



Dublin Business School

excellence through learning

Web Development for Information Systems Assignment

FastRoomBook Web Application

Module Title: Web Development for Information System (B9IS109)

Module Leader: Mr. Obinna Izima

Student Name: Manik Mahashabde (10518579)

Table of Contents

1. Introduction	5
2. Overview and features.....	5
3. Technical, UX design and framework used.....	7
4. Webservice.....	7
5. Implementation, research, and planning	7
6. Testing.....	13
6.1 Room booking testing.....	13
6.2 Contact us testing.....	15
6.3 Forgot password.....	16
6.4 User registration.....	17
6.5 Room addition by admin with image.....	17
6.6 Room availability status change.....	18
6.7 Room deletion testing.....	19
7. Security vulnerability and measures.....	20
7.1 Password hashing.....	20
7.2 OTP based authentication.....	21
7.3 Session management.....	21
7.4 Validations.....	22
7.5 Deployment to Heroku cloud.....	23
8. Github.....	25
9. Learning outcomes.....	25
References & Bibliography.....	26

List of Tables

Tab. 1	Site Map	Pg. 06
Tab. 2	List of technologies table	Pg. 07

List of Figures

Fig. 1	Folder structure	Pg. 08
Fig. 2	Booking controller	Pg. 08
Fig. 3	Booking service	Pg. 09
Fig. 4	Booking DAO	Pg. 09
Fig. 5	Home page	Pg. 10
Fig. 6	Booking Ajax call	Pg. 11
Fig. 7	User dashboard	Pg. 12
Fig. 8	Admin dashboard	Pg. 12
Fig. 9	User registration validation	Pg. 13
Fig. 10	Custom exceptions	Pg. 13
Fig. 11	Registration form validation	Pg. 13
Fig. 12	Room booking confirmation	Pg. 14
Fig. 13	Room booking email	Pg. 15
Fig. 14	Room booking on admin dashboard	Pg. 15
Fig. 15	Contact us Page	Pg. 16
Fig. 16	Contact us Email	Pg. 16
Fig. 17	Contact us booking on admin dashboard	Pg. 17
Fig. 18	Forgot password	Pg. 17
Fig. 19	Contact us and OTP mail	Pg. 18
Fig. 20	OTP page	Pg. 18
Fig. 21	Password changed	Pg. 19
Fig. 22	Registration Email	Pg. 19
Fig. 23	Add room Page	Pg. 19
Fig. 24	New room addition on the admin page	Pg. 20
Fig. 25	New room addition on the user page	Pg. 20
Fig. 26	Room status changed screenshot	Pg. 20
Fig. 27	Room change reflection on the user page	Pg. 21

Fig. 28	Room deletion testing and result	Pg. 21
Fig. 29	Hashed password storage	Pg. 21
Fig. 30	OTP mail	Pg. 22
Fig. 31	User session created at login	Pg. 22
Fig. 32	User session null at login	Pg. 22
Fig. 33	Description cannot be empty	Fig. 23
Fig. 34	Custom exception codes	Pg. 23
Fig. 35	Password change validation	Pg. 23
Fig. 35	Heroku deployment	Pg. 24
Fig. 36	Heroku deployment build status on webpage	Pg. 25

1: Introduction:

This is an artifact that demonstrates the implementation of a web application from scratch. Right from gathering the requirements, picking out various technologies, making design patterns. Then progressively moving towards implementation of backend API, HTML webpages, AJAX requests. Finally moving towards form validation, backend logic, responsiveness and deployment on Heroku cloud.

The topic I have selected was making a web application about hotel room booking. The name I gave was **FastRoomBook**. There are 2 major entities in my website:

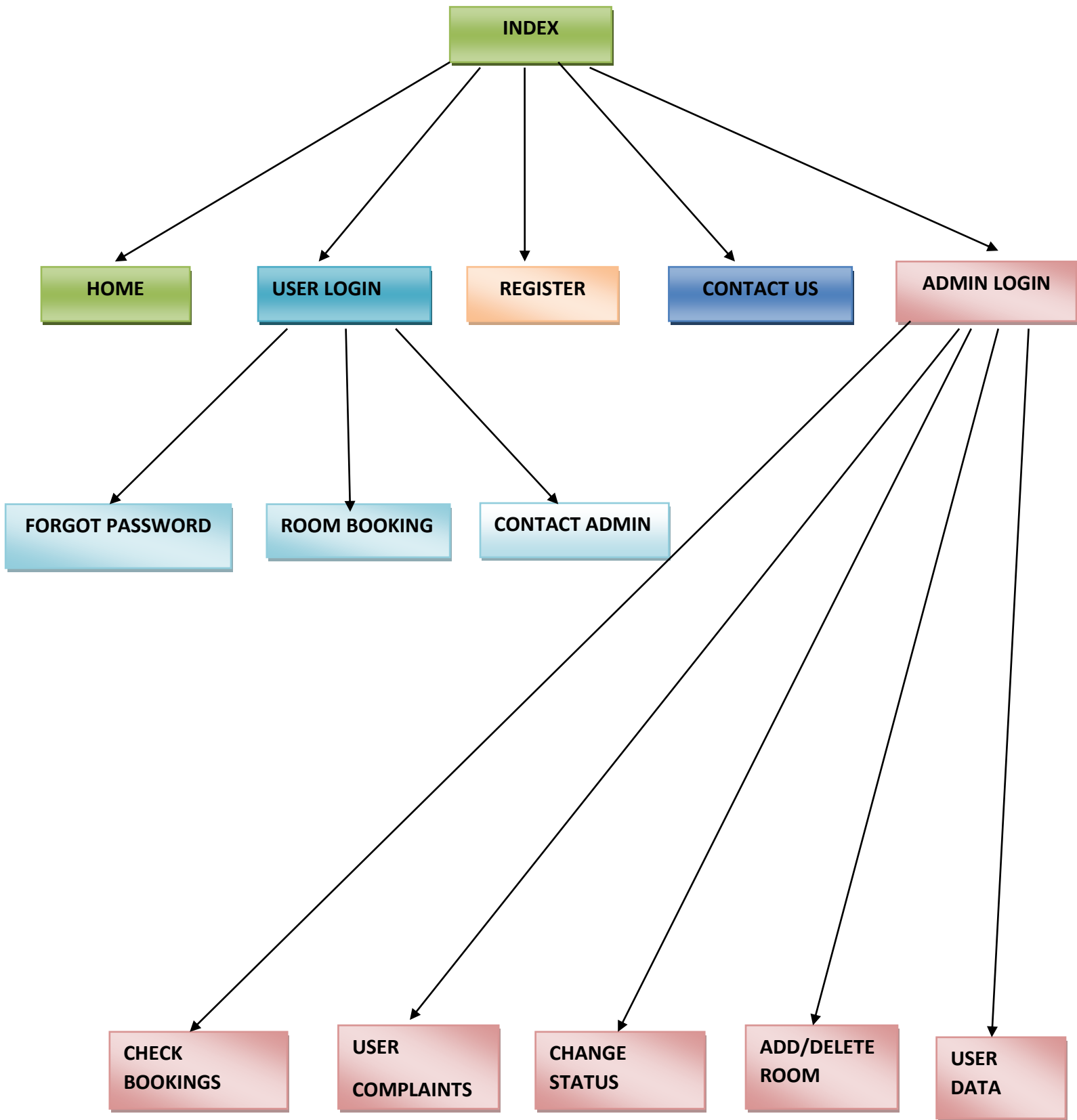
- 1) **User/Customer:-** They have a feature of selecting from the variety of rooms based on their price, Average ratings, facilities, images. Once the customer finalizes they get a mail on their email for room booking confirmation.
- 2) **Admin:-** They are responsible for managing the website. An admin can add and delete rooms. They can change the availability status of the room. Admin can also see the dashboard of booking details, customer complaints, existing customer details.

2: Overview and features:

The overview of the website is as below:-

- 1) First, the application is started on the pycharm at the localhost. After opening the webpage on the browser. A user accesses the home page. The options available are user login, admin login, registration, contact us.
- 2) Users access the user login page. A user login into the portal and book a room. A user can also reset the password with OTP authentication. A new user can also register on the page.
- 3) Admin can access the portal via admin login. After logging in admin see various options of change room status, user complaints, user registration details. Admin is also responsible for adding or deleting rooms. Room images can also be added by the admin.
- 4) Validation of all the forms is also present.
- 5) The Room table consists of details such as room_id, room_number, price, Average Ratings, availability.
- 6) User parameters are user_id, name, username, password, email, contact_no.
- 7) A slider is included on the front page which consists of 4 wide images which automatically keeps swiping.
- 8) There is also a header that is present on the index page, user page and admin page.
- 9) A footer is present at the bottom.
- 10) Various kind of validations such as incorrect email format, incorrect name format, contact number format

Site Map of the website is presented below:



3: Technologies, UX design and framework used:

Below are the details of the software, technology, and tools used along with version.

Software/Technology/Tool	Version
PyCharm and Python (<i>3.8.2 Documentation, 2020</i>)	3.9.9
Pip	20.0.2
HTML (<i>HTML Tutorial, 2020</i>)	5
CSS	3
Bootstrap (<i>Mark Otto, 2020</i>)	4
Javascript (<i>Free JavaScript training, resources and examples for the community, 2020</i>)	ES6
Jquery (<i>js.foundation, 2020</i>)	3.15
PostMan	3.03
Flask (<i>Flask, 2020</i>)	3
MySQL (<i>MySQL :: MySQL Documentation, 2020</i>)	6
Backend Framework	Python flask
Frontend framework	HTML5,CSS3, Javascript, Jquery, Jinja2, bootstrap

Tab 2: List of technologies table

4: Webservice:

Webservice I have used is RestFul web service. In my controller classes, I am using @app.route in which I am using HTTP methods such as GET, POST from my front end AJAX call. Data format I am using is JSON.

5: Implementation, Research, and Planning:

Implementation details are given in the below steps:

- 1) First step was the selection of an information system. I selected hotel management system. The name is used is **FastRoomBook**.
- 2) Second step was gathering requirements and various kinds of functionalities to be included in the information system. I made a database in MySQL. It consists of 6 tables namely customer, admin, query, incident, room and booking.
- 3) After this, I made a rough design on a notebook about the functionalities to be included on the webpage. Then I used HTML, CSS to make the webpages and different components of the webpage such as header, footer, slider, navigation.

- 4) After setting up the basic webpages. I moved on to developing the backend. I divided the backend into 3 modules i.e controllers, services and DAO. Controllers consist of API endpoint, Service consists of business logic and DAO(data access layer) consists of SQL logics.
- 5) Once backend was designed I added custom validation for the website such as incorrect credentials, incorrect email format, unauthorized.

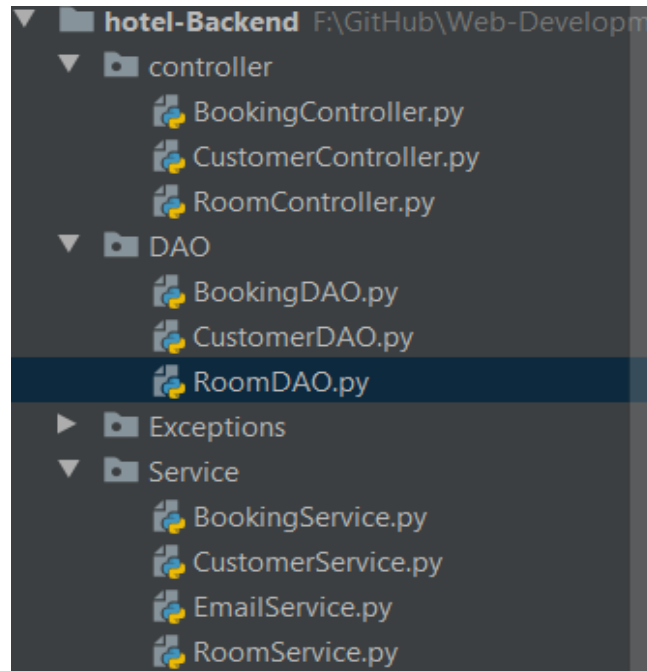


Fig 1: Folder structure

```
@app.route("/addBooking", methods=['POST'])
def addBooking():
    wsResponse = {"resultSet": None, "operationStatus": None}
    try:
        responseData = bookingService.addBooking(request.headers, request.json)
        wsResponse['resultSet'] = responseData
        wsResponse['operationStatus'] = 1
    except RoomNotAvailable:
        wsResponse['resultSet'] = None
        wsResponse['operationStatus'] = CustomUtils.ROOM_NOT_AVAILABLE
    except CustomerNotLoggedIn:
        wsResponse['resultSet'] = None
        wsResponse['operationStatus'] = CustomUtils.CUSTOMER_NOT_LOGGED_IN
    return wsResponse
```

Fig 2: booking controller

6) Below are samples of service and DAO for one of the API

```
@classmethod
def addBooking(cls, header, data):
    if cls.customerService.checkCustomerFromSessionID(header.get('session_id')):
        checkCustomer = cls.customerDAO.checkCustomerFromSessionID(header.get('session_id'))
        if cls.roomService.checkRoomIsAvailable(data.get('room_id')):
            roomData = cls.roomDAO.checkRoomIsAvailable(data.get('room_id'))
            responseData = cls.bookingDAO.addRoomBooking(checkCustomer.get('customer_id'), data.get('room_id'),
                                                         checkCustomer.get('email'))

            cls.emailService.sendEmail(checkCustomer.get('email'), responseData.get('booking_id'),
                                       roomData.get('room_number'), roomData.get('price'),
                                       roomData.get('facilities'), roomData.get('Average_Rating'))
            currentRoomData = cls.roomService.getCurrentRoomData(data.get('room_id'))
            session['currentRoomData'] = currentRoomData
        else:
            raise RoomNotAvailable
    else:
        raise CustomerNotLoggedIn
    return responseData
```

Fig 3: Booking service

```
@classmethod
def addRoomBooking(cls, customer_id, room_id, email):
    try:
        bookingId = str(uuid.uuid4())
        conn = mysql.connect()
        cursor = conn.cursor(pymysql.cursors.DictCursor)

        cursor.execute(
            "insert into booking (booking_id, customer_id, room_id, email) value (%s, %s, %s, %s)",
            (bookingId, customer_id, room_id, email))
        conn.commit()
        cursor.execute("UPDATE room set availability='No' where room_id=%s ",
                       room_id)
        conn.commit()
        cursor.execute("SELECT * from booking r WHERE r.booking_id = %s",
                       bookingId)
        rows = cursor.fetchone()
        return rows
    except Exception as e:
        print(e)
    finally:
        cursor.close()
        conn.close()
```

Fig 4: Booking DAO

7) After completing backend implementation API's related to customer, admin, booking, incident, query, room. Next step was making the index page of the application as shown below.

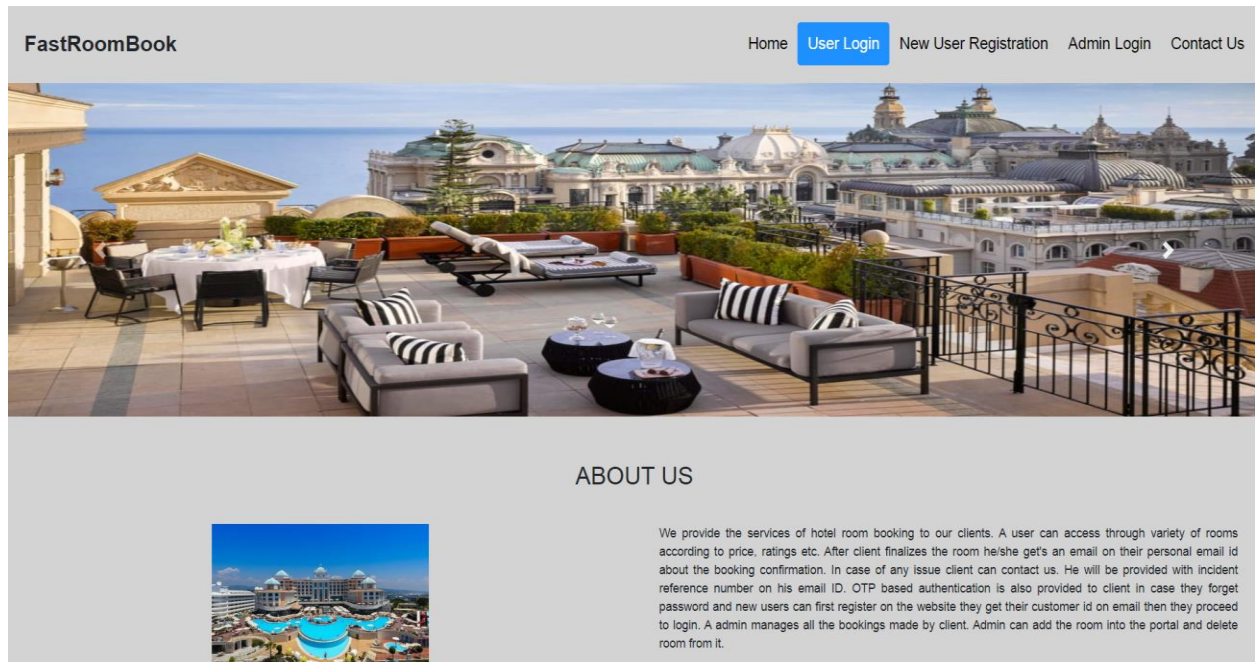


Fig 5: Home page

8) Then I moved on towards using Javascript, jquery for my AJAX calls, form validations as below

```

$("document").ready(function()
{
    $("#sendingData").click(function()
    {
        var head = $("#userSession").val();
        var roomID = $("#RoomBookingId").val();

        $.post({url: "http://127.0.0.1:5000/addBooking",
            headers: {
                session_id:head
            },
            data: JSON.stringify({room_id: roomID}),
            contentType: "application/json",
            success: function(result)
            {
                window.location.href = "http://127.0.0.1:5000/RoomBookingConfirmation"
                console.log(result)
            }
        });
    });
});

```

Fig 6: Booking Ajax call

9) After this designing of dashboards for admin and user as below

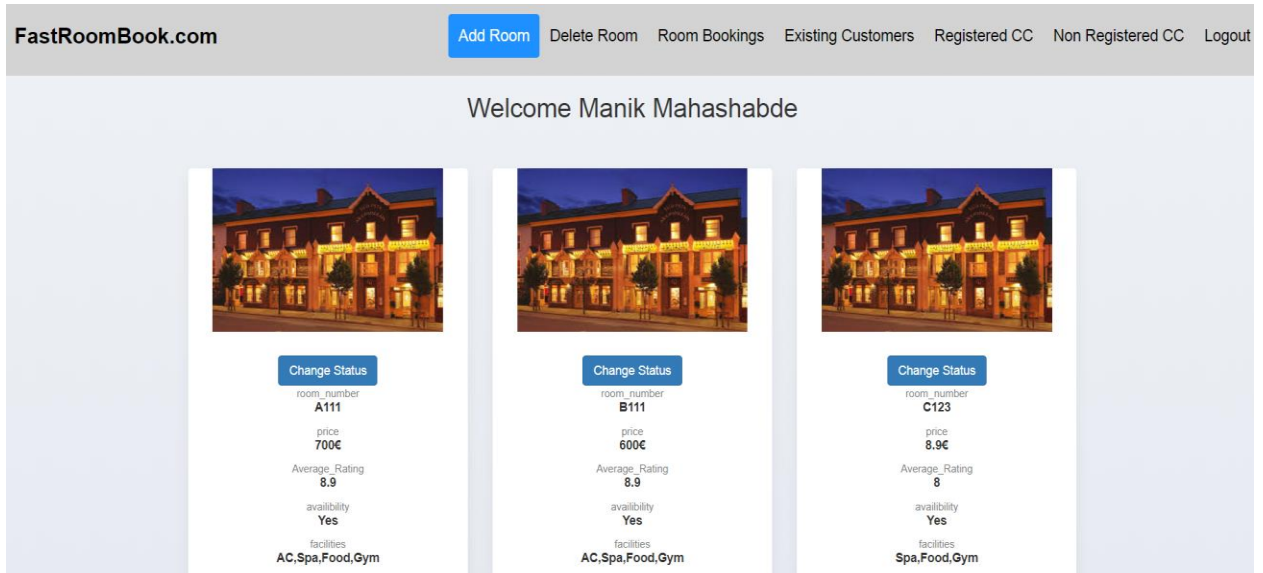


Fig 7: User dashboard

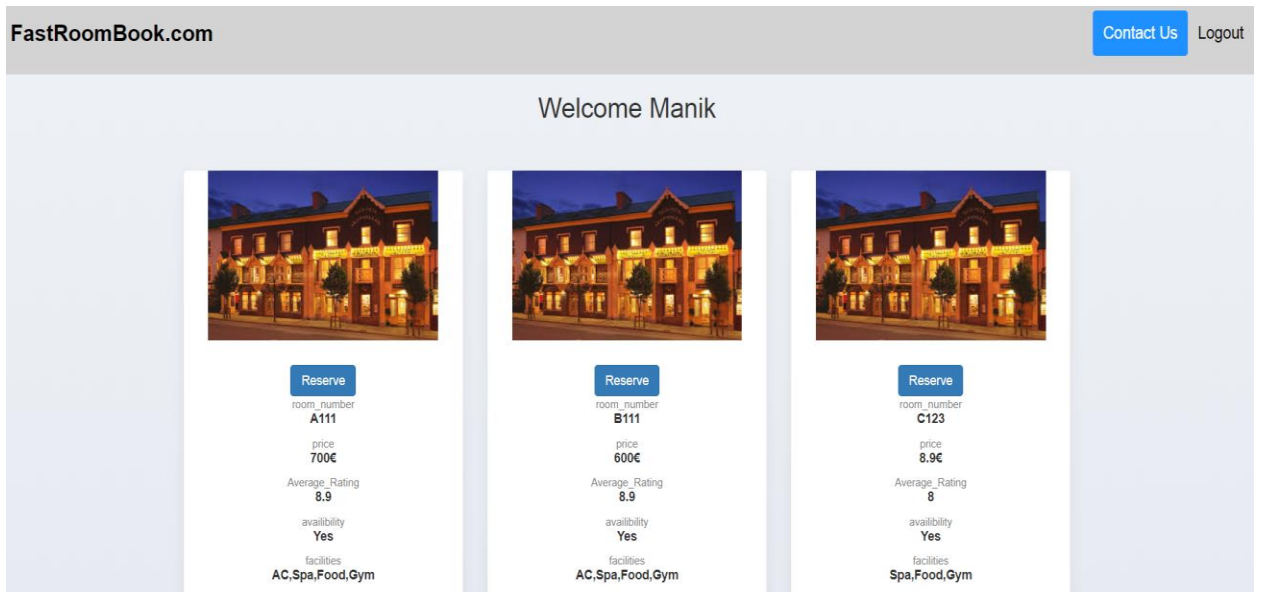


Fig 8: Admin dashboard

10) After this, I added frontend validation w.r.t my backend response code.

```

success: function(result)
{
    console.log(result);
    if (result.operationStatus===-14){
        window.alert('Invalid Email')
    }
    else if(result.operationStatus===-15){
        window.alert('Invalid Contact Number!!')
    }
    else if(result.operationStatus===-19){
        window.alert('Password too short!!')
    }
    else if(result.operationStatus===-16){
        window.alert('Invalid Name!!')
    }
    else if(result.operationStatus===1){
        window.alert('Registration Successful!!')
    }
}
});

```

Fig 9: User registration validation

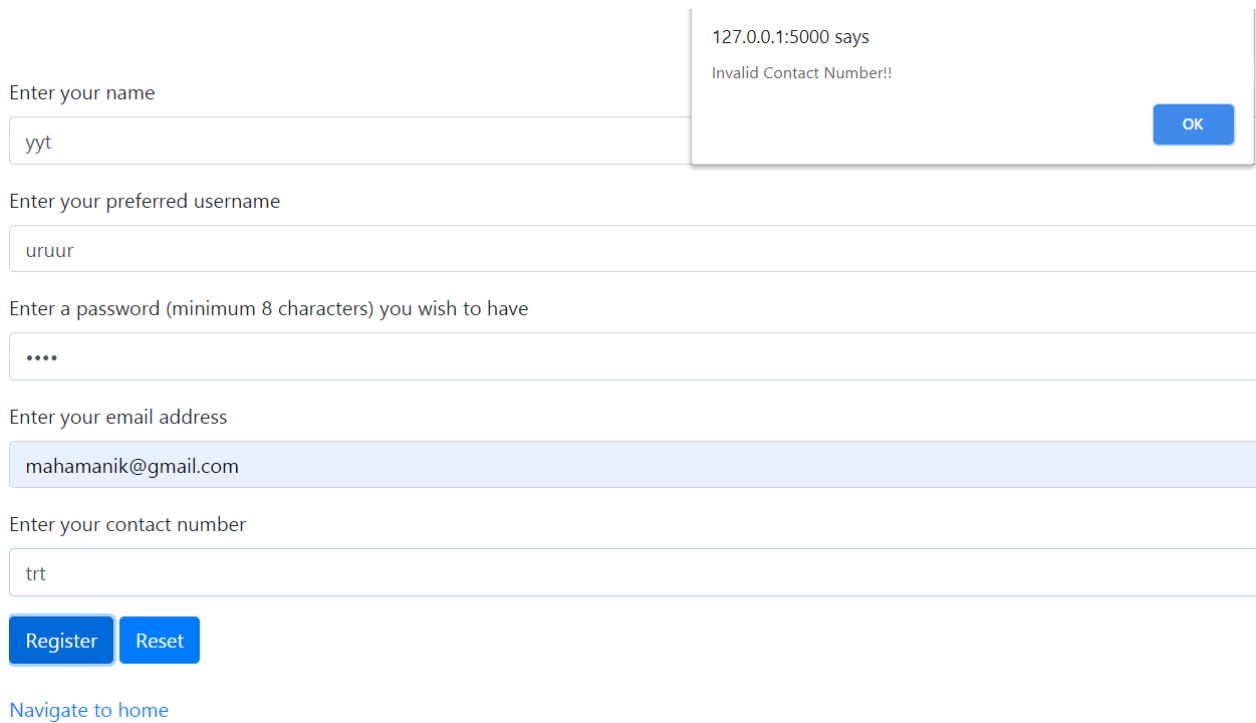
11) Backend validation codes are shown below along with the validation of the user registration. As it can be seen that the contact number should be in numeric format hence it is giving invalid contact number.

```

class CustomUtils:
    SUCCESSFUL = 1
    SOMETHING_WENT_WRONG = -1
    WRONG_CREDENTIALS = -2
    CUSTOMER_DOES_NOT_EXIST = -3
    ROOM_NOT_AVAILABLE = -4
    ROOM_WITH_GIVEN_NUMBER_ALREADY_EXIST = -5
    CUSTOMER_NOT_LOGGED_IN = -6
    EMAIL_SENDING_FAILED = -7
    CUSTOMER_WITH_EMAIL_NOT_EXIST = -8
    OTP_NOT_CORRECT = -9
    NO_BOOKINGS_EXIST = -10
    NO_CUSTOMERS_EXIST = -11
    ROOM_WITH_GIVEN_ID_DOES_NOT_EXIST = -12
    NOT_AUTHORIZED = -13
    INVALID_EMAIL = -14
    INVALID_CONTACT_NUMBER = -15
    INVALID_NAME = -16
    NEW_PASSWORD_SAME_AS_OLD_PASSWORD = -17
    DESCRIPTION_EMPTY = -18
    PASSWORD_TOO_SHORT = -19

```

Fig 10: Custom exceptions



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

- Enter your name: yyt
- Enter your preferred username: uruur
- Enter a password (minimum 8 characters) you wish to have:
- Enter your email address: mahamanik@gmail.com
- Enter your contact number: trt

At the bottom of the form are two buttons: "Register" and "Reset". Below the form is a link: "Navigate to home".

A validation error message box is displayed in the top right corner, containing the text: "127.0.0.1:5000 says Invalid Contact Number!!" and an "OK" button.

Fig 11: Registration form validation

12) Finally, I added the SMTP configuration on hotel room booking by the user.

For more details regarding the implementation please refer to below GitHub URL where I have given details of implementation.

<https://github.com/mahas500/Web-Development-CA>

6: Testing:

a) Room Booking Testing.

Room Booked Successfully

Below are your room details

Facilities:

AC,Spa,Food,Gym

Room Price:

700€

Room Number:

A111

We have sent you a mail regarding your booking. Kindly check it for more details

Fig 12: Room booking confirmation

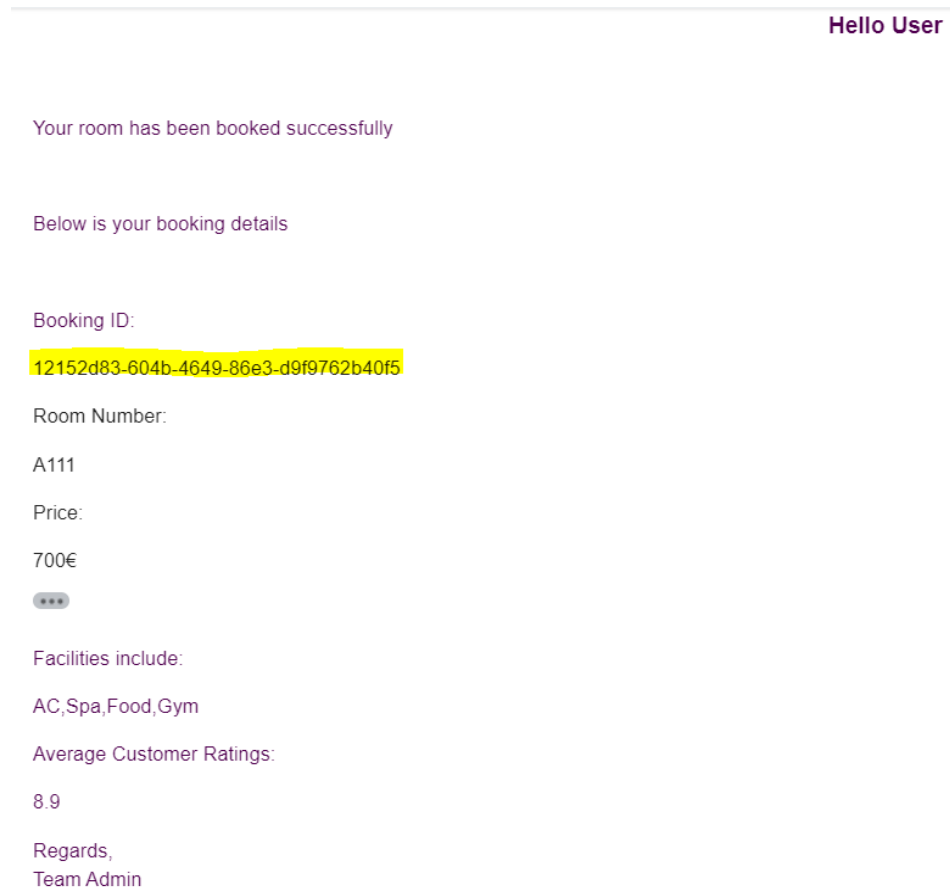


Fig 13: Room booking email

All Bookings Dashboard

Booking ID	Customer ID	Room ID	Email ID
12152b83-604b-4649-86a3-d9f9762b40f5	31f0a4e4-0f35-4d76-b16f-9ffbcd1acd99	32ebf7be-2d04-47c4-bac6-c9bd0cc5f4b7	mahamanik@gmail.com
622f38cd-0fdb-4beb-9e6e-79f8e538d3c7	31f0a4e4-0f35-4d76-b16f-9ffbcd1acd99	b44c978c-dc28-4a2b-9437-d6c85b2639e1	mahamanik@gmail.com

Fig 14: Room booking on admin dashboard

b) Contact us testing

127.0.0.1:5000 says
Mail sent to admin!!!
OK

SubmitReset

Fig 15: Contact us Page

Hello User

Thanks for reaching out to us. We will get back to you shortly. Please find below **incident ID** for reference

fc93d4ca-4b69-4fe8-9dce-6cc241797083

Description: Testing for CA

Regards,
Team Admin

Fig 16: Contact us Email

The message sent by the user appears on the customer complaints heading in admin dashboard as shown below

fc93d4ca-4b69-4fe8-9dce-6cc241797083	31f0a4e4-0f35-4d76-b16f-9ffbcd1acd99	mahas500	Testing for CA
--------------------------------------	--------------------------------------	----------	----------------

[Navigate to Dashboard](#)
[Back to dashboard](#)

Fig 17: Contact us booking on admin dashboard

c) Forgot password

OTP (one- time password) authentication in which OTP is received on email.

Forget Password page

Enter your user ID

31f0a4e4-0f35-4d76-b16f-9ffbcd1acd99

Enter your Email address

mahamanik@gmail.com

Generate OTP

Generate

Reset

[Navigate to home](#)

Fig 18: Forgot password

Hello User

Thanks for reaching out to us. Below is the 6 digit OTP for resetting your password for customer ID

31f0a4e4-0f35-4d76-b16f-9ffbcd1acd99

556d4f

Regards,
Team Admin

Fig 19: Contact us and OTP mail

Enter the OTP received via Email

556d4f6

Enter your New Password

.....

CONFIRM

Reset

127.0.0.1:5000 says

OTP not correct

OK

Fig 20: OTP page

When the user enters the correct OTP password is changed successfully and OTP is again set to null.

Password changed successfully!!!

[Navigate to login Page](#)

[Navigate to home](#)

Fig 21: Password changed

d) User Registration

Once the user fills the details on the registration page. After successful registration user receives the mail as below:

Hello User

You have been registered successfully. Please find below your customer ID

Customer ID: **a05bd779-29b7-4eda-b3a8-61df3b26349d**

Regards,
Team Admin

Fig 22: Registration mail

e) Room addition by admin with image

As can be seen admin logs in and add the room on the add room booking page.

Enter room number

Enter room price

Enter rating for the room out of 10

Enter the facilities

Upload the room picture

Fig 23: Add room Page

Once admin navigates to dashboard again he can see the newly added room C121

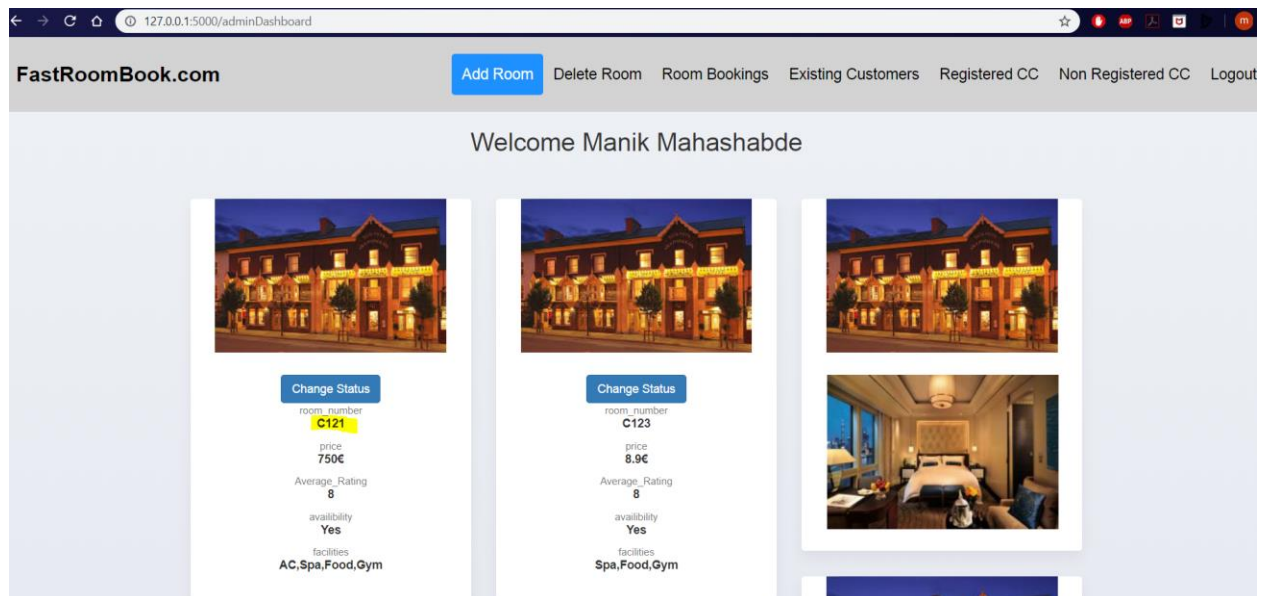


Fig 24: New room addition on the admin page

The same room C121 is also available on user dashboard with availability status as 'Yes'

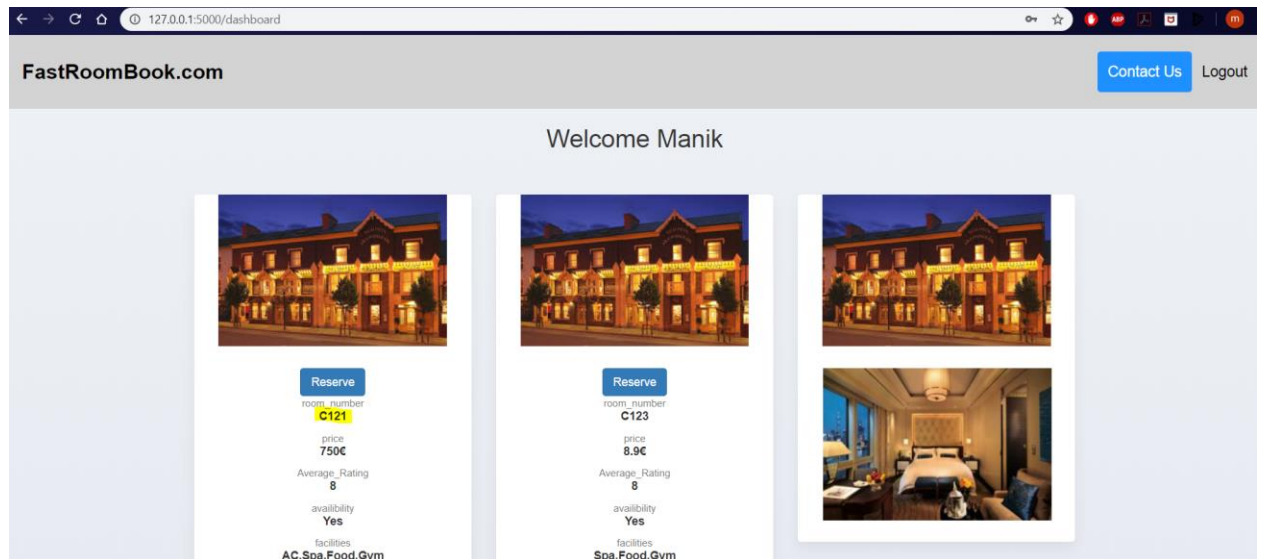


Fig 25: New room addition on the user page

f) Room availability status change

Let us change the availability status to 'No' of room C121 in the admin portal after clicking change status button

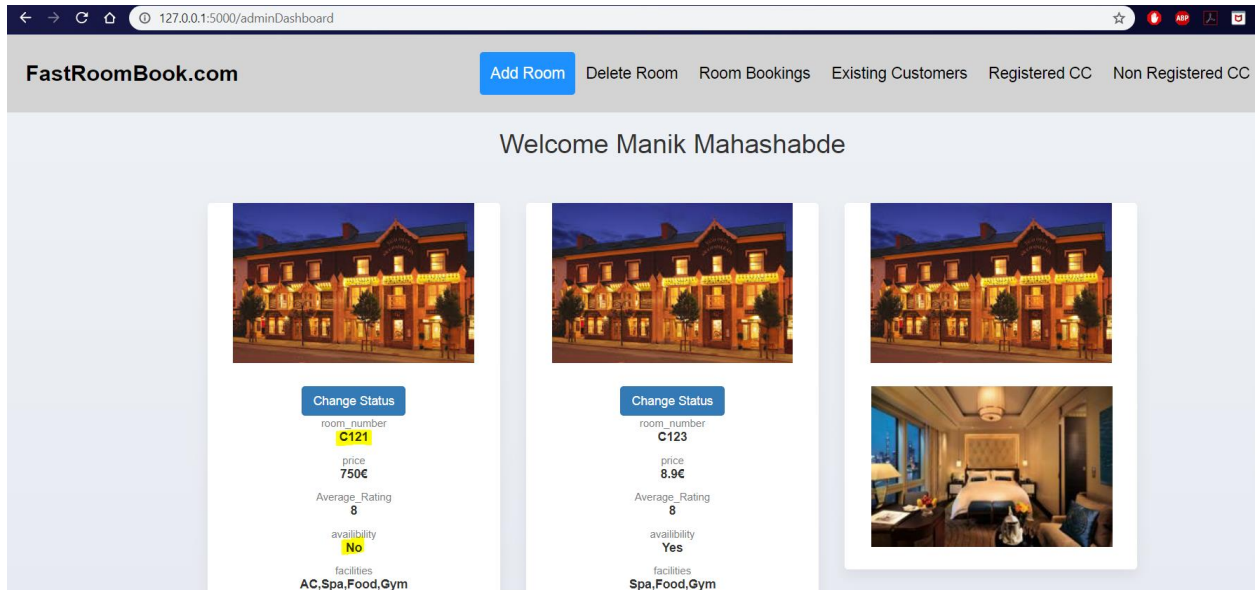


Fig 26: Room status change screenshot

Now the room C121 is not available on the user dashboard as seen below as it is not available.

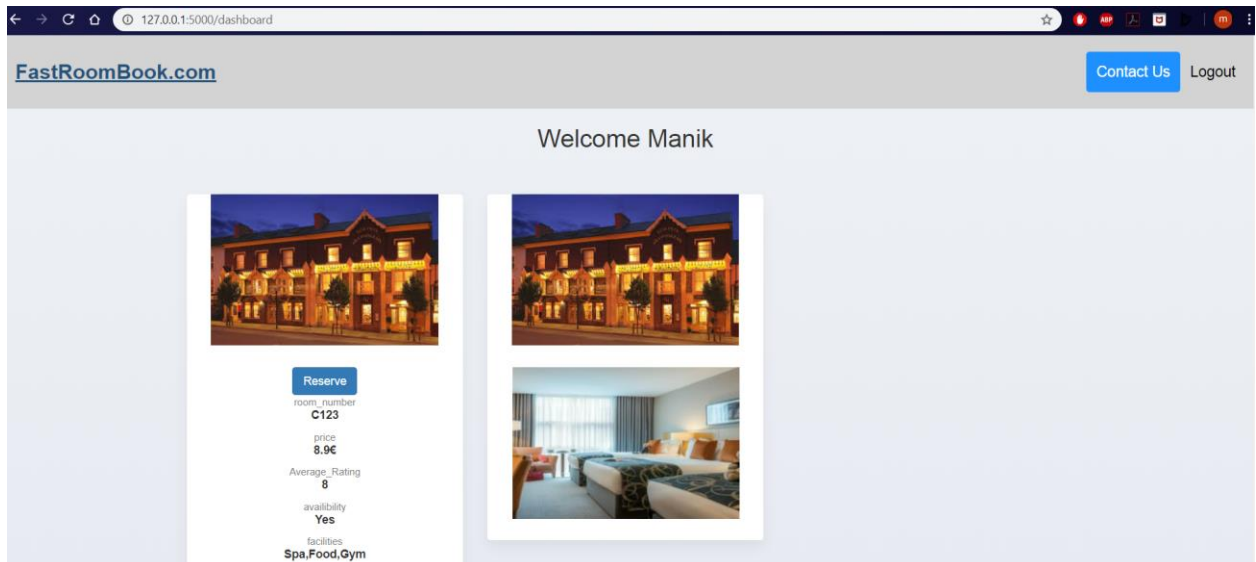


Fig 27: Room change reflection on the user page

g) Room deletion testing

Let us test room deletion by admin. As it can be seen that currently, 2 rooms exist on the admin page

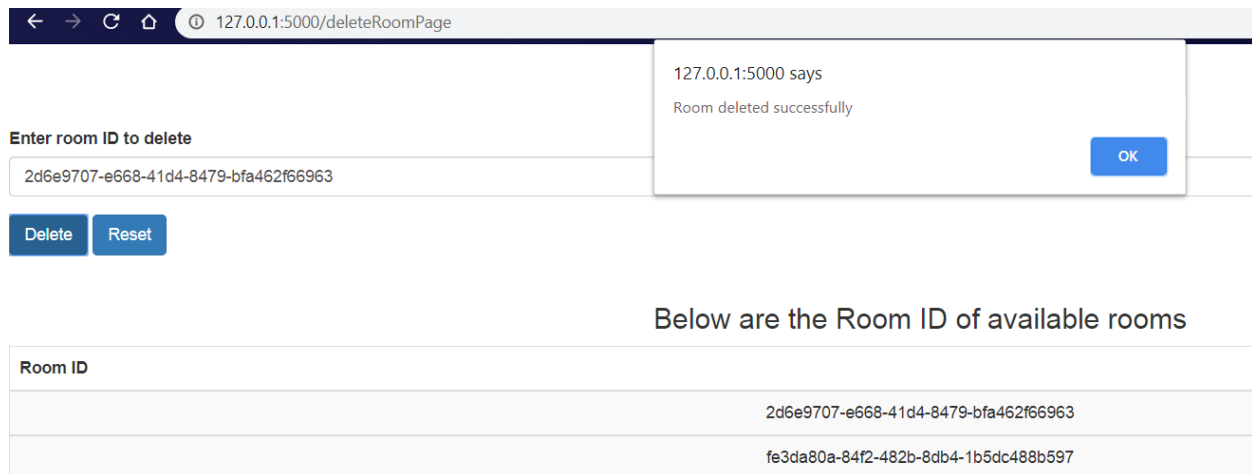


Fig 28: Room deletion testing

After the successful deletion, only one room exist



Fig 28: Room deletion result

Room with ID **2d6e9707-e668-41d4-8479-bfa462f66963** has been deleted successfully.

7: Security vulnerability and measures:

For any application, security is the most important thing. I have implemented the list of following securities to reduce vulnerabilities

a) Hashing of user password:

Whenever a user performs registration or changes his password. His password is stored in the database in the hashed form so that it is safe even if it is accessed by the hacker. I have used `werkzeug.security` package for implementing password hashing (*Utilities — Werkzeug Documentation (1.0.x), 2020*)

customer_id	name	username	password	email	contact_no	session_id	OTP
0247f8f1-9072-4b6e-8b63-2217de9...	Manik Mahas	manik123	obkdf2:sha256:150000\$cBwxtizG\$ae...	mahamanik@gmail.com	32324234	NULL	NULL

Fig 29: Hashed password storage

b) OTP based authentication in case of password reset.

It is explained in detail in testing case c. One important thing to note is that OTP is set to null as soon as the user creates a new password.

Hello User

Thanks for reaching out to us. Below is the 6 digit OTP for resetting your password for customer ID

31f0a4e4-0f35-4d76-b16f-9ffbcd1acd99
556d4f

Regards,
Team Admin

Fig 30: OTP email

c) Session management.

I have implemented session management in my application. As soon as a user or admin logs into the application the session is stored as shown below (endpoint **/dashboard**)

customer_id	name	username	password	email	contact_no	session_id	OTP
0247f8f1-9072-4b6e-8b63-2217de9...	Manik Mahas	manik123	obkdf2:sha256:150000\$cBwxtizG\$ae...	mahamanik@gmail.com	32324234	NULL	NULL
27ed3cc3-5942-4874-8172-84964b...	Ankit Das	coolchap	qwerty	mahamanik@gmail.com	3545454	NULL	NULL
31f0a4e4-0f35-4d76-b16f-9ffbcd1ac...	Manik	mahas500	obkdf2:sha256:150000\$02LvMAIVsf6f...	mahamanik@gmail.com	98782242	446bb560-4d7e-4c4d-91b7-f71ced6...	NULL

Fig 31: user session created at login

At this point dashboard is available. As soon as the user clicks logout button session_id is set to null

customer_id	name	username	password	email	contact_no	session_id	OTP
0247f8f1-9072-4b6e-8b63-2217de9...	Manik Mahas	manik123	obkdf2:sha256:150000\$cBwxtizG\$ae...	mahamanik@gmail.com	32324234	NULL	NULL
27ed3cc3-5942-4874-8172-84964b...	Ankit Das	coolchap	qwerty	mahamanik@gmail.com	3545454	NULL	NULL
31f0a4e4-0f35-4d76-b16f-9ffbcd1ac...	Manik	mahas500	obkdf2:sha256:150000\$02LvMAIVsf6f...	mahamanik@gmail.com	98782242	NULL	NULL

Fig 32: user session null at logout

And user tries to access **/dashboard** directly he gets a 500 error as he is not logged in.

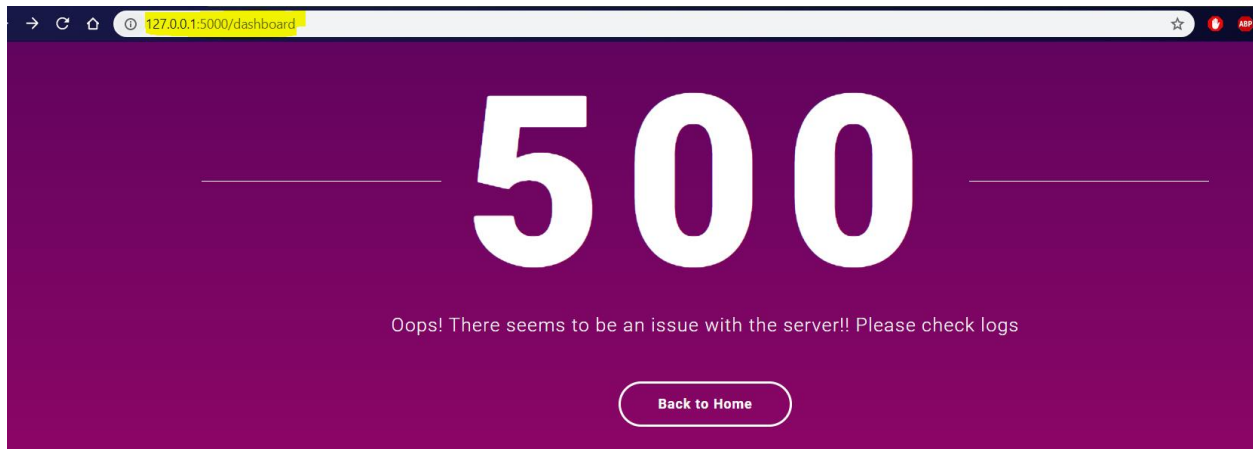


Fig 33: 500 HTTP status due to no session

d) Validation at the frontend and backend.

I have added validations at the frontend and backend such as password should be minimum 8 digits, no numbers in the name, no alphabets in contact number, correct email id format. If the user tries to change the password then the new password should not be as same as the old password. All of these details can be seen on my GitHub commits.

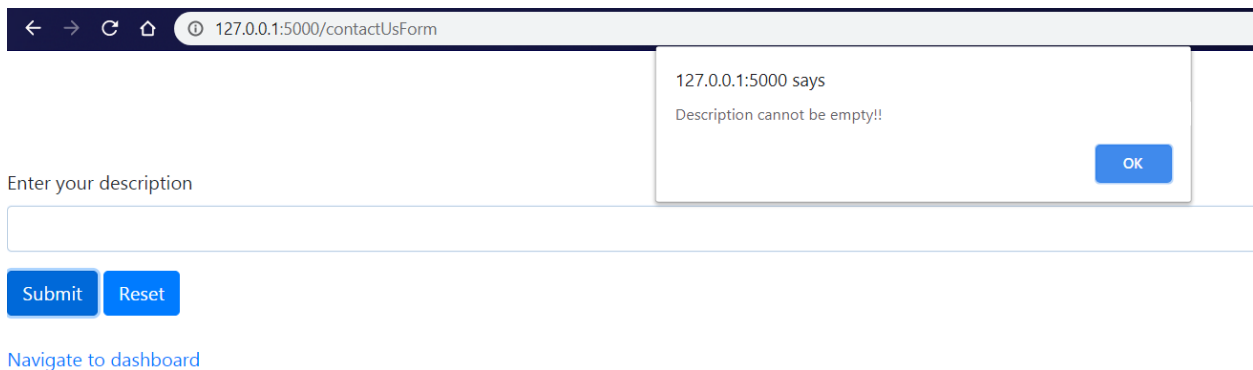


Fig 34: Description cannot be empty

The list of validation present on the website can be seen below. I have made different codes for various kind of exceptions. I am giving window alert on the basis of these codes on the frontend.

```

class CustomUtils:
    SUCCESSFUL = 1
    SOMETHING_WENT_WRONG = -1
    WRONG_CREDENTIALS = -2
    CUSTOMER_DOES_NOT_EXIST = -3
    ROOM_NOT_AVAILABLE = -4
    ROOM_WITH_GIVEN_NUMBER_ALREADY_EXIST = -5
    CUSTOMER_NOT_LOGGED_IN = -6
    EMAIL_SENDING_FAILED = -7
    CUSTOMER_WITH_EMAIL_NOT_EXIST = -8
    OTP_NOT_CORRECT = -9
    NO_BOOKINGS_EXIST = -10
    NO_CUSTOMERS_EXIST = -11
    ROOM_WITH_GIVEN_ID_DOES_NOT_EXIST = -12
    NOT_AUTHORIZED = -13
    INVALID_EMAIL = -14
    INVALID_CONTACT_NUMBER = -15
    INVALID_NAME = -16
    NEW_PASSWORD_SAME_AS_OLD_PASSWORD = -17
    DESCRIPTION_EMPTY = -18
    PASSWORD_TOO_SHORT = -19

```

Fig 35: Custom exception codes

Fig 36: Password change validation

e) Deployment to Heroku cloud

After completing the coding at local machine. I moved on to deploy my application on cloud. The platform I used is Heroku (*Getting Started on Heroku with Python | Heroku Dev Center, 2020*). The name of my app on Heroku platform is “**manik-webapp1**”. The reason I chose Heroku cloud platform is that it offers good integration with python application. I had to create a **Procfile** with below command:

```
web: gunicorn app:app
```

where app is my app name and gunicorn is web HTTP server for python web applications.

And another file is **requirement.txt** file in which there are all the dependencies needed for my application on cloud. Details about both these files can be seen on my github.

Cloud endpoint is <https://manik-webapp1.herokuapp.com/>

```

k, Flask-Cors, Flask-Dropzone, flask-heroku, Flask-Login, blinker, Flask-Mail, PyMySQL, Flask-MySQL, aniso8601, pytz, Flask-RESTful, SQLAlchemy, Flask-SQLAlchemy, Flask-Uploads, pycparser, cffi, bcrypt, cryptography, WTForms, Flask-WTF, passlib, Flask-User, gunicorn, pyOpenSSL, json5, attrs, pypersistent, jsonschema
remote: Successfully installed Flask-1.1.1 Flask-Bootstrap-3.3.7.1 Flask-Bootstrap4-4.0.2 Flask-Cors-3.0.8 Flask-Dropzone-1.5.4 Flask-Login-0.5.0 Flask-Mail-0.9.1 Flask-MySQL-1.5.1 Flask-RESTful-0.3.8 Flask-SQLAlchemy-2.4.1 Flask-Uploads-0.2.1 Flask-User-1.0.2.2 Flask-WTF-0.14.3 Jinja2-2.10.1 MarkupSafe-1.1.1 PyMySQL-0.9.3 SQLAlchemy-1.3.15 Six-1.14.0 WTForms-2.2.1 Werkzeug-1.0.0 aniso8601-3.0.0 attrs-19.3.0 bcrypt-3.1.7 blinker-1.4 cffi-1.14.0 click-7.1.1 cryptography-2.8 dominate-2.5.1 flask-heroku-0.1.9 gunicorn-20.0.4 itsdangerous-1.1.0 json5-0.8.4 jsonschema-3.0.1 passlib-1.7.2 pyOpenSSL-19.0.0 pycparser-2.20 pypersistent-0.15.7 pytz-2019.3 visitor-0.1.3
remote:
remote: -----> Discovering process types
remote: Procfile declares types => web
remote:
remote: -----> Compressing...
remote: Done: 57.5M
remote: -----> Launching...
remote: Released v3
remote: https://manik-webapp1.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/manik-webapp1.git
 * [new branch]      master -> master

C:\Users\Manik325\Heroku\Web-Development-CA>heroku ps:scale web=1
Scaling dynos... done, now running web at 1:Free

C:\Users\Manik325\Heroku\Web-Development-CA>heroku open

C:\Users\Manik325\Heroku\Web-Development-CA>heroku logs --tail
2020-03-13T20:25:33.568456+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.6/site-packages/gunicorn/arbiter.py", line 393, in stop
2020-03-13T20:25:33.568702+00:00 app[web.1]: time.sleep(0.1)
2020-03-13T20:25:33.568707+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.6/site-packages/gunicorn/arbiter.py", line 242, in handle_chld
2020-03-13T20:25:33.568908+00:00 app[web.1]: self.reap_workers()
2020-03-13T20:25:33.568914+00:00 app[web.1]: File "/app/.heroku/python/lib/python3.6/site-packages/gunicorn/arbiter.py", line 525, in reap_workers
2020-03-13T20:25:33.569417+00:00 app[web.1]: raise HaltServer(reason, self.WORKER_BOOT_ERROR)
2020-03-13T20:25:33.569475+00:00 app[web.1]: gunicorn.errors.HaltServer: <HaltServer 'Worker failed to boot.' 3>
2020-03-13T20:25:36.000000+00:00 app[api]: Build succeeded
2020-03-13T20:25:37.746214+00:00 heroku[web.1]: Starting process with command `gunicorn app:app`
2020-03-13T20:25:40.224606+00:00 heroku[web.1]: State changed from starting to up

```

Fig 37: Heroku deployment


```
Building wheel for SQLAlchemy (PEP 517): started
Building wheel for SQLAlchemy (PEP 517): finished with status 'done'
Created wheel for SQLAlchemy: filename=SQLAlchemy-1.3.15-cp36-cp36m-linux_x86_64.whl size=1230214
sha256=166be062d5c0e4b1eb2154c7bcdbe51974090347d55c9f4094b8dd08b404e17
Stored in directory: /tmp/pip-ephem-wheel-cache-v0ewtwxb/wheels/4a/1b/3a/c73044d7be48baeb47cbee343334f7803726ca1e9ba7b29095
Building wheel for pyparsing (setup.py): started
Building wheel for pyparsing (setup.py): finished with status 'done'
Created wheel for pyparsing: filename=pyparsing-0.15.7-cp36-cp36m-linux_x86_64.whl size=115000 sha256=232bddaef85faae4d3c1d86827f457c9e29474cf184f642c5745b6d0f753c6b2
Stored in directory: /tmp/pip-ephem-wheel-cache-v0ewtwxb/wheels/43/9a/e4/7f687f2bb934e9a26a6e1158778ed7c5436b9ea15db48c888a
Successfully built Flask-Bootstrap flask-heroku Flask-Mail Flask-Uploads Flask-User visitor blinker SQLAlchemy pyparsing
Installing collected packages: MarkupSafe, Jinja2, click, Werkzeug, itsdangerous, Flask, dominate, visitor, Flask-Bootstrap, Flask-Bootstrap4, Six, Flask-Cors, Flask-Dropzone, Flask-heroku, Flask-Login, blinker, Flask-Mail, PyMySQL, Flask-MySQL, aniso8601, pytz, Flask-RESTful, SQLAlchemy, Flask-SQLAlchemy, Flask-Uploads, pycparser, cffi, bcrypt, cryptography, WTForms, Flask-WTF, passlib, Flask-User, gunicorn, pyOpenSSL, json5, attrs, pyparsing, jschema
Successfully installed Flask-1.1.1 Flask-Bootstrap-3.3.7.1 Flask-Bootstrap4-4.0.2 Flask-Cors-3.0.8 Flask-Dropzone-1.5.4 Flask-Login-0.5.0 Flask-Mail-0.9.1 Flask-MySQL-1.5.1 Flask-RESTful-0.3.8 Flask-SQLAlchemy-2.4.1 Flask-Uploads-0.2.1 Flask-User-1.0.2.2 Flask-WTF-0.14.3 Jinja2-2.10.1 MarkupSafe-1.1.1 PyMySQL-0.9.3 SQLAlchemy-1.3.15 Six-1.14.0 WTForms-2.2.1 Werkzeug-1.0.0 aniso8601-8.0.0 attrs-19.3.0 bcrypt-3.1.7 blinker-1.4 cffi-1.14.0 click-7.1.1 cryptography-2.8 dominate-2.5.1 flask-heroku-0.1.9 gunicorn-20.0.4 itsdangerous-1.1.0 json5-0.8.4 jschema-3.0.1 passlib-1.7.2 pyOpenSSL-19.0.0 pycparser-2.20 pyparsing-0.15.7 pytz-2019.3 visitor-0.1.3
----> Discovering process types
Procfile declares types -> web
----> Compressing...
Done: 57.5M
----> Launching...
Released v3
https://manik-webapp1.herokuapp.com/ deployed to Heroku

Build finished
```

Fig 38: Heroku deployment build status on webpage

8: Github:

The complete code with 129 commits is available on below URL

<https://github.com/mahas500/Web-Development-CA>

Please access it for in depth details about my implementation.

9: Learning Outcomes:

The learning I received in this outcome is really important for me for my future. Giving individual assignment turned out pretty well for me as I learned about pressure handling, self-confidence, independent research. I learned about flask, jinja2, python in the module and combined them with my existing knowledge and let of independent research to achieve my target.

REFERENCES & BIBLIOGRAPHY

Docs.python.org. 2020. *3.8.2 Documentation*. [online] Available at: <<https://docs.python.org/3/>> [Accessed 13 March 2020].

W3schools.com. 2020. *HTML Tutorial*. [online] Available at: <<https://www.w3schools.com/html/>> [Accessed 13 March 2020].

Mark Otto, a., 2020. *Introduction*. [online] Getbootstrap.com. Available at: <<https://getbootstrap.com/docs/4.4/getting-started/introduction/>> [Accessed 13 March 2020].

Javascript.com. 2020. *Free Javascript Training, Resources And Examples For The Community*. [online] Available at: <<https://www.javascript.com/>> [Accessed 13 March 2020].

js.foundation, J., 2020. *Jquery API Documentation*. [online] Api.jquery.com. Available at: <<https://api.jquery.com/>> [Accessed 13 March 2020].

Fullstackpython.com. 2020. *Flask*. [online] Available at: <<https://www.fullstackpython.com/flask.html>> [Accessed 13 March 2020].

Dev.mysql.com. 2020. *Mysql :: Mysql Documentation*. [online] Available at: <<https://dev.mysql.com/doc/>> [Accessed 13 March 2020].

Werkzeug.palletsprojects.com. 2020. *Utilities — Werkzeug Documentation (1.0.X)*. [online] Available at: <<https://werkzeug.palletsprojects.com/en/1.0.x/utils/>> [Accessed 13 March 2020].

Devcenter.heroku.com. 2020. *Getting Started On Heroku With Python | Heroku Dev Center*. [online] Available at: <<https://devcenter.heroku.com/articles/getting-started-with-python>> [Accessed 13 March 2020].