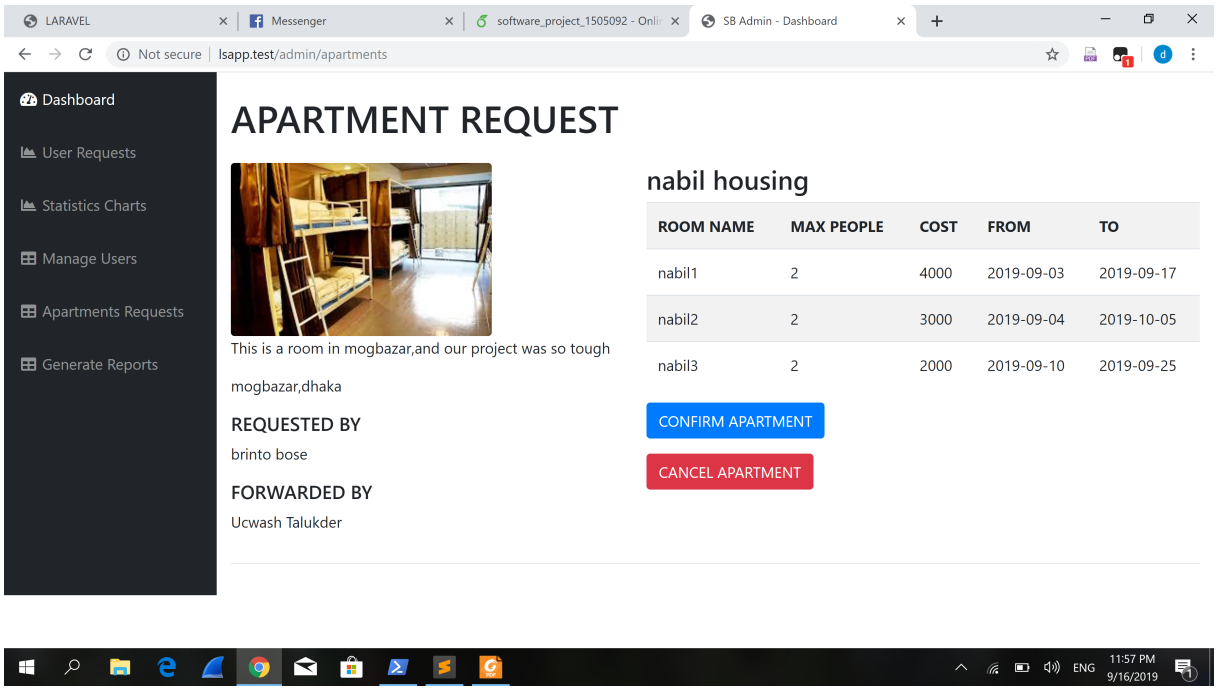

Figure 29: Request to admin about Apartment Confirmation

tion,

  * User and host both can register and login
  * Host can create apartment
  * User can see the apartments

- Use Cases Use cases are implemented
  * User and host login
  * Register
  * Create Apartment
  * Showing Apartment
  * Showing Dashboard

- **Iteration #2**
  - **Objectives** The purpose of this iteration is to ensure edit the apartments, deleting and showing the rooms.
    * host can edit apartments
    * Host can add rooms
    * User can see the rooms of apartments
    * Host can delete apartments
  - **Use Cases** Use cases are following
    * Edit Apartment
    * Delete Apartment
    * Add rooms

    * Show Rooms

- **Iteration #3**
  - **Objectives** In this iteration, user can book rooms, Host can confirm booking or cancel booking watching his/her profile. User can also cancel the request. So after this iterations, User can
    * book rooms
    * Cancel the request

    Host can
    * Confirm request
    * Cancel request
  - **Use Cases**
    * Book room
    * Confirm Booking
    * Cancel Booking
    * Cancel request
    * Approval rooms list
    * Occupied rooms of hosts

- **Iteration #4**
  - **Objectives** In this iteration, admin and User will get notifications upon confirm and cancel the booking request.
    * Admin charts
    * User gets notifications
    * User can search rooms
  - **Use Cases**
    * Area chart
    * Room type Pie chart
    * notifications
    * label chart
    * Search rooms

- **Iteration #5**
  - **Objectives** User can
    * Search according to choices given by the users
    * checkout on the last date they stayed
    * give rating to rooms
    * give rating to owner
  - **Use Cases**
    * Advanced search

        * checkout
        * review
        * rating

- **Iteration #6**
    - **Objectives** The purpose of this iteration is to ensure to make recommendation system according to the total cost and total rating parametre of the apartment.Host can review and rate the users as well. after this iteration
        * Recommending rooms
        * Host giving ratings
    - **Use Cases**
        * Recommendation System
        * Ratings and Reviews to User

- **Iteration #7**
    - **Objectives** After checking out, host can advertise rooms with people left there. Review and ratings added to the profile of host. Rooms avaiability are shown. So
        * Multiple booking of same room
        * Host and user profile with review and ratings
        * Showing room availability
        * Advertising rooms
    - **Use Cases**
        * Multiple booking
        * Show availability
        * Advertise

- **Iteration #8**
    - **Objectives** the purpose is to delete individual rooms, authentication of user and apartment, block and unblocking of users and report generating, So admin can
        * authenticate user and apartment
        * block unblock user and host

      host can
        * delete the rooms

6 Functional Description

Overall functional description for each function in the controllers is given below-

7 Conclusion

For this project, we have interactions with the people of different areas in Dhaka city and we got a huge response. As this is a solution of a certain problem in our prospective, people take this positively. Now the primary target of this project is the job seekers and the students who participate in the admission tests in Dhaka. We can plan it further. Our future plan is to connect with he other people of different profession who comes to Dhaka frequently and the hosts who are willing to give others sublet.We have implemented it as a website, our future plan is to launch a mobile application of this website. Now we are working it in Dhaka, we will try to connect people from outside of Dhaka. Our service will spread throughout Bangladesh. As this a kind of concept of Air bnb and it is new in Bangladesh, we hope people will accept it and admire our efforts in near future

NAV

Info

Welcome to the generated API reference. Get Postman Collection

general

services

Example request:

```
const url = new URL("http://localhost/services");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (500):

```
{
    "message": "Server Error"
}
```

HTTP Request

`GET services`

NAV

Display the information of the Host

Example request:

```
const url = new URL("http://localhost/pos/1");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user": "libero",
    "review": "ipsa"
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET pos/{id}`

Body Parameters

Parameter	Type	Status	Description
user	find	optional	the user who is the owner of that room

NAV

Parameter	Type	Status	Description
review	the	optional	review that the host giving

Normal Search

Normal search function to maintain the normal search field of the View page.

Based on Apartment name and Location of the apartment

Example request:

```
const url = new URL("http://localhost/search");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "products": "porro",
    "posts": "officiis",
    "users": "ullam"
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (500):

```
{
    "message": "Server Error"
}
```

NAV

HTTP Request

```
GET search
```

Body Parameters

Parameter	Type	Status	Description
products	select	optional	the id for recommendation system.
posts	is	optional	the collection of the post that is generate by the query.
users	is	optional	defined taht whther the user is blocked or not

Advance Search

Advance search function to maintain the advance search field of the Home page.

Example request:

```
const url = new URL("http://localhost/advancesearch");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "namesearch": "quis",
    "areasearch": "quaerat",
    "is_family": "odio",
    "is_friend": "harum",
    "$pet": "impedit",
    "student": "dolores",
    "job_seeker": "adipisci",
    "late_night": "illo"
}

fetch(url, {
    method: "GET",
```

NAV

```
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (500):

```
{
    "message": "Server Error"
}
```

HTTP Request

`GET advancesearch`

Body Parameters

Parameter	Type	Status	Description
namesearch	required	optional	The title of the search.
areasearch	string	required	The area name of the search.
is_family	string	optional	The family of checkbox
is_friend	string	optional	The friend of checkbox
$pet	string	optional	The friend of checkbox
student	string	optional	The student of checkbox
job_seeker	string	optional	The job_seeker of checkbox
late_night	string	optional	The late_night of checkbox

To track the Checkout hostory of the user

NAV

Example request:

```
const url = new URL("http://localhost/usingroom/1/1");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "room_rating": "ullam",
    "post": "perspiciatis",
    "room_reveiew": "eaque",
    "status": "similique",
    "room_id": 13,
    "user_name": "laboriosam",
    "avg": "vitae",
    "user_average_rating": 15,
    "user_indi_review": "veniam",
    "user_id": 17,
    "contact": "ullam",
    "cover_image": "quo",
    "family": "unde"
}

fetch(url, {
    method: "POST",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST usingroom/{id}/{id1}`

Body Parameters

Parameter	Type	Status	Description
room_rating	check	optional	whther all the variables are correctly given
post	string	optional	collection of the room.
room_reveiew	string	optional	review given to taht room

NAV

Parameter	Type	Status	Description
status	string	optional	Detail description of the apartment
room_id	integer	optional	id of the user who giving the rating
user_name	string	optional	name of the user who gives rating
avg	string	optional	calculate the avg rating of the current user
user_average_rating	integer	optional	average rating given from the user
user_indi_review	string	optional	Individual review of the user
user_id	integer	optional	the id of the user that givinf the rating
contact	string	optional	contact no of the apartment
cover_image	string	optional	picture of the cover image of the apartment
family	family	optional	is allowed or not

Book Room

Request for booking a room to the owner of that room also update the necessary information in the database

Example request:

```
const url = new URL("http://localhost/dpractice/1");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "post": "ea",
    "request_from_date": "nesciunt",
    "request_to_date": "repellendus",
    "body": "qui",
    "booking": "et",
```

NAV

```
    "user_name": "est",
    "rpname": "exercitationem"
}

fetch(url, {
    method: "POST",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST dpractice/{id}`

Body Parameters

Parameter	Type	Status	Description
post	collection	optional	list of the books that is necessary for the booking
request_from_date	date	optional	input for the given from date.
request_to_date	date	optional	input for the given to date.
body	string	optional	Detail description of the apartment
booking	string	optional	decide whther the room is booked or not.
user_name	string	optional	name of the owner of the apartment
rpname	the	optional	name of the consecutive apartment of that room

Give review to a specefic room for an apartment

Example request:

```
const url = new URL("http://localhost/dashboard/usingroom/1/1");
```

NAV

```
let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "room_rating": "vel",
    "post": "culpa",
    "room_reveiew": "et",
    "status": "maxime",
    "room_id": 18,
    "user_name": "quibusdam",
    "avg": "rerum",
    "user_average_rating": 2,
    "user_indi_review": "nesciunt",
    "user_id": 9,
    "contact": "accusantium",
    "cover_image": "eum",
    "family": "omnis"
}

fetch(url, {
    method: "POST",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST dashboard/usingroom/{id}/{id1}`

Body Parameters

Parameter	Type	Status	Description
room_rating	check	optional	whther all the variables are correctly given
post	string	optional	collection of the room.
room_reveiew	string	optional	review given to taht room
status	string	optional	Detail description of the apartment
room_id	integer	optional	id of the user who giving the rating
user_name	string	optional	name of the user who gives rating
avg	string	optional	calculate the avg rating of the current user

NAV

Parameter	Type	Status	Description
user_average_rating	integer	optional	average rating given from the user
user_indi_review	string	optional	Individual review of the user
user_id	integer	optional	the id of the user that givinf the rating
contact	string	optional	contact no of the apartment
cover_image	string	optional	picture of the cover image of the apartment
family	family	optional	is allowed or not

Give review to a specefic user

Example request:

```
const url = new URL("http://localhost/dashboard/own/1");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user_rating": "dolorem",
    "post": "itaque",
    "user_reveiew": "odio",
    "status": "excepturi",
    "user_id": 19,
    "user_name": "voluptas",
    "avg": "eum",
    "user_average_rating": 10,
    "user_indi_review": "repellendus",
    "cost_basis": "consequatur",
    "contact": "voluptas",
    "cover_image": "quia",
    "family": "incidunt",
    "friends": "provident",
    "pet_allow": "voluptatibus",
    "student": "eveniet",
```

NAV

```
    "job_seeker": "voluptatum"
}

fetch(url, {
    method: "POST",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST dashboard/own/{id}`

Body Parameters

Parameter	Type	Status	Description
user_rating	check	optional	whther all the variables are correctly given
post	string	optional	collection of the room.
user_reveiew	string	optional	review given to taht room
status	string	optional	Detail description of the apartment
user_id	integer	optional	id of the user who giving the rating
user_name	string	optional	name of the user who gives rating
avg	string	optional	calculate the avg rating of the current user
user_average_rating	integer	optional	average rating given from the user
user_indi_review	string	optional	Individual review of the user
cost_basis	string	optional	Payment type of apartment
contact	string	optional	contact no of the apartment
cover_image	string	optional	picture of the cover image of the apartment
family	family	optional	is allowed or not
friends	string	optional	friend is allowed or not
pet_allow	string	optional	pet is allowed or not

NAV

Parameter	Type	Status	Description
student	string	optional	pet is allowed or not
job_seeker	string	optional	pet is allowed or not

Display the information of the User

Example request:

```
const url = new URL("http://localhost/dashboard/want/1");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user": "asperiores",
    "review": "nihil"
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

NAV

HTTP Request

```
GET dashboard/want/{id}
```

Body Parameters

Parameter	Type	Status	Description
user	find	optional	the user who is the owner of that room
review	the	optional	review that the user giving

NAV

Info

Welcome to the generated API reference. Get Postman Collection

general

show indi users

Display users of the system.

Example request:

```
const url = new URL("http://localhost/admin/showuser/1");

    let params = {
            "id": "3",
            "type": "Resident",
            "created_at": "aliquid",
            "location": "earum",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "posts1": 7,
    "posts2": 10,
    "room1": 7,
    "room2": 17,
    "room3": 4
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
```

NAV

```
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/showuser/{id}`

Body Parameters

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description
id	optional	int required id of user.
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment

admin/pdfini/{id}

NAV

Example request:

```
const url = new URL("http://localhost/admin/pdfini/1");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/pdfini/{id}`

---

admin/table/pdf

Example request:

```
const url = new URL("http://localhost/admin/table/pdf");

let headers = {
    "Accept": "application/json",
```

NAV

```
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/table/pdf`

---

show chart

Display charts.

Example request:

```
const url = new URL("http://localhost/admin/chart");

    let params = {
            "id": "3",
            "type": "Resident",
            "created_at": "sit",
            "location": "nisi",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
```

NAV

```
    "Accept": "application/json",
}

let body = {
    "posts1": 14,
    "posts2": 16,
    "room1": 12,
    "room2": 13,
    "room3": 20
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/chart`

Body Parameters

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description

NAV

Parameter	Status	Description
id	optional	int required id of user.
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment

create

Show the form for creating a new chart.

Example request:

```
const url = new URL("http://localhost/admin/table");

    let params = {
            "id": "3",
            "type": "Resident",
            "created_at": "quam",
            "location": "corrupti",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "posts1": 3,
    "posts2": 9,
    "room1": 20,
    "room2": 1,
    "room3": 15
}

fetch(url, {
    method: "GET",
    headers: headers,
```

NAV

```
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/table`

Body Parameters

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description
id	optional	int required id of user.
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment

NAV

showrequestedapartments

Display requested apartment of users.

Example request:

```
const url = new URL("http://localhost/admin/apartments");

    let params = {
            "type": "Resident",
            "created_at": "enim",
            "location": "asperiores",
            "admin": "pending",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "posts1": 13,
    "posts2": 18,
    "room1": 10,
    "room2": 14,
    "room3": 11
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

NAV

HTTP Request

`GET admin/apartments`

Body Parameters

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment
admin	optional	string required status of request.

admin/reports

Example request:

```
const url = new URL("http://localhost/admin/reports");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}
```

NAV

```
fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/reports`

---

request user

Display pending requests of users.

Example request:

```
const url = new URL("http://localhost/admin/requestuser");

    let params = {
            "id": "3",
            "type": "Resident",
            "created_at": "porro",
            "location": "consequuntur",
            "admin": "pending",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}
```

NAV

```
let body = {
    "posts1": 16,
    "posts2": 9,
    "room1": 6,
    "room2": 10,
    "room3": 7
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/requestuser`

Body Parameters

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description
id	optional	int required id of user.

NAV

Parameter	Status	Description
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment
admin	optional	string required status of request.

index

Display a listing of the resource.

Example request:

```
const url = new URL("http://localhost/admin");

    let params = {
            "type": "Resident",
            "created_at": "autem",
            "location": "cumque",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "posts1": 15,
    "posts2": 14,
    "room1": 17,
    "room2": 15,
    "room3": 6
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
```

NAV

```
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin`

Body Parameters

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment

create

Show the form for creating a new chart.

NAV

Example request:

```
const url = new URL("http://localhost/admin/create");

    let params = {
            "id": "3",
            "type": "Resident",
            "created_at": "voluptatem",
            "location": "est",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "posts1": 8,
    "posts2": 5,
    "room1": 11,
    "room2": 12,
    "room3": 19
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (200):

```
null
```

HTTP Request

`GET admin/create`

Body Parameters

NAV

Parameter	Type	Status	Description
posts1	integer	required	count of hostel type
posts2	integer	required	count of resident type
room1	integer	required	count of location type 1
room2	integer	required	count of location type 2
room3	integer	required	count of location type 3

Query Parameters

Parameter	Status	Description
id	optional	int required id of user.
type	optional	string required type of apartment.
created_at	optional	date required date of the creation
location	optional	string required type of apartment

Store a newly created resource in storage.

Example request:

```
const url = new URL("http://localhost/admin");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

NAV

HTTP Request

`POST admin`

Display the specified resource.

Example request:

```
const url = new URL("http://localhost/admin/1");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response:

```
null
```

HTTP Request

`GET admin/{admin}`

NAV

Show the form for editing the specified resource.

Example request:

```
const url = new URL("http://localhost/admin/1/edit");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response:

```
null
```

HTTP Request

`GET admin/{admin}/edit`

Update the specified resource in storage.

Example request:

NAV

```
const url = new URL("http://localhost/admin/1");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "PUT",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

```
PUT admin/{admin}

PATCH admin/{admin}
```

Remove the specified resource from storage.

Example request:

```
const url = new URL("http://localhost/admin/1");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "DELETE",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

NAV

```
DELETE admin/{admin}
```

confirmuser

admin confirm user

Example request:

```
const url = new URL("http://localhost/admin/confirmuser/1");

    let params = {
            "id": "doloremque",
            "admin": "YES",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

```
POST admin/confirmuser/{id}
```

Query Parameters

Parameter	Status	Description
id	optional	int required id of aparment
admin	optional	string required status of request.

NAV

confirroomadmin

admin confirm host room

Example request:

```
const url = new URL("http://localhost/admin/confirmroom/1");

    let params = {
            "id": "quis",
            "isapproved": "CONFIRM",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST admin/confirmroom/{id}`

Query Parameters

Parameter	Status	Description
id	optional	int required id of aparment
isapproved	optional	string required status of request.

cancelroomadmin

NAV

admin cancels host room

Example request:

```
const url = new URL("http://localhost/admin/cancelroom/1");

    let params = {
            "id": "reprehenderit",
            "isapproved": "CANCEL",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST admin/cancelroom/{id}`

Query Parameters

Parameter	Status	Description
id	optional	int required id of aparment
isapproved	optional	string required status of request.

blockuser

admin can block user

NAV

Example request:

```
const url = new URL("http://localhost/admin/blockuser/1");

    let params = {
            "id": "aut",
            "BLOCK": "1",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST admin/blockuser/{id}`

Query Parameters

Parameter	Status	Description
id	optional	int required id of aparment
BLOCK	optional	int required status flag of request.

unblockuser

admin can unblock user

Example request:

NAV

```
const url = new URL("http://localhost/admin/unblockuser/1");

    let params = {
            "id": "et",
            "BLOCK": "0",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST admin/unblockuser/{id}`

Query Parameters

Parameter	Status	Description
id	optional	int required id of aparment
BLOCK	optional	int required status flag of request.

sendmail

admin sends mail to host for confirmation

Example request:

NAV

```
const url = new URL("http://localhost/video");

    let params = {
            "users": "omnis",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response:

```
null
```

HTTP Request

`GET video`

Query Parameters

Parameter	Status	Description
users	optional	int required id of user

NAV

Info

Welcome to the generated API reference. Get Postman Collection

general

Owner rating page

Host can give ratings to the users after checkout of the user.

Example request:

```
const url = new URL("http://localhost/dashboard/own");

    let params = {
            "status": "dicta",
            "owner_id": "consequatur",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user_id": 11,
    "cnt": 1
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

NAV

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET dashboard/own`

Body Parameters

Parameter	Type	Status	Description
user_id	integer	required	id of the user.
cnt	integer	required	count of the checkouts of that hosts.

Query Parameters

Parameter	Status	Description
status	optional	string required staus of check out.
owner_id	optional	int required id of the owner.

dashboard/advertise/{id}/{id1}/{id2}/{id3}

Example request:

```
const url = new URL("http://localhost/dashboard/advertise/1/1/1/1");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
```

NAV

```
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST dashboard/advertise/{id}/{id1}/{id2}/{id3}`

Confirm Room

Host can confirm room booking of the users

Example request:

```
const url = new URL("http://localhost/dashboard/confirmroom/1/1");

    let params = {
            "id": "laudantium",
            "id1": "ut",
            "user_id": "voluptatem",
            "user_name": "dolor",
            "guest_id": "quod",
            "from_date": "ipsam",
            "to_date": "at",
            "status": "quasi",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
```

NAV

```
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST dashboard/confirmroom/{id}/{id1}`

Query Parameters

Parameter	Status	Description
id	optional	int required id of the room.
id1	optional	int required id of the request.
user_id	optional	int optional id of the user
user_name	optional	int optional name of the user
guest_id	optional	int optional id of the guest
from_date	optional	date optional date of booking start
to_date	optional	date optional date of booking ends
status	optional	string optional status of booking of the user

dashboard/checkout/{id}

Example request:

```
const url = new URL("http://localhost/dashboard/checkout/1");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
```

NAV

```
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

```
POST dashboard/checkout/{id}
```

Cancel room

Host can cancel room booking of the users

Example request:

```
const url = new URL("http://localhost/dashboard/cancelroom/1/1");

    let params = {
            "id": "dolorum",
            "id1": "occaecati",
            "from_date": "est",
            "hostid": "eaque",
            "to_date": "incidunt",
            "status": "excepturi",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

NAV

HTTP Request

```
POST dashboard/cancelroom/{id}/{id1}
```

Query Parameters

Parameter	Status	Description
id	optional	int required id of the room.
id1	optional	int required id of the request.
from_date	optional	date optional date of booking start
hostid	optional	id of hosts
to_date	optional	date optional date of booking ends
status	optional	string optional status of booking of the user

Dashboard

Show the application dashboard.

Example request:

```
const url = new URL("http://localhost/dashboard");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

NAV

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET dashboard`

Request Room

Host can confirm room booking of the users

Example request:

```
const url = new URL("http://localhost/dashboard/requestroom");

    let params = {
            "id": "aut",
            "id1": "qui",
            "user_id": "deleniti",
            "user_name": "nesciunt",
            "guest_id": "molestiae",
            "from_date": "sed",
            "to_date": "aut",
            "status": "minus",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
```

NAV

```
    headers: headers,
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):


```
{
    "message": "Unauthenticated."
}
```


HTTP Request

`GET dashboard/requestroom`

Query Parameters

Parameter	Status	Description
id	optional	int required id of the room.
id1	optional	int required id of the request.
user_id	optional	int optional id of the user
user_name	optional	int optional name of the user
guest_id	optional	int optional id of the guest
from_date	optional	date optional date of booking start
to_date	optional	date optional date of booking ends
status	optional	string optional status of booking of the user

Occupied room

Showing occupied room page with the users who occupied host rooms at present

NAV

Example request:

```
const url = new URL("http://localhost/dashboard/occupiedroom");

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user_id": 20
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET dashboard/occupiedroom`

Body Parameters

Parameter	Type	Status	Description
user_id	integer	required	id of the host.

NAV

Useroom

Showing the room currently user is using. After checkout, he can give rating.

Example request:

```
const url = new URL("http://localhost/dashboard/usingroom");

    let params = {
            "requested_by_id": "eum",
            "status": "omnis",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user_id": 12,
    "cc": 18,
    "cc1": 20,
    "cnt": 13,
    "todayDate": "consectetur"
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

NAV |||

HTTP Request

`GET dashboard/usingroom`

Body Parameters

Parameter	Type	Status	Description
user_id	integer	required	id of the user.
cc	integer	required	count of the confirm requests.
cc1	integer	required	count of the checkouts
cnt	integer	required	sum of cc and cc1
todayDate	date	required	date of today

Query Parameters

Parameter	Status	Description
requested_by_id	optional	int required id of the user.
status	optional	string required status of the request.

dashboard/occupiedroom/pdf

Example request:

```
const url = new URL("http://localhost/dashboard/occupiedroom/pdf");

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "GET",
    headers: headers,
})
```

NAV

```
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET dashboard/occupiedroom/pdf`

Cancel Wanting rooms

User can cancel requested room

Example request:

```
const url = new URL("http://localhost/dashboard/cancelwantingroom/1");

    let params = {
            "id": "aut",
            "requested_to_date": "dolor",
            "requested_from_date": "iste",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Accept": "application/json",
    "Content-Type": "application/json",
}

fetch(url, {
    method: "POST",
    headers: headers,
```

NAV

```
})
    .then(response => response.json())
    .then(json => console.log(json));
```

HTTP Request

`POST dashboard/cancelwantingroom/{id}`

Query Parameters

Parameter	Status	Description
id	optional	int required id of the room.
requested_to_date	optional	date required date of checkout.
requested_from_date	optional	date required date of the entry.

Wanting room

Showing users requested rooms

Example request:

```
const url = new URL("http://localhost/dashboard/wantingroom");

    let params = {
            "hostid": "voluptatum",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user_id": 5
}

fetch(url, {
```

NAV

```
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET dashboard/wantingroom`

Body Parameters

Parameter	Type	Status	Description
user_id	integer	required	id of the user.

Query Parameters

Parameter	Status	Description
hostid	optional	int required id of the host.

Notification

showing notifications for the user account

Example request:

NAV

```
const url = new URL("http://localhost/notify");

    let params = {
            "guest_id": "sint",
            "requested_from_date": "eos",
        };
    Object.keys(params).forEach(key => url.searchParams.append(key, params[key]));

let headers = {
    "Content-Type": "application/json",
    "Accept": "application/json",
}

let body = {
    "user_id": 13
}

fetch(url, {
    method: "GET",
    headers: headers,
    body: body
})
    .then(response => response.json())
    .then(json => console.log(json));
```

Example response (401):

```
{
    "message": "Unauthenticated."
}
```

HTTP Request

`GET notify`

Body Parameters

Parameter	Type	Status	Description
user_id	integer	required	id of the user.

Query Parameters

Parameter	Status	Description

NAV

Parameter	Status	Description
guest_id	optional	date required date of user.
requested_from_date	optional	date required date of the entry.