San Francisco State University
Computer Science Department
SW Engineering CSC648/848 Fall 2021 Section 02
Group 4
Milestone 4
11/09/2021

Team lead, GitHub lead/master, Document lead: Georgina Shirazi (email: gshirazi@mail.sfsu.edu)

Gabriel Pena, Database lead
Eanguy Eng, Database

Swetha C, Backend lead
Zhiling Huang, Backend

Sanket Naik, Frontend lead
William Zhong, Frontend

GatorRoomer

Version	**Date submitted**	**Action**
1.0	11/09/2021	Submitted
2.0	11/16/2021	

1) Product Summary:

Executive Summary:
Our motivation for creating GatorRoomer is to make it easy and intuitive for students and professionals alike to find the perfect roommate or living space. This is an app for students and professionals, made by students.

Finding a room can be a stressful time--whether you are a freshman student going off to college, or a professional who has just been relocated to a different city, you are stepping into the unknown. We hope to mitigate that unknown with GatorRoomer as much as possible and make finding a roommate or place to stay quick and easy. To achieve our goal of an easy and intuitive experience, we shall implement an array of different filters and tags to make finding the perfect roommate easier. We shall implement a bookmark feature to make sure users can find listings they wish to check out. Our superior messaging system will give the user the option to ask a line of questions with the potential roommate or landlord. Users shall have notifications as a preference option . When uploading a room, users shall be able to upload pictures of the room. Users shall have separate sessions for each user. These are just a few services GatorRoomer will offer to make finding a roommate the easiest it has ever been.

Our team is composed of a diverse array of university students across many different college years, with different backgrounds and experience. We want to channel our unique skill sets to work together and produce a useful and intuitive app . We are hungry to compete with the big players in the roommate finding industry. As university students, we know what is important when it comes to finding a roommate. Thus, we want to make the process simple and easy--because we know students and professionals have more important things to worry about.

Some of GatorRoomer's distinguishable features of quality include:

1) Users shall be able to filter through room listings using price, location, tags, and if the search parameters do not produce any results, then a message is displayed to the user indicating nothing was found.

2) Users shall be able to use bookmark/ "like" room listings.

3) Users shall be able to have separate sessions for each user.

4) User shall have new post notifications

5) User shall be able to ask questions to landlord or roommate (webmail) -

6) User shall be able to upload a picture(s) of the room

What makes our product unique:
Our application provides a simplistic and calming design which is persistent throughout the website. In addition, our UI displays artwork.

Each tool in the UI is centralized where it makes sense to do so. The search API is consolidated, so users go to one location on the screen to search. The webpage shows something but is not overwhelming--a nice way to relax customers who may be stressed while using the product, as they are seeking housing.

Users can create an account on our website and use our service for free! There are no fees or expenses to be paid by using our product. Creating an account is easy and intuitive with our create-an-account form which provides form validation. However, unlike our competitors, users do not have to login to view rooms or roommates.

Switching between search-types: choosing to search by room or roommate can be toggled by a button.

Room-positing: logged-in users can post a new room posting with ease, with our clean and neat form, which provides form validation.

Bookmarking: logged-in users can revisit rooms and roommates of interest with ease by bookmarking them.

Notifications: logged-in users are provided with optional notifications based on their notification preferences, which can be modified on the alerts page.

Customizable preferences: logged-in users can choose roommate preferences with the roommate preferences page.

In-site messaging: our website provides an in-site messaging system, where logged-in users can message about rooms of interest to the owner/renter.

URL: http://ec2-18-191-86-17.us-east-2.compute.amazonaws.com

2) Usability Test Plan:

Scenario being tested:

We are testing the uploading and posting of rooms.

Test objective:

We are testing this function as it is a central part of this application for users to find rooms and roommates. Users can upload rooms on our website and view posted rooms.

Only logged-in users can upload rooms. This is to prevent spam, and prevents scams as it holds logged in users accountable. Unregistered users can still view the rooms.

Users can input the following data into the upload room form:
Location, zip code, type, description, price, size, number of bedrooms, number of bathrooms, has pets, smoking preference, parking availability, college preference, disability, negotiation and private restroom ability and pictures of the room.

The newly added rooms are accessible via the search API.
Users can search for rooms by using these filters:
location, zip code, room type (i.e apartment, mobile house), number of bedrooms, number of bathrooms, smoking preference, pet preference, parking availability, college oriented or not, disability friendly, rent negotiation and restroom privacy.

Test background and setup:

System setup:
The GatorRoomer website homepage.

Starting point:
The user clicks the website's URL. This is on the homepage, before the user logs in. The user needs to login as not logged in users cannot create new rooms. After logging in, the user is redirected to their dashboard page where the add room option is seen.

Intended users:
Only logged in users can add rooms. So the users who are already registered can add rooms.

URL to be tested: http://ec2-18-191-86-17.us-east-2.compute.amazonaws.com

Usability Task description:
The user is on the website's homepage. The user must have an account. The user logs into their account.

The user proceeds to add a room. In the “add a room” form, the user supplies details and a picture for their room.

Then, they use the search API to search for the room via room, zip code, location, and/or applicable filters (has pets, et cetera).

Effectiveness:
The logged in user should be able to access the “add a room” form. The user should add details for the room. The user should be able to see the newly added room with all the details supplied.

Efficiency:
The user should be able to find the “add a room” button easily and quickly. The user should have an easy time filling out the form. The user should be able to find the new room post using the search API without too much effort.

Lickert subjective test:

I found the GUI easy to use	Strongly disagree Disagree Neutral Agree Strongly agree
I found the “add room” button easy to find	Strongly disagree Disagree Neutral Agree Strongly agree
the form was intuitive and easy to use On a scale from 0 to 5, how easy was it to use the form?	0 1 2 3 4 5
I found the search tool easy to use when finding my uploaded room	Strongly disagree Disagree Neutral Agree Strongly agree

3) QA Test Plan:

- Test objectives: - what is being tested
We are testing the uploading and posting of rooms.

- HW and SW setup (including URL):
Hardware setup: virtual hardware -- using AWS EC2, RDS, and S3 aer virtual hardware.

URL: http://ec2-18-191-86-17.us-east-2.compute.amazonaws.com

Software setup:
The codebase for our website. Users should be on the homepage, should login with a valid user account.

Username = "test123",
Password :"test"

- Feature to be tested
Adding rooms to our website.

- QA Test plan: - table format: 3 test cases and results of testing them on your system: appr. 1 page. Use tabular formats as in the class, You must provide QA test plan in the format of the easy to read tabular form allowing easy reading and analysis by management e.g. like presented in the class slides on SW QA.

For Firefox:

number	description	test input	expected output	PASS/FAIL
1	Add the room with these fields: location: % ZipCode: % type: % description: % price: %	324 Grand View Avenue 94114 Apartment QA Test #1 1 800 1	Successfully added (missing an input will show there is an error and will not submit the form) Go to homepage. Search location/zipcode	PASS

	size (sq. ft.): % bedrooms: % bathrooms: % smoking: % pets: % parking: % college: % disability: % negotiation: % restroom: %	1 1 1 no smoking Pet Friendly Parking Available Not College Oriented Not Disability Friendly Rent Negotiable Private Restroom Hit submit	and room should appear with search API. Search for “Grand View” to find this post on the website.	
2	Add the room with these fields: location: % ZipCode: % type: % description: % price: % size (sq. ft.): % bedrooms: % bathrooms: % smoking: %	 “13300 Test St.” “92400” Choose House “QA Test #2” 15300 1200 2 2 Smoking Friendly	 Successfully added. Go to home screen and type in search bar “92400” To check if it appears.	PASS

	pets: % parking: % college: % disability: % negotiation: % restroom: %	No Pets Allowed No Parking Available College Oriented Disability Friendly Rent Negotiable Private Room Hit submit		
3	Add the room with these fields: location: % ZipCode: % type: % description: % price: % size (sq. ft.): % bedrooms: % bathrooms: % smoking: % pets: % parking: % college: % disability: % negotiation: % restroom: %	“419 Ocean Street” 94112 “Apartment” “A 2 bedroom apartment with great views and all amenities.” 2500 800 2 2 No Smoking Pet Friendly Parking Available Not College Oriented Disability Friendly Rent Not Negotiable Private Restroom Hit Submit	Successfully added Go to the home page and type “94112” in the search bar to search for the listing	PASS

For Chrome:

number	description	test input	expected output	PASS/FAIL
1	Add the room with these fields: location: % ZipCode: % type: % description: price: size (sq. ft.): bedrooms: bathrooms: smoking: pets: parking: college: disability: negotiation: restroom:	 “13300 Test St.” “92400” Choose House “QA Test #2” 15300 1200 2 2 Smoking Friendly No Pets Allowed No Parking Available College Oriented Disability Friendly Rent Negotiable Private Room Hit submit	Successfully added (missing an input will show there is an error and will not submit the form) Go to home screen and type in search bar “92400” To check if it appears.	PASS
2	location: % ZipCode: % type: % description:	324 Grand View Avenue 94114 Apartment QA Test #1	Successfully added Go to homepage. Search location/zipcode and room should appear with	PASS

	price: size (sq. ft.): bedrooms: bathrooms: smoking: pets: parking: college: disability: negotiation: restroom:	1 800 1 1 1 no smoking Pet Friendly Parking Available Not College Oriented Not Disability Friendly Rent Negotiable Private Restroom Hit submit	search API. Search for “Grand View” to find this post on the website.	
3	Add the room with these fields: location: % ZipCode: % type: % description: % price: % size (sq. ft.): % bedrooms: %	“419 Ocean Street” 94112 “Apartment” “A 2 bedroom apartment with great views and all amenities.” 2500 800 2	Successfully added Go to the home page and type “94112” in the search bar to search for the listing	PASS

	bathrooms: % smoking: % pets: % parking: % college: % disability: % negotiation: % restroom: %	2 No Smoking Pet Friendly Parking Available Not College Oriented Disability Friendly Rent Not Negotiable Private Restroom Hit Submit		

4) Code Review:

Coding style:

We use PEP 8 for python coding style

1. Use 4 space indentation
2. Use docstrings
   Def foo()
   “”” define a function”””
3. A lines doesn’t exceed 79 characters
4. Default UTF-8 or ASCII encodings
5. Using space after commas
6. Name
   Snake_case naming style for function
   Pascal_case naming style for class
7. Name classes and function consistently

Code chosen:

Add Room
Database.py, method name: use add_data() // use for add data to db
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/db/database.py
S3.py, method name: upload_file_to_s3() // upload files to aws s3
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/db/s3.py
Roomhandler.py: method name: add_new_room() , check_no_dup(), and add_media()
//Adding the new room to db , add file to s3 and check duplicated
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/model/roomhandler.py
Roomscontroller.py: method name: add_room() // combine aboves method, Add Room API
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/controller/roomscontroller.py
Main.py : method name : add_room() // Router
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/main.py

Login:
To login i.e check if the user exists:
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/master/application/backend/controller/userscontroller.py
Method name: login()

Creating a session for the user who logged in:
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/master/application/backend/controller/sessioncontroller.py
Method name: create_session()

Search API to get rooms:
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/master/application/backend/controller/roomscontroller.py
Method name: get_all_rooms()

Peer review for uploading room code:

Reviewee (please send code to reviewer)	Reviewer
Zhiling	Georgina
Swetha	Gabriel
Sanket	Ryan

Gmail ALIN HUANG <alinhuang4991@gmail.com>

Re: Code Review

Georgina Catherine Shirazi <gshirazi@mail.sfsu.edu> 2021年11月5日 下午3:59
收件人：ALIN HUANG <alinhuang4991@gmail.com>

Hello Zhiling,

I used this formatter to help me apply PEP 8.
http://pep8online.com/checkresult

Here are the following suggestions:

Database.py:

Method names:
add_data()

Method add_new_room():

current code is with the change I suggested in the previous email
Current code:

```
def add_data(self, sql):
    """Add data to DB using connection object from connection pool."""
    connection_object = self.connection_pool.get_connection()
    try:
        if connection_object.is_connected():
            cursor = connection_object.cursor()
            cursor.execute(sql)
            connection_object.commit()
            record = cursor.rowcount
            LOGGER.info(' Data to DB: ' + str(record)+ " row")
            return "Successfully added "
    except Error as error:
        LOGGER.error(error)
        return "Error in adding"
```

Proposed changed code:

Proposed changed code:

```
def add_data(self, sql):
    """Add data to DB using connection object from connection pool."""
    connection_object = self.connection_pool.get_connection()
    try:
```



```
        if connection_object.is_connected():
            cursor = connection_object.cursor()
            cursor.execute(sql)
            connection_object.commit()
            record = cursor.rowcount
            LOGGER.info(' Data to DB: ' + str(record) + " row")
            return "Successfully added "
    except Error as error:
        LOGGER.error(error)
        return "Error in adding"
```

Changes:

Line	description
48	Added spaces between + (so there is a space between +)

S3.py:

Method names:
upload_file_to_s3()

Method add_new_room():

Current code:

```
def upload_file_to_s3(self,file,acl="public-read"):
    """This method is used to upload the files ."""
    filename =secure_filename(file.filename)
    unique_str_id =uuid4().__str__()[:8]
    file.filename = unique_str_id +filename # setup file name is unique
    name = file.filename
```

```
        file.filename = unique_str_id +filename # setup file name is unique
        name = file.filename
        path_to_file_name=f"public/title_images/{name}"
        LOGGER.info(f"Uploded file to s3: {path_to_file_name}")
        try:
            self.s3.upload_fileobj(
                    file, # file in bytes
                    AWS_BUCKET_NAME,
                    path_to_file_name, # Bucket path/../filename
                    ExtraArgs={   # ExtraArgs pass to client
                        "ACL":acl,
                        "ContentType":file.content_type})
        except Exception as error:
            LOGGER.error(error)
```



```
            return error
        return path_to_file_name
```

Proposed changed code:

```
def upload_file_to_s3(self, file, acl="public-read"):
    """This method is used to upload the files ."""
    filename = secure_filename(file.filename)
    unique_str_id = uuid4().__str__()[:8]
    file.filename = unique_str_id + filename  # setup file name is unique
    name = file.filename
    path_to_file_name = f"public/title_images/{name}"
    LOGGER.info(f"Uploded file to s3: {path_to_file_name}")
    try:
        self.s3.upload_fileobj(
                file,  # file in bytes
                AWS_BUCKET_NAME,
                path_to_file_name,  # Bucket path/../filename
                ExtraArgs={   # ExtraArgs pass to client
                    "ACL": acl,
                    "ContentType": file.content_type})
    except Exception as error:
        LOGGER.error(error)
```

```
        LOGGER.error(error)
        return error
    return path_to_file_name
```

Changes:

Line	description
19	added a space between “self,” “file,” and ‘acl=”public-read”’
21	the assignment operator has a space around it (2 total)
22	the assignment operator has a space around it (2 total)
23	the + operator has a space around it (2 total) and the inline comment has two spaces in front of it now.
25	the assignment operator has a space around it (2 total)
29	the inline comment has two spaces in front of it now.
31	the inline comment has two spaces in front of it now.
33	the colon now has a space after it
34	the colon now has a space after it

Roomhandler.py:

Method names:
add_new_room() , check_no_dup(), and add_media()

Method add_new_room():

Current code:

```
def add_new_room(self,body):
    """Construct the queries to execute add room to the db."""
```

```
    today= datetime.now().strftime("%Y-%m-%d")
    columns = ["ListTime"]
    day=f"'{today}'"
    values = [day]
    for element in body:
        columns.append(element)
        if isinstance(body.get(element), str):
            values.append("'"+body.get(element)+"'")
        else:
            values.append(body.get(element))
    col_str = ",".join(map(str, columns))
    val_str = ",".join(map(str, values))
    sql = f'INSERT INTO Test1.RoomListing ({col_str}) VALUES ({val_str});'
    self.dbinstall.add_data(sql)
    # use check_no_dup to find the room id
    room_id = self.check_no_dup(body)
    return room_id
```

Proposed changed code:

```
def add_new_room(self, body):
    """Construct the queries to execute add room to the db."""
    today = datetime.now().strftime("%Y-%m-%d")
    columns = ["ListTime"]
    day = f"'{today}'"
    values = [day]
    for element in body:
        columns.append(element)
        if isinstance(body.get(element), str):
            values.append("'"+body.get(element)+"'")
```



```
        else:
            values.append(body.get(element))
    col_str = ",".join(map(str, columns))
    val_str = ",".join(map(str, values))
    sql = f'INSERT INTO Test1.RoomListing ({col_str}) VALUES ({val_str});'
    self.dbinstall.add_data(sql)
    # use check_no_dup to find the room id
    room_id = self.check_no_dup(body)
```

```
    return room_id
```

Changes:

Line	description
44	added a space between “self” and “body”
46	the assignment operator has a space around it (2 total)
48	the assignment operator has a space around it (2 total)

Method
check_no_dup():

Current code:

```
def check_no_dup(self,body):
    """This Method check Duplicate Room and get a room id."""
    location = body["Location"]
    room_type = body["Type"]
    zip_code = body["ZipCode"]
    des=body["Description"]
    user_id = body["Lister"]
    req= f"Location = '{location}' and Type='{room_type}' and ZipCode={zip_code} and Description = '{des}' and
Lister = {user_id};"
    sql= f" SELECT R.Room_ID FROM Test1.RoomListing R WHERE (Available=0) and {req}"
    room_id = self.dbinstall.get_data(sql)
    if room_id == 0: # no dup
        return room_id
    elif len(room_id)>1: ## have dup
        return -1
    else:       ## only one
        tup=room_id[0]
        return tup[0]
```

Proposed changed code:

```
def check_no_dup(self, body):
    """This Method check Duplicate Room and get a room id."""
    location = body["Location"]
    room_type = body["Type"]
    zip_code = body["ZipCode"]
    des = body["Description"]
    user_id = body["Lister"]
    req = f"Location = '{location}' and Type='{room_type}'"
    +"and ZipCode={zip_code} and Description = '{des}' and Lister = {user_id};"
    sql = f" SELECT R.Room_ID FROM Test1.RoomListing R "
    +"WHERE (Available=0) and {req}"
    room_id = self.dbinstall.get_data(sql)
    if room_id == 0:  # no dup
        return room_id
    elif len(room_id) > 1:  # have dup
        return -1
    else:       # only one
        tup = room_id[0]
        return tup[0]
```

Changes:

Line	description
25	added a space between “self” and “body”
30	the assignment operator now has a space around it (2 total)
32	The assignment operator now has a space around it (2 total). The line has been separated into two lines. The current line exceeds 79 characters.
33	The assignment operator now has a space around it (2 total). The line has been separated into two lines. The current line exceeds 79 characters.
35	The spaces between the code and inline comment are now 2 (needs to have at least 2 spaces between code and inline comment).
37	The spaces between the code and inline comment are now 2 (needs to have at least 2 spaces between code and inline comment). Inline comments must start with a single “#” character. This comment has one #. The > operator now has a space around it (2 total).
39	Inline comments must start with a single “#” character. This comment has one #.
40	The assignment operator now has a space around it (2 total).

Method
add_media():

Current code:

```
def add_media(self,room_id,file):
    """Construct the queries to execute add media the db and s3."""

    output = self.s3install.upload_file_to_s3(file)
    url = AWS_BUCKET_HEAD+output
    sql = f"INSERT INTO Test1.RoomMedia (RoomID,RoomPic) VALUES ({room_id},'{url}');"
    message=self.dbinstall.add_data(sql)
    return message
```

Proposed changed code:

```
def add_media(self, room_id, file):
  """Construct the queries to execute add media the db and s3."""

  output = self.s3install.upload_file_to_s3(file)
  url = AWS_BUCKET_HEAD+output
  sql = f"INSERT INTO Test1.RoomMedia "
  +"(RoomID,RoomPic) VALUES ({room_id},'{url}');"
  message = self.dbinstall.add_data(sql)
  return message
```

Changes:

Line	description
66	added a space between “self” and “room_id,” and “file”
70	The line has been separated into two lines. The current line exceeds 79 characters.
71	The assignment operator now has a space around it (2 total).

Roomscontroller.py:

Method name:

add_room()

Method add_room() :

Current code:

```
def add_room():
    """add room for the given room attributes."""


    checker=sessioncontroller.check_loggedin()
    roomhandle = RoomHandler()
    if checker:
        req_body =request.form.get('json')
        body = json.loads(req_body)
        rjson = ast.literal_eval(json.dumps(body))
        rjson["Lister"]=session["userid"]
        id = session["userid"]
        checkdup = roomhandle.check_no_dup(rjson)
        if checkdup == 0:
            LOGGER.info(' JSON from frontend {%s}',rjson)
            room_id=roomhandle.add_new_room(rjson)
            if room_id > 0:
                LOGGER.info('Add Json Info to RoomID{%s}',room_id)
                filemessage="None"
                if 'files' in request.files:
                    req_files = request.files.getlist("files")
                    if len(req_files) > 0:
                        for file in req_files:
                            name = file.filename
                            LOGGER.info('Upload File {%s }to RoomId : {%s}',name,room_id)
                            filemessage = roomhandle.add_media(room_id,file)
                    return {"Roominfo": "success added room info","Roomfile": filemessage}

                LOGGER.info('Successfully Add Json info to Room_ID {%s}',room_id)
                return "successfuly add room"
            elif id == 0 :
                LOGGER.error('Faild Add Json info to Room_ID {%s}',room_id)
                return "faild to adding room"
            elif id == -1:
                LOGGER.error('Faild Add Room info to Room_ID{%s}',room_id)
                return "You have created many same Rooms"
```

```
        LOGGER.error("Have same id and info in RoomListing")
        return "You have created a same Room"
    return "please login first"
```


Proposed code:

```
def add_room():
    """add room for the given room attributes."""

    checker = sessioncontroller.check_loggedin()
    roomhandle = RoomHandler()
    if checker:
        req_body = request.form.get('json')
        body = json.loads(req_body)
        rjson = ast.literal_eval(json.dumps(body))
        rjson["Lister"] = session["userid"]
        id = session["userid"]
        checkdup = roomhandle.check_no_dup(rjson)
        if checkdup == 0:
            LOGGER.info(' JSON from frontend {%s}', rjson)
            room_id = roomhandle.add_new_room(rjson)
            if room_id > 0:
                LOGGER.info('Add Json Info to RoomID{%s}', room_id)
                filemessage = "None"
                if 'files' in request.files:
                    req_files = request.files.getlist("files")
                    if len(req_files) > 0:
                        for file in req_files:
                            name = file.filename
                            LOGGER.info(
                                'Upload File {%s }to RoomId : {%s}',
                                name, room_id)
                            filemessage = roomhandle.add_media(room_id, file)
                    return {"Roominfo": "success added room info",
                            "Roomfile": filemessage}
```

```
            LOGGER.info('Successfully Add Json info to Room_ID {%s}',
                        room_id)
            return "successfuly add room"
        elif id == 0:
            LOGGER.error('Faild Add Json info to Room_ID {%s}', room_id)
            return "faild to adding room"
        elif id == -1:
            LOGGER.error('Faild Add Room info to Room_ID{%s}', room_id)
            return "You have created many same Rooms"
    LOGGER.error("Have same id and info in RoomListing")
    return "You have created a same Room"
```



```
    return "please login first"
```

Changes:

Line	description
119	Removed empty line.
120	The assignment operator now has a space around it (2 total).
123	The assignment operator now has a space around it (2 total).
126	The assignment operator now has a space around it (2 total).
130	Added a space after “,”
131	The assignment operator now has a space around it (2 total).
133	Added a space after “,”
134	The assignment operator now has a space around it (2 total).
140	Line exceeds 79 characters. Broke it into separate lines.
141	Added a space after “,”
142	Line exceeds 79 characters. Broke it into separate lines.
144	Added a space after “,” and split the line into two separate lines.
146	removed the space before the “:”
147	Added a space after “,”
150	Added a space after “,”

Main.py:
Method name:
add_room()

Everything is perfect here.

You've done a lot of work for this project. There are just a few minor code style changes to be made in the files.

The code in this email should be copy and paste-able. If the formatting is off, please view the document here. It might have better formatting.
https://docs.google.com/document/d/1okn_tYL4B-eFE6PMeo62xkYzsw4D9mSf1_fKhQx3ZUg/edit

Thanks!

Sincerely,
Georgina

From: ALIN HUANG <alinhuang4991@gmail.com>
Sent: Friday, November 5, 2021 12:13 PM
To: Georgina Catherine Shirazi <gshirazi@mail.sfsu.edu>
Subject: Re: Code Review

Thanks for let me know. I will fix all this kind of problems.

On Fri, Nov 5, 2021 at 11:08 AM Georgina Catherine Shirazi <gshirazi@mail.sfsu.edu> wrote:

> Hello Zhiling,
>
> I haven't finished reviewing all the files yet, but I have some suggested changes for database.py.
>
> for add_data():
>
> line 39:
> def add_data(self,sql):

From: ALIN HUANG <alinhuang4991@gmail.com>
Sent: Friday, November 5, 2021 12:13 PM
To: Georgina Catherine Shirazi <gshirazi@mail.sfsu.edu>
Subject: Re: Code Review

Thanks for let me know. I will fix all this kind of problems.

On Fri, Nov 5, 2021 at 11:08 AM Georgina Catherine Shirazi <gshirazi@mail.sfsu.edu> wrote:

> Hello Zhiling,
>
> I haven't finished reviewing all the files yet, but I have some suggested changes for database.py.
>
> for add_data():
>
> line 39:
> def add_data(self,sql):
> A space is needed between "self" and "sql."
> Proposed change:
> def add_data(self, sql):
>
> Sincerely,
> Georgina Shirazi

From: ALIN HUANG <alinhuang4991@gmail.com>
Sent: Tuesday, November 2, 2021 10:15 PM
To: Georgina Catherine Shirazi <gshirazi@mail.sfsu.edu>
Subject: Code Viewing

Hi, Georgina
Can you review my code and give me some feedback?
Here is my GitHub link:

Function:
Add Room

Database.py,
method name: use add_data() // use for add data to db
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/db/database.py

S3.py,
Method
name: upload_file_to_s3()
//
upload files to aws s3

https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/db/s3.py

Roomhandler.py:
Method
name: add_new_room() , check_no_dup(), and add_media()
//Adding
the new room to db , adding the file to s3 and checking duplicated
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/model/roomhandler.py

Roomscontroller.py:
Method
name: add_room()
//
combine aboves method, Add Room API
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/controller/roomscontroller.py

Main.py
:
Method
name : add_room() // Router
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/dev-backend/application/backend/main.py

Please, Let me know if you have any questions or improvements about my code. Your feedback is important for me to improve my code.

Thanks,
Zhiling

Code Review

Swetha C <c.saiswetha@gmail.com> 9:39 PM (3 minutes ago)

to gabrielpena147

Hey Gabriel,

Would you review the code & give me feedback so that I can improvise the code?
I have provided the links & the method names for each functionality which you need to review.
Login:
To login i.e check if the user exists:
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/master/application/backend/controller/userscontroller.py
Method name: login()

Creating a session for the user who logged in:
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/master/application/backend/controller/sessioncontroller.py
Method name: create_session()

Search API to get rooms:
https://github.com/CSC-648-SFSU/CSC648-fall21-02-team04/blob/master/application/backend/controller/roomscontroller.py
Method name: get_all_rooms()

Regards,
Swetha

Reply Forward

Gabriel Pena <gabrielpena147@gmail.com>
to Swetha

def login()
Lines 348 - 354 have inconsistent indentation compared to the rest of the lines.
Good relevant names for everything as well as snake case naming used.
A bit light on the comments. I understand it's a login method and self-explanatory but perhaps explain what each step does.

```
330  def login():
331      """Login method."""
332
333      '''Perhaps explain the steps the method takes to login - Gabe'''
334      '''Overall good naming - Gabe'''
335
336      req_body = request.get_data()
337      body = json.loads(req_body)
338      uijson = ast.literal_eval(json.dumps(body))
339      LOGGER.info(uijson)
340      '''The line below goes past col 99 - Gabe'''
341      sql = "SELECT Password,Admin, UserID FROM Base_User where LoginId='"+ uijson['loginid'] + "';"
342      LOGGER.info(' SQL to check logged in user {%s}', sql)
343      result = DBINSTANCE.get_data(sql)
344      LOGGER.info('The userid is {%s}', result[0][1])
345      if result == 0:
346          return 'Incorrect credentials'
347      elif uijson['password'] == result[0][0]:
348
349              '''The indentation here is not consistent with the indentation above - Gabe'''
350
351              sessioncontroller.create_session(result[0][2])
352              '''This line below exceeds 79 characters - Gabe'''
353              response = {'message': "Logged in successfully'"+ uijson['loginid'], 'roleid': result[0][1]}
354              return response
355
```

def create_session()

def create_session()
Consistent identation and names
Maybe add comments describing how the userid that is being passed in is used.
Snake case used for the method name.

```
 9  def create_session(userid):
10      """This method creates a new session for that particular username."""
11
12      '''Perhaps explain how userid is used in this method - Gabe'''
13      '''Good consistent names and indentation'''
14      req_body = request.get_data()
15      body = json.loads(req_body)
16      uijson = ast.literal_eval(json.dumps(body))
17      session['loginid'] = uijson['loginid']
18      session['userid'] = userid
19      LOGGER.info (' In create_session {%s}', session)
20
```

def get_all_rooms()
Inconsistent indentation for lines 38, 41, and 51 compared to lines 44 and 54.
The comment at the top gives a general overview of the method
Perhaps expand upon it and go into more detail and list the search attributes being used.
Consistent names.
Snake case used for the method name.

```
19  def get_all_rooms():
20      """Get all rooms for the given search attributes."""
21
22      ''' List which attributes are being used to search
23          to better describe the method '''
24
25      ''' lines 38, 41 and 51 have different indentations
26          compared to lines 44 and 54'''
27
28      req_body = request.get_data()
29      body = json.loads(req_body)
30      uijson = ast.literal_eval(json.dumps(body))
31      LOGGER.info(' JSON from frontend {%s}',uijson)
32      filter = ''
33      sql = 'SELECT * FROM RoomListing WHERE (Available=0) and '
34      if 'zipcode' in uijson.keys():
35          filter = filter + '( zipcode=' + str(uijson['zipcode']) + ' )  '
36      if 'location' in uijson.keys():
37          filter = filter + '( location like ' + "'%" + uijson['location'] \
38              + "%') "
39      if 'type' in uijson.keys():
40          filter = filter + 'and (Type like ' + "'%" + uijson['type'] \
41              + "%') "
42      if 'desc' in uijson.keys():
43          filter = filter + 'and (Description like ' + "'%" + uijson['desc'
44                  ] + "%') "
45      if 'price' in uijson.keys():
46          filter = filter + 'and (Price=' + str(uijson['price']) + ')'
47      if 'size' in uijson.keys():
48          filter = filter + 'and (Type=' + str(uijson['size']) + ')'
49      if 'numbathrooms' in uijson.keys():
50          filter = filter + 'and (NumBathrooms=' \
51              + str(uijson['numbathrooms']) + ')'
52      if 'numbedrooms' in uijson.keys():
53          filter = filter + 'and (NumBedrooms=' + str(uijson['numbedrooms'
54                  ]) + ')'
55      LOGGER.info(' Filter to get rooms: {%s}',filter)
56      sql = sql + filter
```

```
    ''' lines 38, 41 and 51 have different indentations
        compared to lines 44 and 54'''

    req_body = request.get_data()
    body = json.loads(req_body)
    uijson = ast.literal_eval(json.dumps(body))
    LOGGER.info(' JSON from frontend {%s}',uijson)
    filter = ''
    sql = 'SELECT * FROM RoomListing WHERE (Available=0) and '
    if 'zipcode' in uijson.keys():
        filter = filter + '( zipcode=' + str(uijson['zipcode']) + ' )  '
    if 'location' in uijson.keys():
        filter = filter + '( location like ' + "'%" + uijson['location'] \
            + "%') "
    if 'type' in uijson.keys():
        filter = filter + 'and (Type like ' + "'%" + uijson['type'] \
            + "%') "
    if 'desc' in uijson.keys():
        filter = filter + 'and (Description like ' + "'%" + uijson['desc'
                ] + "%') "
    if 'price' in uijson.keys():
        filter = filter + 'and (Price=' + str(uijson['price']) + ')'
    if 'size' in uijson.keys():
        filter = filter + 'and (Type=' + str(uijson['size']) + ')'
    if 'numbathrooms' in uijson.keys():
        filter = filter + 'and (NumBathrooms=' \
            + str(uijson['numbathrooms']) + ')'
    if 'numbedrooms' in uijson.keys():
        filter = filter + 'and (NumBedrooms=' + str(uijson['numbedrooms'
                ]) + ')'
    LOGGER.info(' Filter to get rooms: {%s}',filter)
    sql = sql + filter
    LOGGER.info(' SQL to get rooms: {%s}',sql)
    result = DBINSTANCE.get_data(sql)
    if result == 0:
        return {}
    else:
        result_json = get_room_json(result, uijson)
        return json.dumps(result_json)

```

Code Review - Frontend

Sanket Naik <sanketnaik99@gmail.com>
to uyeang5678

Hi Ryan,

Please visit the following links to find the code for the Add Room functionality -

- Add Room Page
- Search Room Page

You will be able to find the UI components used for both pages in the following folder -

- Components

Do let me know if you have any questions or concerns regarding the code or the implementation of the functionality.

Thanks,
Sanket

Reply Forward

```jsx
import React, { useEffect, useState } from "react";
import "./search-rooms.css";
import RoomCard from "../../components/search/room-card";
import NoRoomsFound from "../../assets/images/no-rooms-found.png";
import FilterPanel from "../../components/search/filter-panel";
import axiosInstance from "../../axios-config";

const SearchRooms = () => {
  // This stores and sets the search input field's value
  const [inputField, setInputField] = useState("");
  // Store the filters values.
  const [typeSelect, setTypeSelect] = useState("");
  const [bedroomSelect, setBedroomSelect] = useState("");
  const [bathroomSelect, setBathroomSelect] = useState("");
  const [petSelect, setPetSelect] = useState("");
  const [smokingSelect, setSmokingSelect] = useState("");
  const [parkingSelect, setParkingSelect] = useState("");
  const [collegeSelect, setCollegeSelect] = useState("");
  const [disabilitySelect, setDisabilitySelect] = useState("");
  const [negotiationSelect, setNegotiationSelect] = useState("");
  const [restroomSelect, setRestroomSelect] = useState("");

  const [currentTab, setCurrentTab] = useState("");
  // This stores and sets the room data value which is received from the Flask API.
  const [roomData, setRoomData] = useState([]);
  // Loading boolean flag.
  const [isLoading, setLoading] = useState(false);
  const [hasNotSearched, setNotSearched] = useState(true);

  // This function calls the `getRooms` API endpoint and passes the search query to
```

Ryan Eng <uyeang5678@gmail.com> Tue, Nov 2, 9:48 PM (6 days ago)

to Sanket

Hi Sanket,
I received the links to the code, I will get back to you soon.
best,
Eanguy(Ryan) Eng

Ryan Eng <uyeang5678@gmail.com> Wed, Nov 3, 1:22 PM (5 days ago)

to Sanket

Hello Sanket,
Your code for Add rooms looks great. It is easy to understand and well organized. For your search room, it is more detailed and has great comments. I can understand what each function is used for. I don't see anything that needs to be changed. Keep it up with this well organized coding style.
best,
Eanguy(Ryan) Eng

5) Self-Check on best practices for security:

Protecting the whole database:
There is a username and password to access the database.

Password:
We are in the process of encrypting the user's password once the account is made.
User provides a password, encrypted into a hash and stored in the database
When the user logs in, the password they provide is encrypted into a hash and compared with what is in the database.

Social Security number: This can be encrypted as well at a later date.

Email: required when logging in. Not encrypted, and not shown on the website.

Messages: private, as in only logged in users who are senders or recipients can view the specific message, but not encrypted.

Confirm that you encrypt PW in the DB
As of Milestone 4, we are in progress of encrypting our passwords.

6) Self-check: Adherence to original Non-functional specs – performed by team leads

1. **Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO).**

   DONE

2. **Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers**

   DONE

3. **Selected application functions must render well on mobile devices**

   DONE

4. **Data shall be stored in the team's chosen database technology on the team's deployment server.**

   DONE

5. **No more than 100 concurrent users shall be accessing the application at any time**

   DONE

6. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users.

   ON TRACK - to be finished before 11/30. We have a terms of service page--needs to be implemented on our website.

7. **The language used shall be English.**

DONE

8. **Application shall be very easy to use and intuitive.**

   DONE

9. Google maps and analytics shall be added

   DONE

10. **No e-mail clients shall be allowed. You shall use webmail.**

    DONE

11. **Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.**

    DONE

12. **Site security: basic best practices shall be applied (as covered in the class)**
    a. sql injection (input sanitation)

    ON TRACK - to be finished before 11/30

13. **Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development**

    DONE

14. **The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Fall 2021. For Demonstration Only" at the top of the WWW page. (Important so not to confuse this with a real application).***

    DONE