PROJECT REPORT

ON

HOTEL

RESERVATION

APPLICATION

Submitted By: Sahith Vardhan Reddy Vancha

KSU ID 001079049

MS in Software Engineering

Kennesaw State University

Background and Overview:

The fact that guests are necessary for a successful hotel operation is not exactly a surprise. The use of a reservation system is crucial to your prospects of luring and maintaining those customers. Let's look at why.

A hotel reservation system is exactly what?

A hotel reservation application is a sophisticated piece of software that allows customers to book their arrival and departure dates and collects money. They can select their rooms when making their reservations, and more sophisticated systems even let customers choose extras like wine, flowers, fruit, or other luxuries to be delivered to their room when they arrive

Hotel Reservation is an end-to-end application where a customer can perform the following actions

- search for hotels
- Log in to the application
- Select the type of room
- Book the required number of rooms.
- Cancel the reservation

In order to achieve this, the backend has been coded in JAVA. I have used the Spring STS tool for the same. Front End has been coded in Angular and I have used the visual studio tool for the same. Hotel data will be stored in an MYSQL database which I will connect to the backend using a JDBC connection inside the application properties file

The main advantages of using this application are below-

- No involvement of the third party as the customer directly interacts with the hotel website

- This reduces the chance of human error to a great extent
- Improved efficiency as the data gets updated instantly compared to traditional booking methods.

Planning and Management:

Phase 1: Initial Analysis	Phase 2: Application Development	Phase 3: Verification:
Step 1.1: Brainstorming	Step 1.1: Creating the DB	Step 1.1: Test
Activity 1.1.1: Decide on the scope of the project	Activity 1.1.1: Configure DB	Activity 1.1.1: Test the back end
Activity 1.1.2: Create an activity graph	Activity 1.1.2: Create tables	Activity 1.1.2: Test the front end
	Activity 1.1.3: Insert data into tables	Activity 1.1.3: Test the DB
Step 1.2: Tech Stack	Step 1.2: Back End development	Step 1.2: Bug Fixing
Activity 1.2.1: Finalize front-end stack	Activity 1.2.1: Create Entities	Activity 1.2.1: Fix the errors found in the above steps
Activity 1.2.2: Finalize back-end stack	Activity 1.2.2: Create APIs	
Activity 1.2.3: Finalize the DB	Activity 1.2.3: JDBC connection	Step 1.3: Retest
	Activity 1.2.4: Service layer	Activity 1.3.1: Retest the application
	Activity 1.2.5: DAO layer	
	Activity 1.2.6: Connect to the front end	Step 1.4: Deploy
		Activity 1.4.1: Deploy the code into a repository
	Step 1.3: Front End development	

	Activity 1.3.1: Login Page	
	Activity 1.3.2: Home page	
	Activity 1.3.3: Booking page	
	Activity 1.3.4: Styling of pages	
	Activity 1.3.5: Code Implementation	

Below is the time cost estimate of the project in the number of days.

Activity	Time Estimate in days
Step 1: Initial Analysis	
Activity 1.1: Brainstorming	3
Activity 1.2: Finalizing tech stack	3
Step 2: Create a Database	
Activity 2.1: DB configuration	4
Activity 2.2: Create tables	4
Activity 2.3: Insert data into tables	5
Step 3: Back-End Development	
Activity 3.1: Create Entity	6
Activity 3.2: Create API	8
Activity 3.3: JDBC connection	5
Activity 3.4: Service Layer	9
Activity 3.5: DAO layer	9
Activity 3.6: Front End Connection	7
Step 4: Front-End Development	
Activity 4.1: Login Page	8
Activity 4.2: Home Page	10
Activity 4.3: Booking page	5
Activity 4.4: Styling pages	15
Activity 4.5: Develop components	11
Step 5: Testing	
Activity 5.1: Testing all components	5
Activity 5.2: Bug fixes	10

Activity 5.3: Retest	5
Activity 5.4: Deployment	6

Hardware Requirements for Hotel Reservation Application:

- RAM: Minimum 1GB or higher
- HDD: Minimum 50GB or higher
- Processor: Intel i5 or higher or AMD

Software Requirements include:

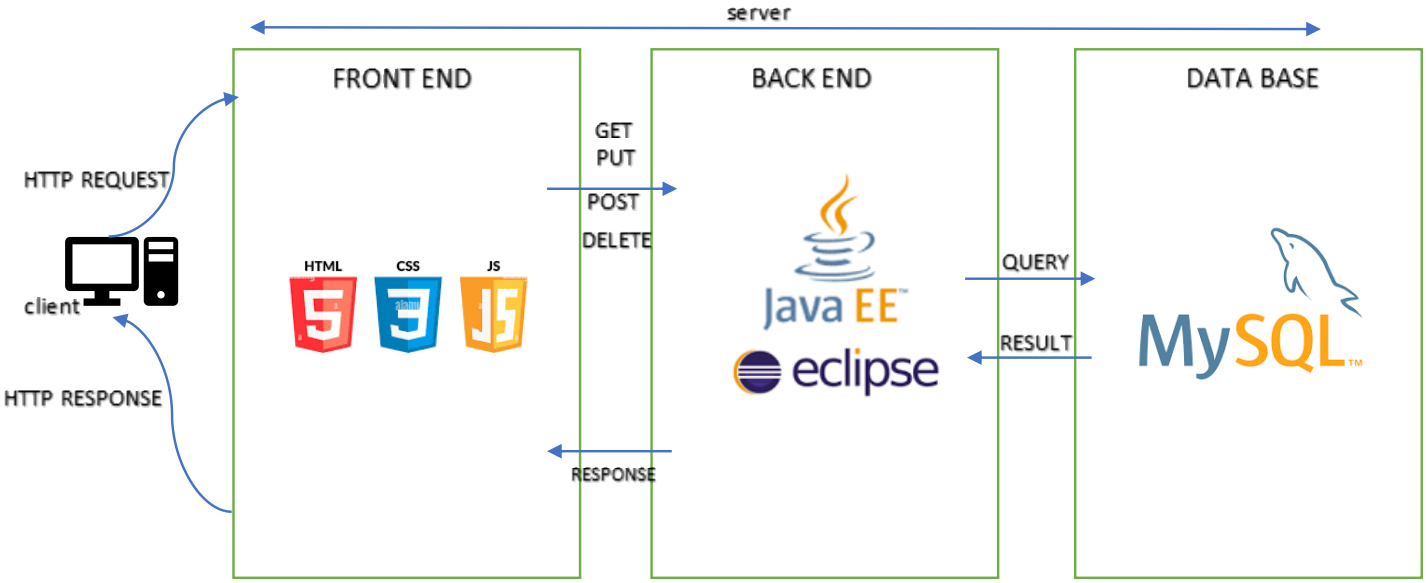
- Operating System: Windows, MAC, or any system which supports a web browser
- Web Browser: 

- Data Base: MySQL
- Web Development: Spring STS, visual studio, MySQL Workbench
- Server: Tomcat

The following are the **internal interfaces** that will be used as a part of the hotel reservation system:

- Login API – which allows the users to login
- Home API- Home page of the application where a list of hotels is displayed
- Booking API-where the user can select a particular hotel and room and see details
- Payment API-where the user can make payment for the hotel
- Orders API-list all the bookings of the customer

ARCHITECTURE DESIGN

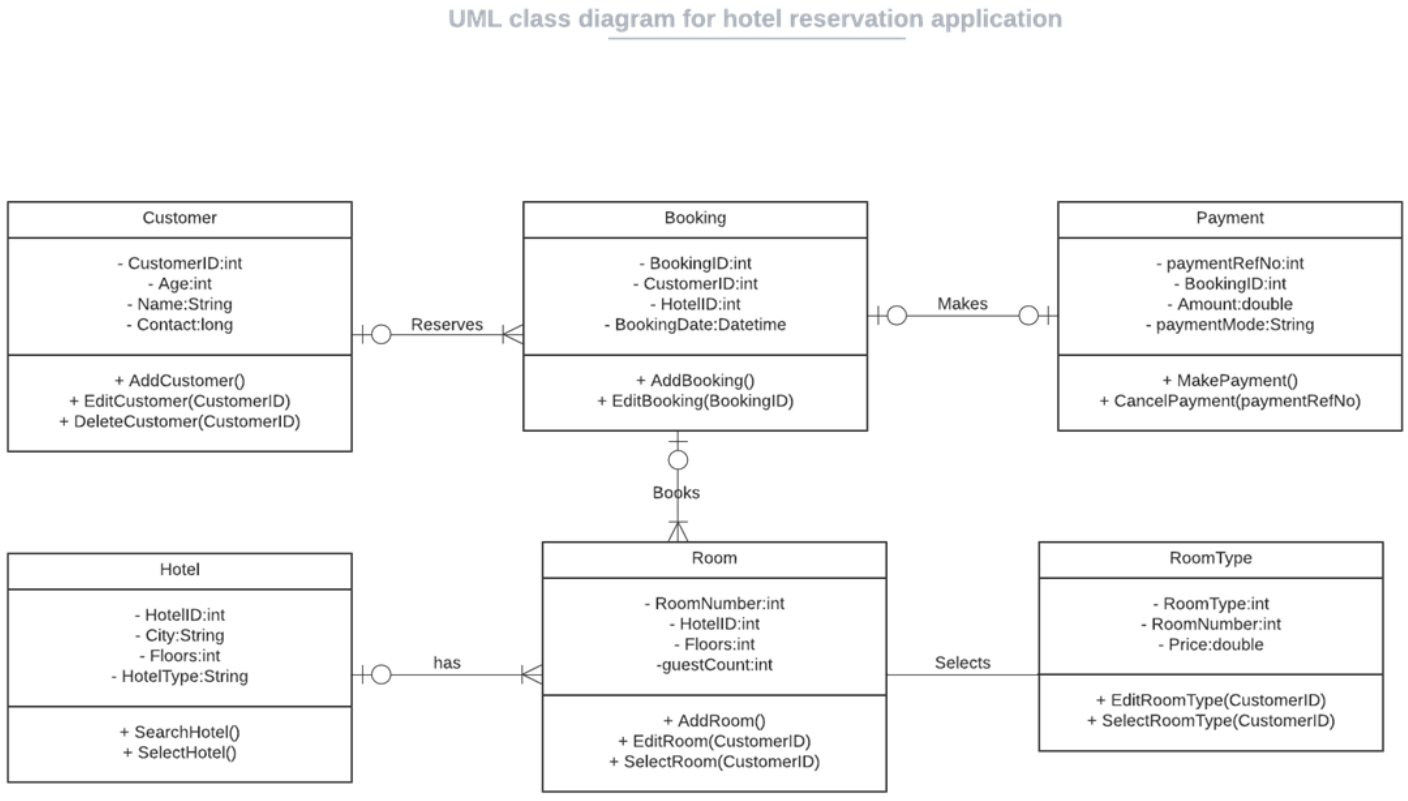


Client Server Architecture is a computing model in which the server hosts, delivers, and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or internet connection. This system shares computing resources.

The client-server model is a design of a computer network in which numerous clients demand and get administration from a concentrated server. Client PC gives a connection point to permit a PC client to demand administrations of the server and to show the outcomes the server returns. Servers trust that solicitations will show up from clients and afterward answer them. In a perfect world, a server gives a normalized straightforward connection point to clients, so clients need not know about the particulars of the framework (i.e., the equipment and programming) that is offering the support. Clients are frequently arranged at workstations or on PCs, while servers are found somewhere else on the network, for the most part on additional strong machines. This processing model is particularly powerful when clients and the server each have different things in hand that they regularly perform.

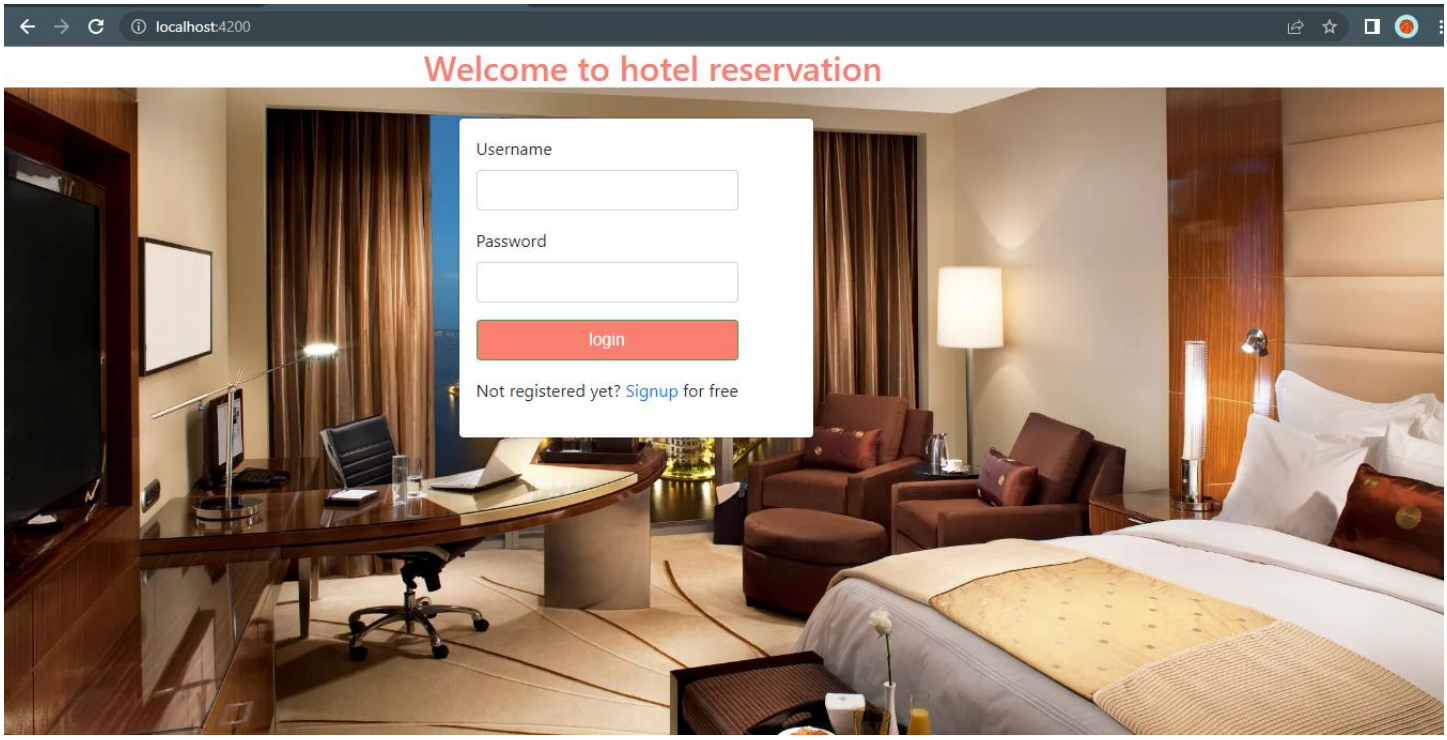
I chose the Client-server model for my project because here the clients can book the hotel from anywhere using any device like a mobile phone, web browser, iPad, etc., and the server receives the HTTP request and sends back the related HTTP response. Moreover, the clients don't know about the internal working of the server, but they are still able to use the system. In a similar way, servers don't have any details about the client. All they have is a request which gets executed and an appropriate response is sent back to whomever the client may be. So I think Client-server architecture is perfect for the hotel reservation application.

UML DIAGRAM:

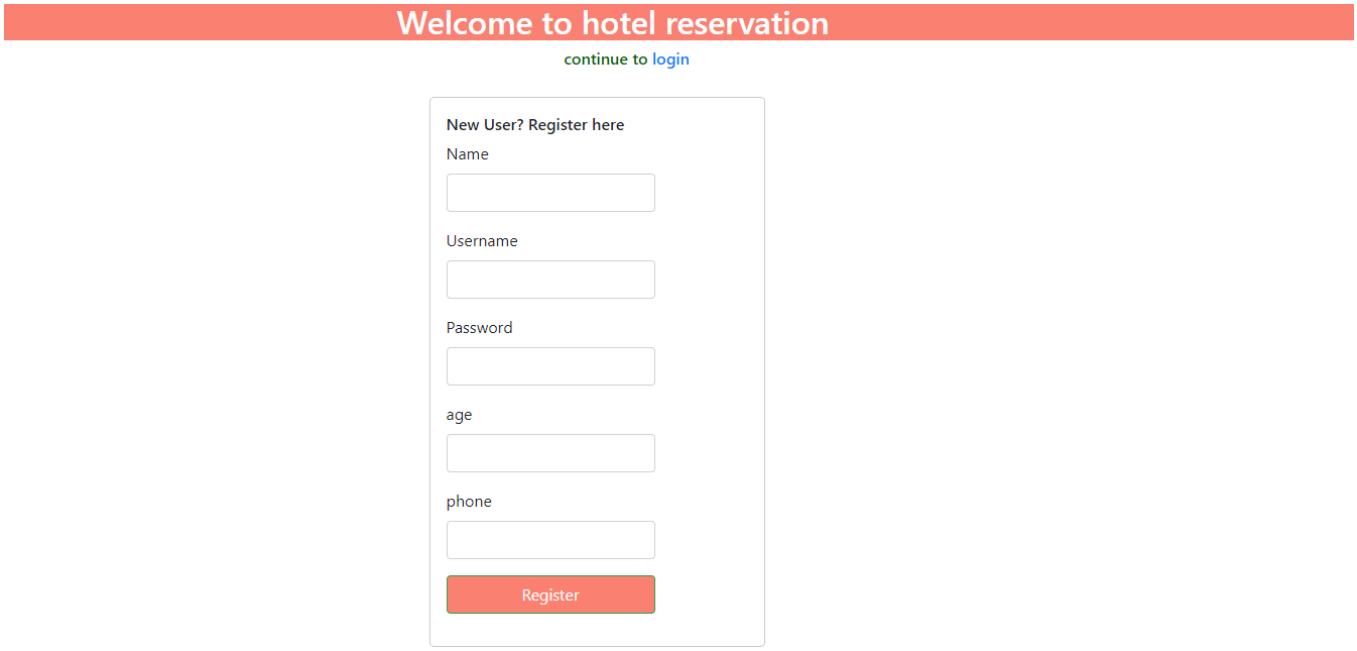


Below is an illustration of the UI and feel of the Application. Have added a video link to the end of this slide where we can see the full working of the application.

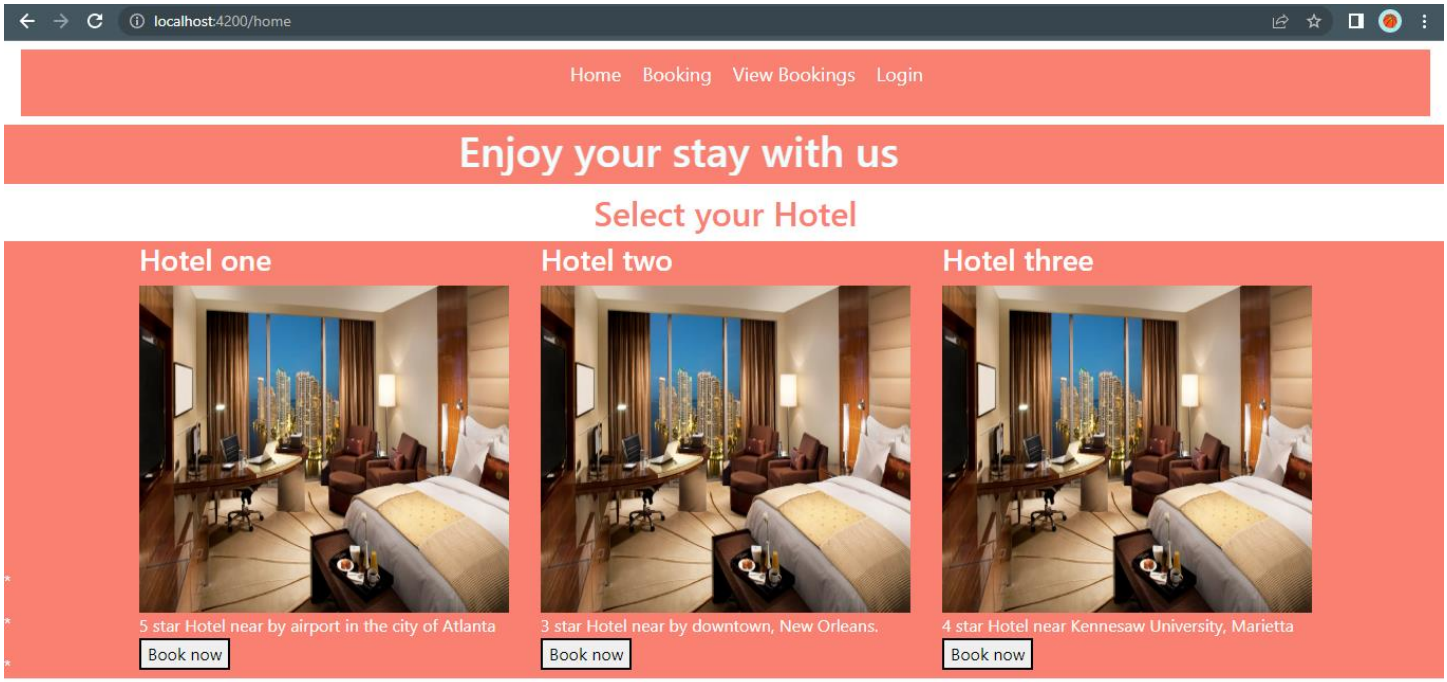
Login Page:



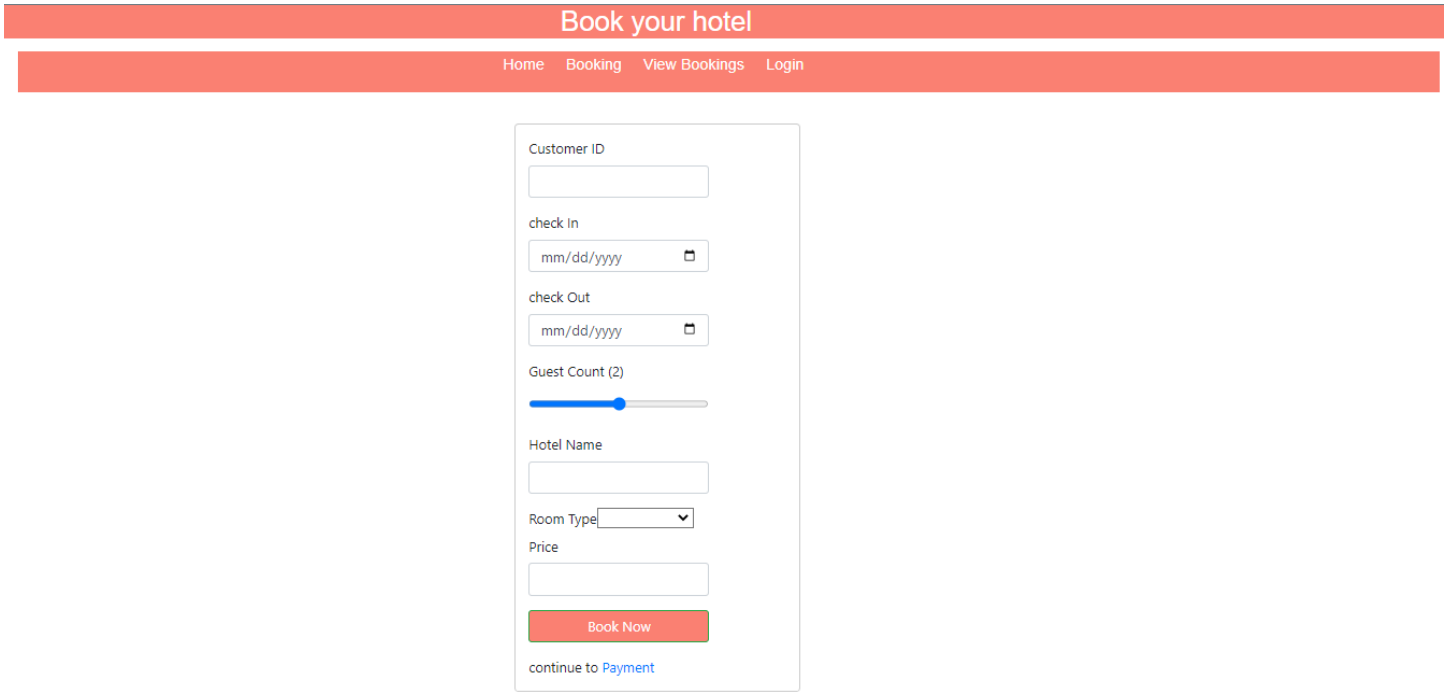
Registration Page:



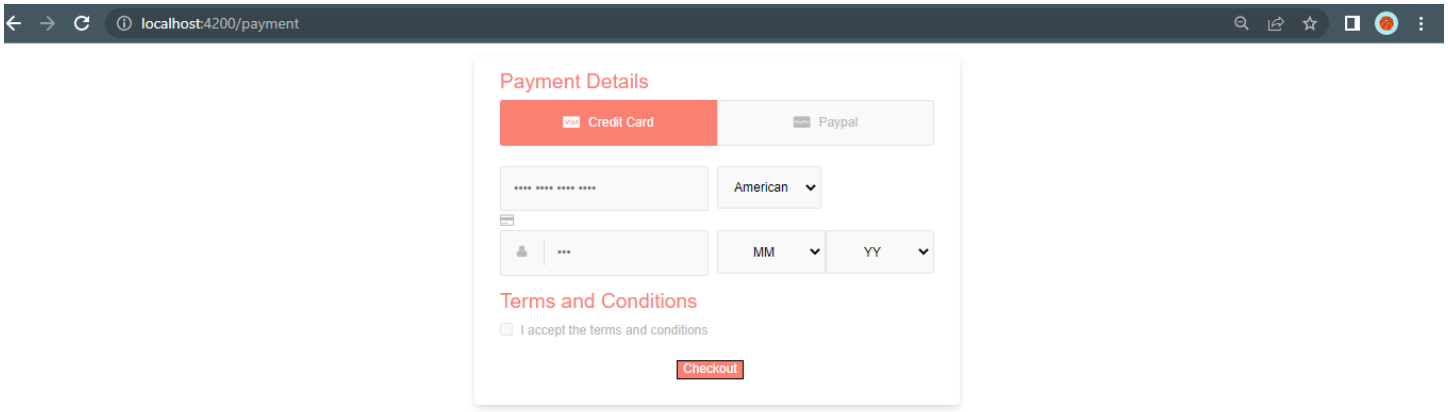
Home Page:



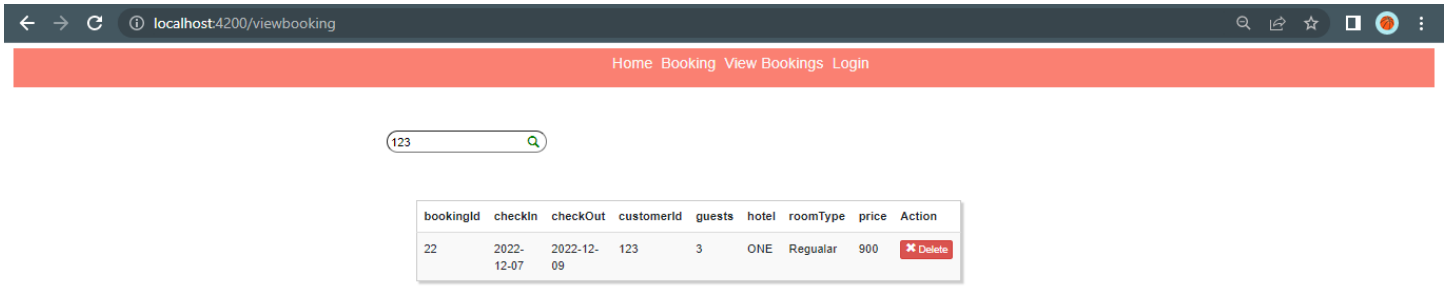
Booking Page:



Payment Page:



View Bookings Page



The following list of difficulties encountered during development:

- I had to acquire a number of new concepts due to the quick progress of technology. I once used to code in JAVA and Angular, for example. However, to improve the project's efficiency and speed, I studied typescript and Spring.
- Code commit to GIT was difficult since it kept giving errors, and I had to restart the process to figure it out.
- Initial project setup was also challenging because it required distinct versions and libraries for each module.

These were the difficulties encountered during testing:

- Despite having fixed the code, I had to run the test several times because of bugs in the program. To repair it, I had to restart the program.
- Some of the functions could not be tested. As a result, I tried them using the inspect element on the front end to see whether we could receive a success status.
- Both the front end and back end performed well when tested separately, but the true difficulty came when we tried to merge them because there were so many issues. Testing the combined code was therefore time-consuming.

Measuring the quality of project:

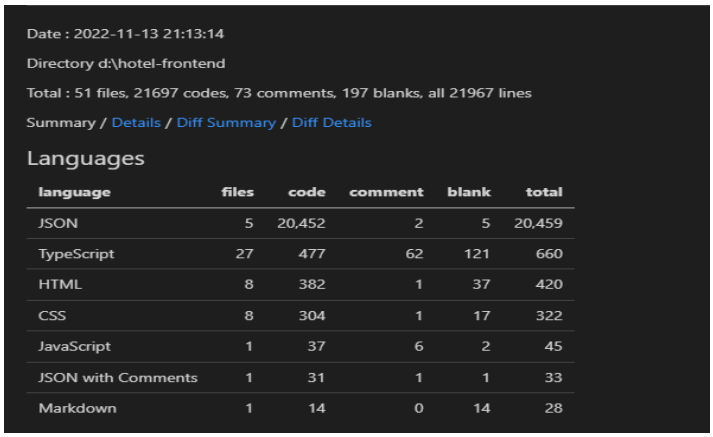
You can use a variety of metrics to assess an application's quality and code size. Lines of Code, often known as LoC, is one of the well-liked metrics that you can employ.

Any line of text in a piece of code that is neither a comment nor blank line, as well as header lines, is referred to as a line of code (LOC), regardless of the number of statements or fragments of statements on the line. All lines containing the declaration of any variable, as well as executable and non-executable statements, are categorically included in the LOC. You can only compare or estimate projects that use the same language and are coded following the same coding standards since Lines of Code (LOC) only counts the volume of code.

Typically, the number of lines of code in a software project can be used to assess its size. It is one of the measures that customers might use to figure out how much they can spend on a specific software project. Code components like classes and functions will always be present in software projects. The functions and classes with more code lines are typically more difficult to read and comprehend. Therefore, it is typically advisable to divide such classes or functions into smaller chunks for easier comprehension and reading. One of the most trustworthy metrics for evaluating the quality of a software application is the LoC. You'll frequently find that even when working on the same project, experienced engineers tend to write less code than novices. An effective language requires composing less code to achieve an errand that would have to compose a few lines of code in the event that is written in another dialect that is less proficient.

To calculate the LOC, I have used the Visual studio code extension VSCodeCounter which gives the Lines of Code as shown below in the screenshot

Front End Lies of Code:



Date : 2022-11-13 21:13:14

Directory d:\hotel-frontend

Total : 51 files, 21697 codes, 73 comments, 197 blanks, all 21967 lines

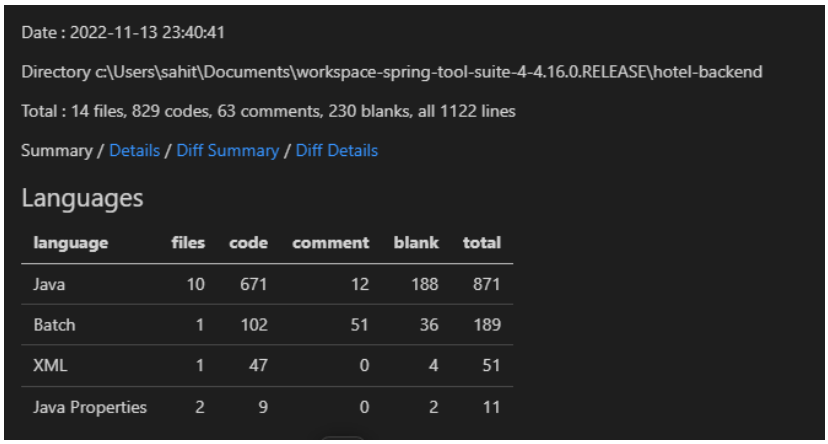
Summary / Details / Diff Summary / Diff Details

Languages

language	files	code	comment	blank	total
JSON	5	20,452	2	5	20,459
TypeScript	27	477	62	121	660
HTML	8	382	1	37	420
CSS	8	304	1	17	322
JavaScript	1	37	6	2	45
JSON with Comments	1	31	1	1	33
Markdown	1	14	0	14	28

The above diagram shows the lines of code in each language for the entire project. This implies that there are about 660 lines of code in TypeScript, 420 lines of code in Html, 322 lines of code in CSS, and 45 lines of code in JS

Back-end Lines of Code:



Date : 2022-11-13 23:40:41

Directory c:\Users\sahit\Documents\workspace-spring-tool-suite-4-4.16.0.RELEASE\hotel-backend

Total : 14 files, 829 codes, 63 comments, 230 blanks, all 1122 lines

Summary / Details / Diff Summary / Diff Details

Languages

language	files	code	comment	blank	total
Java	10	671	12	188	871
Batch	1	102	51	36	189
XML	1	47	0	4	51
Java Properties	2	9	0	2	11

The above diagram shows the lines of code in each language for the entire project. This implies that there are about 880 lines of code in java and 400 lines of code in other languages. Overall there are more than ~3000 lines of code both front-end and backend combined. Once we fix all the bugs and make some corrections the project will be nearby~4500 lines of code.

Although LOC is an effective metric in software engineering for project estimation it has the below disadvantages

- It is very challenging to infer the final program's LOC from the problem definition.
- It has a low correlation to code quality and effectiveness.
- It makes no allowance for intricacy.

GitHub Link: I have deployed my code in the Git Hub

https://github.com/sahith-vancha/SWE6623-HotelApplication

YouTube Link for the demo of the UI:

https://youtu.be/_m1asqb7qRE