**Final Demo**

**Good afternoon, everyone. This is team 8. Our project is to develop an online ordering system for those small hotel business owners and their customers.**

**You can type in the URL we provide to your browser and access our application.**

**When you enter our page, you will see the menu and the content. The highlighted blue label on the top menu indicates you are on the home page.**

1. **Home Page**

Here, you will see some basic information about the hotel, including its introduction, photos, customer feedback, and the contact information of the hotel owners.

1. **OAuth**

You will also notice that there is a login label on the top and a small icon on the far right. Currently, since you aren’t logged in, it will show the default image.

Now, let’s try to log in with OAuth. After clicking the login, and then we will be redirected to the Google Page. When we choose our account and come back. Now you will see you have already successfully logged in with your Google account. We also have created a user in our DB. When you move your cursor and hover over the icon, you will see your profile as the user. Also, the login button has changed to logout, you can choose to exit your account anytime.

Once you choose to log out, you will no longer see your profile and your order history page will display a 403 error if you want to access it.

1. **Make an Order**

***Selection Bar***

Then, let’s go to the room list Page to make an order. On this page, there is a selection bar, a room list layout that later will be generated, and the shopping cart on the right. Before making his order, the customer might have some conditions or preferences. On the selection bar, they can input the conditions and the qualified room will be returned. If we enter nothing, we will return any rooms that are available for today. As you can see on the page, since all rooms are currently not booked, you can see all the rooms are returned.

But, let’s say we want to have some standard rooms that face South, we use the drop-down menu and select the conditions, standard and south. Now let’s click on the search button again. Only A103 is returned as it is the only room that both satisfied the Standard and South conditions.

***Room Details***

If you want to know the details information about a certain room, just move your cursor and click the room name. The room details card will pop up and show you the room picture and other information. Customers can know more about the room and compare different rooms by using this function.

***Shopping Cart***

Now let’s try to check out some rooms. Let’s say, we want to have A102 and A 103, we add them to the shopping cart. Now their information will be displayed on the shopping cart and customers can also see the total amount of orders.

If they change their minds, they can also click on the cancel button to delete the room from the shopping cart. Correspondingly, the total will be deducted by the price of the room which has just been canceled.

***Checkout – Stripe***

Let’s check out our selected rooms. A102 and A103. By clicking the checkout button, we will be directed to the Stripe Payment Page. Here, I will enter my username, card number, and all the information that is needed to finish the checkout. Feel free to remember and forget, these are all fake numbers that Stripe provides to the developers. As you can see from the top left, we are in test mode.

After we finish the transaction, if successful, we will be back on our website and see the successful page.

**Order History**

Now, if we go to the order list Page, we can see these two rooms we just booked.

Room A102 and A103 for today.

Customers can see their order history and payment status here. They can also use the selection bar to filter and check their orders.

Now if we go back to the room list Page, now these two rooms for today are gone. If you really want them, you can only have them on another day.

1. Security Feature
2. Others

Now let’s go to review the codes. In our GitHub repo, you will see there are two folders, frontend and backend. We have divided the project into two part, the frontend back we are using React and for the back end, we are using Django.

Here are something that I want to share with you for our codes.

1. **Router**

Unlike the homework, since we are using React as the frontend framework. We are no longer using the flower bracket with the URL name to redirect the users. Here, we use the router to allow us to jump between each page.

文本

描述已自动生成

At index,js file, We put the Router label on the root. Everytime the react application runs, it will start this router and the router will help to reach different JS files for different pages.

图形用户界面, 文本

描述已自动生成

You will see we have put all the url and the element name which can refer to different js file on the BaseRouters. This is like a traffic police officer, he knows all the destinations you want to go.

You can also see we export this BasicRoute

We also need to put this BaseRouter on every page, which also the users can go to another page by asking this BaseRouter.

1. **get\_csrf\_token**

On this BaseRouter.js file, you can also see there is a useEffect that will be triggered once when the method was called. This method will be called only once when the user first access our website and the home page is rendered. Since we have separated our front and back, we need to design a way that allows them to communicate. And we need to make sure these communications are safe and secure. For all the communication that happen between the front and back, they will all need to carry with the csrf-token, or fail.

So here you will see when the hook is triggered, The front use axios to send a request to the back. On the back we using the Django middle ware to generate the csrf token and store it in a cookie, and see it back. When axios knows it succeeds, it will get the csrf token from the cookies and store it into your local browser storage. Which you can see if you inspect your network activity.

文本

描述已自动生成文本

描述已自动生成

1. **OAuth**

We also do a lot of research on OAuth, read the document of Google OAuth and analyze the response message from the Google API.

On the

1. **Return Available room Logic**

On the model.py you can see our Database Design. Payment, Room and Order.

When users select the conditions on the drop-down menu on room list Page. After clicking the search button, From the frontend Page, we will send a GET request carried with these parameters.

屏幕上有字

描述已自动生成

Then we first pull all the rooms from our database, we use these conditions to filter the qualified. If the user does not select time, we will set today’s time as default. Finally, we will get a query set with the qualified room.

Now, these rooms are qualified. However, how do we know which are available and which are booked?

文本

描述已自动生成

Then, here comes the most important part of this method.

We first use the user’s required start time and the user’s required end time to get all the orders whose end time is earlier than the user’s required start time and start time later than the user’s required end time. Now we get all the un-overlapped orders. Then we exclude it. We will get the orders whose timeslot will be overlapped, or we can say, conflict with the user’s required time.

Then, we get a bunch of ‘bad’ room IDs from these overlapped orders.

Finally, we will use the query set of these qualified rooms we got previously and filter out these ‘bad’ rooms, finally, we will get the qualified and available rooms we need, and we serialize these rooms into JSON and send them back to the front.

1. **Multiple hooks for the room list Page**

For the content of our room list Page, there are three major components, the selection bar, the room list display layout, and the shopping cart. Here we utilize plenty of variables and hooks to realize the search and the display functions.

***List,js***

On our list,js file, you can see when user go to this page, axios will send a request, get empty data and reset the variables first. Since the logic is quite similar, More details will be provided on the Select.js demonstration.

文本

描述已自动生成

***Selection,js***

As you can see from select.js file, this is for the selection bar component. It is responsible for getting the users selected variable from the selection bar, the room type, direction, price range, startTime and endTime and then axios will send request to the back and access the data with all the available rooms.

Then the room list url will invoke Django view.py get\_room\_list method to search for the qualified and available rooms.

If success, React will set the Data and List with returning data. Also, it cleared the rooms and isAddOrder list to empty. These two lists are for the shopping cart display. It will implement a for each loop to initialize the isAddOrder list based on the number of available rooms return.

Here, the isAddOrder is a list containing with a dictionary. The key is the room index on the display layout and the value is a Boolean variable. Ture indicate these room are not added in the shopping cart.

文本

描述已自动生成

**Room\_list.js**

This is for checkout and connect with Stripe.