

ADAPT TRAINING

Java Full Stack Development

Capgemini

A training report

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of technology Hons.

(Electronics and Communication Engineering)

Submitted to

Lovely Professional University



L OVELY
P ROFESSIONAL
U NIVERSITY

From 1/08/2021 to 4/26/2021

Submitted By

Name of student: Karampudi Jyothi Gangarsha

Registration Number: 11701442

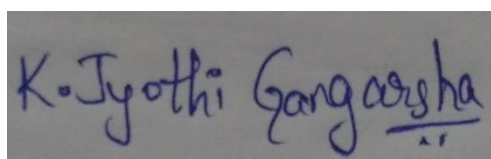
Signature of the student:

Student Declaration

To whom so ever it may concern

I, **Karampudi Jyothi Gangarsha, 11701442**, hereby declare that the work done by me on “ADAPT Training” from January,2021 to May,2021, under the supervision of Amit Mahadik, Manager, Capgemini, and Puneet Kumar, Assistant Professor, Lovely Professional University, Phagwara, Punjab, is a record of original work for the partial fulfilment of the requirements for the award of the degree, Bachelor of Technology Hons.

Name: Karampudi Jyothi Gangarsha (11701442)

A photograph of a handwritten signature in blue ink on a light-colored surface. The signature reads "K. Jyothi Gangarsha" with a horizontal line under the last name and a small mark below it.

(Signature of student)

Dated: 5/22/2021

Supervisor Declaration

To whom so ever it may concern

This is to certify Karampudi Jyothi Gangarsha, 11701442 from Lovely Professional University, Phagwara, Punjab, has worked as a trainee in Capgemini on “ADAPT Training” under my supervision from January 2021 to May 2021. It is further stated that the work carried out by the student is a record of original work to the best of my knowledge for the partial fulfilment of the requirements for the award of degree, Bachelor of Technology Hons.

Name of the External Supervisor:

Amit Mahadik

Name of the Internal Supervisor:

Puneet Kumar

Designation of the External Supervisor:

Manager, Capgemini

Designation of the Internal Supervisor:

Associate professor, Lovely Professional University

Signature of the External Supervisor:

Signature of the Internal Supervisor:

Dated:

Dated:

Acknowledgement

I pay deep sense of gratitude to my Internal Mentor Mr. Puneet Kumar, Lovely Professional University and External Mentor Mr. Amit Mahadik, Capgemini who gave me opportunity to complete this On Job Training “ADAPT Training” which enriched me in Full Stack Development, and it is the First Milestone in my career. I would also thankful to my batch mates helped in solving my critical doubts and putting me forward in the same level as them.

List of Figures

Figure 1.1 Company logo	Figure 4.24 addManager.js
Figure 1.2 Paris business unit	Figure 4.25 Home page
Figure 1.3 Capgemini hierarchy	Figure 4.26 Login Page
Figure 3.1 Training plan	Figure 4.27 owner page
Figure 4.1 Project (block) Flow	Figure 4.28 Add Manager form
Figure 4.2 package description	Figure 4.29 Employee page
Figure 4.3 owner controller	Figure 4.30 Rooms page
Figure 4.4 owner model	Figure 4.31 Rooms Availability
Figure 4.5 owner repository	Figure 4.32 checked-in guests
Figure 4.6 exception handler	Figure 4.33 Guest log
Figure 4.7 Security configuration	Figure 4.34 Checkout
Figure 4.8 Jwt util class	Figure 4.35 Payment
Figure 4.9 Jwt Filter	Figure 4.36 Payment confirmation
Figure 4.10 customUserDetails service	Figure 4.37 Suggestion box
Figure 4.11 Main Class	Figure 4.38 UI testing
Figure 4.12 application.yaml file	Figure 4.39 Backend testing
Figure 4.13 Controller testing	Figure 4.40 Swagger UI
Figure 4.14 mongo db compass	Figure 4.41 Eureka client
Figure 4.15 postman	
Figure 4.16 mysql workbench	
Figure 4.17 react components	
Figure 4.18 manager ops	
Figure 4.19 Guest ops	
Figure 4.20 other ops files	
Figure 4.21 Owner, paypal, service files	
Figure 4.22 App.js file	
Figure 4.23 Guest Service.js	

Contents

Student Declaration	02
Supervisor Declaration	03
Acknowledgement	04
List of Figures	05
Chapter I Introduction of the Company	08
1.1 Capgemini	08
1.2 Company's Vision and Mission	08
1.3 Origin and growth of company	09
1.4 Various departments and their functions	10
1.5 Organization chart of the company	11
Chapter II Introduction of the Project Undertaken	13
2.1 Objectives of the work undertaken	13
2.2 Scope of work	13
2.3 Importance and Applicability	13
Chapter III Technological Stack	14
3.1 Git & GitHub	15
3.2 Mongo DB	16
3.3 Core JAVA	17
3.4 Spring	19
▪ 3.4.1 Spring Core	19
▪ 3.4.2 Spring Mvc	20
▪ 3.4.3 Spring Rest	20
▪ 3.4.4 Spring Security	22

▪ 3.4.5 Spring Cloud	23
3.5 Web Development	24
▪ 3.5.1 HTML	24
▪ 3.5.2 HTML5	24
▪ 3.5.3 CSS	24
▪ 3.5.4 JS	25
▪ 3.5.5 TS	25
▪ 3.5.6 ES6	25
3.6 Angular	25
3.7 React	26
3.8 Docker	27
3.9 Rabbit Mq	28
Chapter IV The Online Hotel Management System	29
4.1 Project Flow	29
4.2 Backend Implementation	30
4.3 Frontend Implementation	39
Chapter V Conclusion	48
References	48

Chapter I

Introduction of the Company

1.1 Capgemini

Capgemini is a worldwide pioneer in collaborating with organizations to change and deal with their business by saddling the force of innovation. The Group is guided regularly by its motivation of releasing human energy through innovation for a comprehensive and feasible future. It is a dependable and assorted association of 270,000 colleagues in almost 50 nations. With its solid long term legacy and profound industry mastery, Capgemini is trusted by its customers to address the whole broadness of their business needs, from methodology and plan to tasks, energized by the quick advancing and imaginative universe of cloud, information, AI, network, programming, computerized designing and stages. The Group detailed in 2020 worldwide incomes of €16 billion.



Figure1.1 Company logo

1.2 Company's Vision and Mission

Our Vision: The business worth of innovation comes from and through individuals.

Capgemini comprehends that business esteem can't be accomplished through innovation alone. It begins with individuals: specialists cooperating to get to the core of your individual business targets and foster the most adjusted answers for fit these prerequisites.

Our Mission: with you, we make and convey business and innovation arrangements that fit your requirements and drive the outcomes you need.

Capgemini empowers you to change your association and improve execution. We intend to engage you to react all the more rapidly and instinctively to changing business sector elements. By supporting your capacity to bridle the correct innovation, we assist you with getting coordinated and serious. Cooperation is vital to the manner in which we work together. Our specialists unite with your kin to frame a durable group. We consider this methodology the Collaborative Business Experience®. It shows in our every cooperation and is our method of producing nearer, more powerful connections. Individuals matter, results count.

1.3 Origin and growth of company

Capgemini SE is a French multinational corporation that provides consulting, technology, professional, and outsourcing services. It is headquartered in Paris, France. Capgemini has over 270,000 employees in over 50 countries, of whom nearly 120,000 are in India.

History

Capgemini was established by Serge Kampf in 1967 as a venture the executives and information preparing organization. The organization was opened as the Société pour la Gestion de l'Entreprise et le Traitement de l'Information (Sogeti). In 1974 Sogeti procured Gemini Computers Systems, a US organization situated in New York. In 1975, having made two significant acquisitions of CAP (Center d'Analyse et de Programmation) and Gemini Computer Systems, and following goal of a question with the likewise named CAP UK over the global utilization of the name 'CAP', Sogeti renamed itself as CAP Gemini Sogeti. Cap Gemini Sogeti dispatched US activities in 1981, following the procurement of Milwaukee-based DASD Corporation, represent considerable authority in information transformation and utilizing 500 individuals in 20 branches all through the US. Following this procurement, The U.S. Activity was known as Cap Gemini DASD. In 1996, the name was rearranged to Cap Gemini with another gathering logo. All working organizations overall were re-marked to work as Cap Gemini. Ernst and Young Consulting was obtained by Cap Gemini in 2000. It at the same time coordinated Gemini Consulting to shape Cap Gemini Ernst and Young. In 2017, Cap Gemini S.A. became Capgemini SE, and its Euronext ticker name comparatively changed from CAP GEMINI to CAPGEMINI. In 2019, Capgemini gets Altran bringing the all out representative check to more than 250,000. This is the biggest procurement in the organization's history. In July 2020, Capgemini revealed that it has been named as a pioneer in Everest Group's Guidewire IT administrations called "Guidewire Services PEAK Matrix® Assessment 2020 – Setting the Key Phase on Cloud.”



Figure 1.2 Paris business unit

Management

The Capgemini Group Executive Committee comprises of 26 members. On 20 May 2020, Aiman Ezzat was designated as the new CEO. He is related with Capgemini for more

than 20 years. From 2005 to 2007, Aiman was Capgemini's Deputy Director of Strategy. In November 2007, Ezzat was designated COO of the Financial Services Global Business Unit, and turned into its Global Head in December 2008 till 2012. From January 2018 to May 2020, he filled in as Chief Operating Officer and before this as Chief Financial Officer, from December 2012 to 2018. From 2012 to 2020 Paul Hermelin filled in as the Group Chairman and CEO. He joined Capgemini in 1993 and was delegated as its CEO in 2002. In May 2012, Hermelin became executive and CEO of the Capgemini Group. He succeeded Serge Kampf, who filled in as the Vice Chairman of the Board until his demise on 15 March 2016.

1.4 Various departments and their functions

Six Sectors of Activity

- Consumer Products, Retail & Distribution: companies in these sectors are facing new constraints linked to productivity and the ever-increasing pace at which their customers' expectations are changing. Capgemini provides them with the technology and expertise they need to access e-commerce platforms that combine flexibility and speed;
- Energy, Utilities & Chemicals: using the latest technology, the Group helps companies in these sectors to overcome the twofold challenge of constantly changing regulations and increasingly stringent environmental standards. Capgemini is the world leader in IT systems for electricity meters known as "smart meters";
- Financial Services (Insurance and banking): the Group supports the rationalization and simplification of financial institutions' applications and infrastructure (particularly within insurance and banking). Our main fields of expertise are mobility, intelligent data management, client experience improvement and regulation compliance;
- Public Sector: Capgemini assists administrations, companies and public agencies, as well as major local authorities, to implement their programs and plans for modernization, with an ever growing focus on digital;
- Manufacturing, Automotive and Life Sciences: these activities may be fertile ground for innovation, but improving competitiveness is a constant challenge. Through our outsourcing and Business Services, among others, Capgemini provides companies in these sectors with a wide array of solutions. These solutions enable companies to make significant savings on IT systems management and support functions, in addition to steering their digital transformation journey towards smart, connected products and plants;

- Telecommunications, Media and Entertainment: operators in this sector are faced with declining revenues from their traditional activities, competition from new players and the saturation of their networks due to the explosion of content. Having supported the sector for over thirty years, Capgemini offers access to in-depth knowledge of telecommunications and digital content, as well as our technological expertise with regard to networks.

1.5 Organization chart of the company

Capgemini Hierarchy structure

Capgemini is a global organization of French beginning and works in the field of consultancy. The settle of Capgemini is arranged in Paris, the capital of France, however it has branch workplaces practically everywhere on the world. The organization utilizes around 145000 individuals in 40 nations of the world. Like most different organizations of such a height, Capgemini additionally have a various levelled inward construction. From the level of the most minimal representative to the most significant level a pyramid like construction exists, and accordingly to reach at a particular level, you need to work for at times in a lower segment. The various levelled design of the organization is clarified underneath.

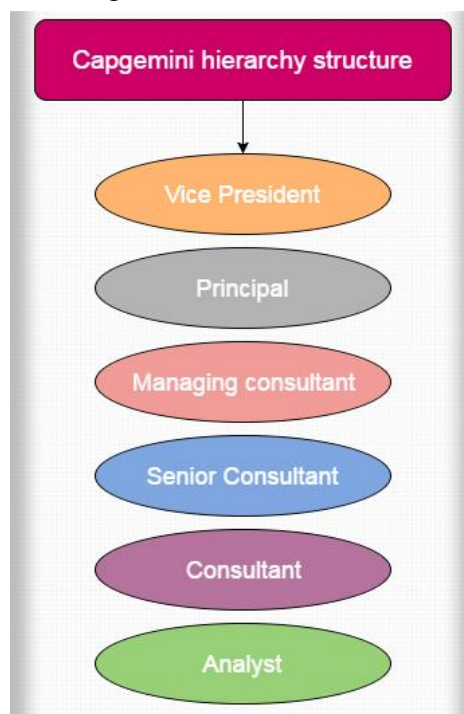


Figure 1.3 Capgemini hierarchy

At the most elevated level of a profession in Capgemini progression, there is the Vice president, trailed by the head, the overseeing specialist, and others, with the examiner at the least level of the order. VP At the Capgemini Consultancy Company, the Vice President is the

highest pioneer. There can be more than each VP in turn, giving administration to the organization. It is their obligation to direct the organization towards the correct channels and roads. To turn into a Vice president in the Capgemini Page 16 of 21 organization, just expertise and ability isn't adequate, however long periods of involvement and a decent history is additionally required.

Head The prompt second position to the VP in the organization has a place with the Principal. The person in question is additionally a significant head of business. It is the obligation of the rule to deliver activities of high effect and significant levels and along these lines framing food business relations with significant organizations. It is additionally the duty of the chief to make groups which are appropriately able to manage explicit undertakings.

Overseeing specialist Next in the various levelled design of the organization is the overseeing advisor. In the organization, this position is regarded by all as it is a vital one. The overseeing specialist has various obligations, for example, demonstrating administration to a group of advisors, expanding the volume of business with the assistance of extra deals, ensuring that the activities are conveyed at the correct time, and overseeing other significant parts of the business.

Senior Consultant-The following individual in the organization chain of importance is the senior specialist. To turn into a senior advisor one necessities long stretches of involvement alongside explicit abilities. A senior expert can chip away at his own or with a group, contingent upon the requests of the venture. On occasion they head little groups and makes sure that cut-off times are met. The individual in question additionally manages the customer.

Expert Under the direction of a senior advisor works at least one specialists. They work in a group utilizing their abilities and information and frequently need to comprehend the prerequisite of the customer and work likewise. Examiner an investigator remains at the most reduced level of the organization order. In certain spots this position doesn't exist, in many others they are administered by experts and senior specialists.

This is the order construction of Capgemini. Likewise Know about Tata Consultancy administrations various levelled structure.

Chapter II

Introduction of the Project Undertaken

2.1 Objectives of the work undertaken

The objective is to build a Fully Functional Full Stack Web Application from Scratch Individually, using the following Technological stack:

- Spring Framework as backend
- MongoDB as Database
- React as frontend
- JMS, Swagger UI, Eureka etc.,

The name of the Project undertaken is “**Online Hotel Management System**” which is a Web Application that lets entire operations of a hotel GRAND AJ INN. to perform online.

2.2 Scope of work

Backend implementation in spring framework(Java), Integration of backend with mongo database, frontend implementation in react, integrating frontend and backend through gateway, implementing add-ons to the project like payment gateway, swagger, rabbit Mq, hystrix, UI styling etc.,

The work has been allocated since March 3rd Week. Weekly Sprints are taken for various feature development and bug fix tasks. Daily Meeting is carried with Manager for status on work progress.

2.3 Importance and Applicability

Project helps in performing all operations of a hotel like checking-in & out guests, to check guest-log, adding employees, to view, update & delete them, adding inventories, rooms, to update and delete them etc.,

Through this project the technological stack that I learned for 3 months is applied. Practical approach made things clear and I have better understanding on learned things now.

Chapter III

Technological stack

The first part of the training process is Learning various technologies and doing Assignments in given time. The technologies and timeline given to complete them are as follows-

Topic	Duration	Start Date	End Date	Holiday
Git & Github	2	8-Jan	9-Jan	
MongoDB	4	11-Jan	14-Jan	
Core Java	10	15-Jan	27-Jan	26-Jan Holiday
JUnit & Testing Framework	2	28-Jan	29-Jan	
Spring Core	3	30-Jan	2-Feb	
Spring MVC	3	3-Feb	5-Feb	
Spring REST	3	6-Feb	9-Feb	
Spring Boot	3	10-Feb	12-Feb	
Spring Security	3	13-Feb	16-Feb	
HTML CSS JavaScript	7	17-Feb	24-Feb	
ES6 & TypeScript	3	25-Feb	27-Feb	
Angular	7	1-Mar	8-Mar	
React	6	9-Mar	15-Mar	
JMS RabbitMQ	3	16-Mar	18-Mar	
Spring Cloud	7	19-Mar	26-Mar	
Docker	4	27-Mar	1-Apr	29-Mar Holiday
Case Study	20	3-Apr	26-Apr	2-Apr Holiday
	90			

Figure 3.1 Training plan

The technologies to be learnt:

- Git & GitHub
- MongoDB
- Core Java
- Junit & Testing
- Spring
 - Spring Core
 - Spring MVC
 - Spring Boot
 - Spring Security
 - Spring Cloud

- Web Development
 - HTML
 - CSS
 - JS
 - TS & ES6
- Angular
- React
- Docker
- Rabbit Mq

let us look into an overview of each technology mentioned above-

3.1 Git & GitHub

Git

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non- linear workflows.



Git is primarily developed on Linux, although it also supports most major operating systems, including BSD, Solaris, macOS, and Windows.

The first Windows port of Git was primarily a Linux-emulation framework that hosts the Linux version. Installing Git under Windows creates a similarly named Program Files directory containing the MinGW port of the GNU Compiler Collection, Perl 5, msys2.0 (itself a fork of Cygwin, a Unix-like emulation environment for Windows) and various other Windows ports or emulations of Linux utilities and libraries. Currently native Windows builds of Git are distributed as 32- and 64-bit installers.

The JGit implementation of Git is a pure Java software library, designed to be embedded in any Java application. JGit is used in the Gerrit code-review tool and in EGit, a Git client for the Eclipse IDE.

GitHub

GitHub is a global company that provides hosting for software development version control using Git. It is a subsidiary of Microsoft, which acquired the company in 2018 for \$7.5 billion. It offers all the distributed version control and source code management functionality of Git as well as adding its own features.

It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.



3.2 Mongo DB

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License.

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but the name "NoSQL" was only coined in the early 21st century, triggered by the needs of Web 2.0 companies. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL systems are also sometimes called "Not only SQL" to emphasize that they may support SQL-like query languages or sit alongside SQL databases in polyglot persistent architectures.



Document Store



The central concept of a document store is the notion of a "document". While each document-oriented database implementation differs on the details of this definition, in general, they all assume that documents encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, and JSON as well as binary forms like BSON. Documents are addressed in the database via a unique key that represents that document. One of the other defining characteristics of a document-oriented database is that in addition to the key lookup performed by a key-value store, the database also offers an API or query language that retrieves documents based on their contents.

Different implementations offer different ways of organizing and/or grouping documents:

- Collections
- Tags
- Non-visible metadata
- Directory hierarchies

Compared to relational databases, for example, collections could be considered analogous to tables and documents analogous to records.

Graph

This kind of database is designed for data whose relations are well represented as a graph consisting of elements interconnected with a finite number of relations between them. The type of data could be social relations, public transport links, road maps, network topologies, etc.

3.3 Core JAVA



Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine regardless of the underlying computer architecture. The syntax of Java is like C and C++, but it has fewer low-level facilities than either of them. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

The syntax of Java is largely influenced by C++. Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language. All code is written inside classes, and every data item is an object, apart from the primitive data types, (i.e., integers, floating-point numbers, boolean values, and characters), which are not objects for performance reasons. Java reuses some popular aspects of C++ (such as the printf method).

Unlike C++, Java does not support operator overloading or multiple inheritance for classes, though multiple inheritance is supported for interfaces.

Java uses comments like those of C++. There are three different styles of comments: a single line style marked with two slashes (`//`), a multiple line style opened with `/*` and closed with `*/`, and the Javadoc commenting style opened with `/**` and closed with `*/`. The Javadoc style of commenting allows the user to run the Javadoc executable to create documentation for the program and can be read by some integrated development environments (IDEs) such as Eclipse to allow developers to access documentation within the IDE.

Implementation

Oracle Corporation is the current owner of the official implementation of the Java SE platform, following their acquisition of Sun Microsystems on January 27, 2010. This implementation is based on the original implementation of Java by Sun. The Oracle implementation is available for Microsoft Windows (still works for XP, while only later versions are currently officially supported), macOS, Linux, and Solaris. Because Java lacks any formal standardization recognized by Ecma International, ISO/IEC, ANSI, or other third-party standards organization, the Oracle implementation is the de facto standard.

The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end users, and the Java Development Kit (JDK), which is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger. Oracle has also released GraalVM, a high-performance Java dynamic compiler and interpreter.

OpenJDK is another notable Java SE implementation that is licensed under the GNU GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, OpenJDK is the official Java reference implementation.

The goal of Java is to make all implementations of Java compatible. Historically, Sun's trademark license for usage of the Java brand insists that all implementations be compatible. This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their

own. Sun sued in 1997, and, in 2001, won a settlement of US\$20 million, as well as a court order enforcing the terms of the license from Sun. As a result, Microsoft no longer ships Java with Windows.

Platform-independent Java is essential to Java EE, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications.

Use outside the java platform

The Java programming language requires the presence of a software platform for compiled programs to be executed. Oracle supplies the Java platform for use with Java. The Android SDK is an alternative software platform, used primarily for developing Android applications with its own GUI system.

Android:

The Java language is a key pillar in Android, an open-source mobile operating system. Although Android, built on the Linux kernel, is written largely in C, the Android SDK uses the Java language as the basis for Android applications but does not use any of its standard GUI, SE, ME, or other established Java standards. The bytecode language supported by the Android SDK is incompatible with Java bytecode and runs on its own virtual machine, optimized for low-memory devices such as smartphones and tablet computers. Depending on the Android version, the bytecode is either interpreted by the Dalvik virtual machine or compiled into native code by the Android Runtime.

3.4 Spring

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an addition to, or even replacement for the Enterprise JavaBeans model. The Spring Framework is open source.

3.4.1 Spring Core

This part of the Spring covers all those technologies that are integral to the Spring

Framework. Foremost amongst these is the Spring Framework's Inversion of Control (IoC) container. A thorough treatment of the Spring Framework's IoC container is closely followed by comprehensive coverage of Spring's Aspect-Oriented Programming (AOP) technologies. The Spring Framework has its own AOP framework, which is conceptually easy to understand, and which successfully addresses the 80% sweet spot of AOP requirements in Java enterprise programming. Coverage of Spring's integration with AspectJ (currently the richest - in terms of features - and certainly most mature AOP implementation in the Java enterprise space) is also provided. Finally, the adoption of the test-driven-development (TDD) approach to software development is certainly advocated by the Spring team, and so coverage of Spring's support for integration testing is covered (alongside best practices for unit testing). The Spring team has found that the correct use of IoC certainly does make both unit and integration testing easier (in that the presence of setter methods and appropriate constructors on classes makes them easier to wire together in a test without having to set up service locator registries and suchlike) the chapter dedicated solely to testing will hopefully convince you of this as well.

3.4.2 Spring MVC



The Spring Framework features its own model–view–controller (MVC) web application framework, which was not originally planned. The Spring developers decided to write their own Web framework as a reaction to what they perceived as the poor design of the (then) popular Jakarta Struts Web framework, as well as deficiencies in other available frameworks. They felt there was insufficient separation between the presentation and request handling layers, and between the request handling layer and the model.

Like Struts, Spring MVC is a request-based framework. The framework defines strategy interfaces for all the responsibilities that must be handled by a modern request-based framework. The goal of each interface is to be simple and clear so that it is easy for Spring MVC users to write their own implementations if they so choose. MVC paves the way for cleaner front-end code.

3.4.3 Spring Rest

In a Web service a Web technology such as HTTP — originally designed for human-to-machine communication — is used for transferring machine-readable file formats such as

XML and JSON. In practice, a Web service commonly provides an object-oriented Web-based interface to a database server, utilized for example by another Web server, or by a mobile app, that provides a user interface to the end-user. Many organizations that provide data in formatted HTML pages will also provide that data on their server as XML or JSON, often through a Web service to allow syndication, for example, Wikipedia's Export. Another application offered to the end-user may be a mashup, where a Web server consumes several Web services at different machines and compiles the content into one user interface.

REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways.

When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular programming language to use because, despite its name, it is language-agnostic, as well as readable by both humans and machines.

Something else to keep in mind: Headers and parameters are also important in the HTTP methods of a RESTful API HTTP request, as they contain important identifier information as to the request's metadata, authorization, uniform resource identifier (URI), caching, cookies, and more. There are request headers and response headers, each with their own HTTP connection information and status codes.

For an API to be considered RESTful, it must conform to these criteria:

- A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.
- Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.
- Cacheable data that streamlines client-server interactions.
- A uniform interface between components so that information is transferred in a standard form. This requires that:
 - resources requested are identifiable and separate from the representations sent to the client.
 - resources can be manipulated by the client via the representation they receive because the representation contains enough information to do so.

- self-descriptive messages returned to the client have enough information to describe how the client should process it.
 - hypertext/hypermedia is available, meaning that after accessing a resource the client should be able to use hyperlinks to find all other currently available actions they can take.
- A layered system that organizes each type of server (those responsible for security, load-balancing, etc.) involved the retrieval of requested information into hierarchies, invisible to the client.
- Code-on-demand (optional): the ability to send executable code from the server to the client when requested, extending client functionality.

Though the REST API has these criteria to conform to, it is still considered easier to use than a prescribed protocol like SOAP (Simple Object Access Protocol), which has specific requirements like XML messaging, and built-in security and transaction compliance that make it slower and heavier.

3.4.4 Spring Security

Spring Security is a Java/Java EE framework that provides authentication, authorization, and other security features for enterprise applications. The project was started in late 2003 as 'Acegi Security' by Ben Alex, with it being publicly released under the Apache License in March 2004. Subsequently, Acegi was incorporated into the Spring portfolio as Spring Security, an official Spring sub-project. The first public release under the new name was Spring Security 2.0.0 in April 2008, with commercial support and training available from Spring Source.

Json Web Token (JWT)

JSON Web Token is an Internet standard for creating JSON-based access tokens that assert some number of claims. For example, a server could generate a token that has the claim "logged in as admin" and provide that to a client. The client could then use that token to prove that it is logged in as admin. The tokens are signed by one party's private key, so that both parties can verify that the token is legitimate. The tokens are designed to be compact, URL-safe, and usable especially in a web-browser single-sign-on context. JWT claims can be typically used to pass identity of authenticated users between an identity provider and a service provider, or any other type of claims as required by business processes.

In authentication, when the user successfully logs in using their credentials, a JSON Web Token will be returned and must be saved locally (typically in local or session storage, but cookies can also be used), instead of the traditional approach of creating a session in the server and returning a cookie.

Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the Authorization header using the Bearer schema. The content of the header might look like the following: **Authorization:** Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5b5CSpyHI

This is a stateless authentication mechanism as the user state is never saved in server memory. The server's protected routes will check for a valid JWT in the Authorization header, and if it is present, the user will be allowed to access protected resources. As JWTs are self-contained, all the necessary information is there, reducing the need to query the database multiple times.

3.4.5 Spring Cloud

Spring Cloud provides tools for developers to quickly build some of the common patterns in distributed systems (e.g., configuration management, service discovery, circuit breakers, intelligent routing, micro-proxy, control bus, one-time tokens, global locks, leadership election, distributed sessions, cluster state). Coordination of distributed systems leads to boiler plate patterns and using Spring Cloud developers can quickly stand-up services and applications that implement those patterns. They will work well in any distributed environment, including the developer's own laptop, bare metal data centers, and managed platforms such as Cloud Foundry.

Features

Spring Cloud focuses on providing good out of box experience for typical use cases and extensibility mechanism to cover others.

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- Global locks
- Distributed messaging

3.5 Web Development

3.5.1 HTML

Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets and scripting languages such as JavaScript.

HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most come in pairs like `<h1>` and `</h1>`, although some represent empty elements and so are unpaired, for example ``. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags).

3.5.2 HTML5

HTML5 is a software solution stack that defines the properties and behaviours of web page content by implementing a markup-based pattern to it. HTML5 was the fifth and last major version of HTML that is a World Wide Web Consortium recommendation. The current specification is known as the HTML Living Standard and is maintained by a consortium of the major browser vendors, the Web Hypertext Application Technology Working Group.



3.5.3 CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

Before CSS, nearly all presentational attributes of HTML documents were contained within the HTML markup. All font colors, background styles, element alignments, borders and sizes had to be explicitly described, often repeatedly, within the HTML. CSS lets authors move much of that information to another file, the style sheet, resulting in considerably simpler HTML.

For example, headings (h1 elements), sub-headings (h2), sub-sub-headings (h3), etc., are defined structurally using HTML. In print and on the screen, choice of font, size, color and emphasis for these elements is presentational.

Before, document authors who wanted to assign such typographic characteristics to, say, all h2 headings had to repeat HTML presentational mark-up for each occurrence of that heading type.

3.5.4 JS

JavaScript, often abbreviated as JS, is a high-level, just-in-time compiled, multi-paradigm programming language that conforms to the ECMAScript specification. JavaScript has curly- bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

3.5.5 TS

Type-Script is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. Type-Script is designed for development of large applications and trans-compiles to JavaScript. As Type-Script is a superset of JavaScript, existing JavaScript programs are also valid Type-Script programs. Type-Script may be used to develop JavaScript applications for both client-side and server-side execution.

3.5.6 ES6

ECMAScript is a scripting-language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262. It was created to standardize JavaScript to help foster multiple independent implementations. JavaScript has remained the best-known implementation of ECMAScript since the standard was first published, with other well-known implementations including JScript and ActionScript. ECMAScript is commonly used for client-side scripting on the World Wide Web, and it is increasingly being used for writing server applications and services using Node.js.

3.6 Angular

AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller and model–view architectures, along with components commonly used in rich Internet applications.

The AngularJS framework works by first reading the Hypertext Markup Language (HTML) page, which has an additional custom HTML attributes embedded into it. Angular

interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code or retrieved from static or dynamic JSON resources.

AngularJS is built on the belief that declarative programming should be used to create user interfaces and connect software components, while imperative programming is better suited to defining an application's business logic. The framework adapts and extends traditional HTML to present dynamic content through two-way databinding that allows for the automatic synchronization of models and views. As a result, AngularJS de-emphasizes explicit Document Object Model (DOM) manipulation with the goal of improving testability and performance.

AngularJS implements the MVC pattern to separate presentation, data, and logic components. Using dependency injection, Angular brings traditionally server-side services, such as view- dependent controllers, to client-side web applications. Consequently, much of the burden on the server can be reduced.



AngularJS sets out the paradigm of a digest cycle. This cycle can be considered a loop, during which AngularJS checks if there is any change to all the variables watched by all the \$scopes. If \$scope.myVar is defined in a controller and this variable was marked for watching, Angular will monitor the changes on myVar in each loop iteration.

This approach potentially leads to slow rendering when AngularJS checks on too many variables in the \$scope every cycle. Miško Hevery suggests keeping fewer than 2000 watchers on any page.

3.7 React

React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, react is only concerned with rendering data to the DOM and so creating React applications usually requires the use of additional libraries for state management, routing, and interaction with an API. Redux, React Router and axios are respective examples of such libraries.

Refs provide a way to access DOM nodes or React elements created in the render

method. In the typical React dataflow, props are the only way that parent components interact with their children. To modify a child, you re-render it with new props. However, there are a few cases where you need to imperatively modify a child outside of the typical dataflow. The child to be modified could be an instance of a React component, or it could be a DOM element. For both cases, react provides an escape hatch.

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation. This page introduces the idea of components. You can find a detailed component API reference [here](#). Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen. The simplest way to define a component is to write a

JavaScript function:

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```



3.8 Docker

Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well- defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines.

Docker can package an application and its dependencies in a virtual container that can run on any Linux server. This helps provide flexibility and portability enabling the application to be run in various locations, whether on-premises, in a public cloud, or in a private cloud. Docker uses the resource isolation features of the Linux kernel and a union-capable file system to allow containers to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines. Because Docker containers are lightweight, a single server or virtual machine can run several containers simultaneously.

The Linux kernel's support for namespaces mostly isolates an application's view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel's cgroups provide resource limiting for memory and CPU. Since version 0.9, Docker includes its own component (called "libcontainer") to directly use virtualization facilities provided by the Linux kernel, in addition to using abstracted virtualization interfaces via libvirt, LXC and systemd-nspawn.

3.9 Rabbit Mq

Rabbit MQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol, Message Queuing Telemetry Transport, and other protocols.

Message Queue

In computer science, message queues and mailboxes are software-engineering components used for inter-process communication, or for inter-thread communication within the same process. They use a queue for messaging – the passing of control or of content. Group communication systems provide similar kinds of functionality.

Message queues provide an asynchronous communications protocol, meaning that the sender and receiver of the message do not need to interact with the message queue at the same time. Messages placed onto the queue are stored until the recipient retrieves them. Message queues have implicit or explicit limits on the size of data that may be transmitted in a single message and the number of messages that may remain outstanding on the queue.



Chapter IV

The Online Hotel Management System

Synopsis:

The introducing software, Hotel Management System which is going to be implemented for Hotel will automate the major operations of the hotel. The End Users Are Owner, Manager and Receptionist. Owner can access to all system functionalities without any restrictions. Manager can access to all system functionalities with limited restrictions. Receptionist can only access to the Reservation management section. To keep restrictions for each End User levels HMS can create different Login functions.

Functionalities:

- Make Reservations
- Search Room availability
- Add Payment
- Issue Bills
- Manage Guest (Add, Update Guest)
- Manage Room Details (Add, Update, Delete)
- Manage Staff (Add, Update, Delete, View)
- Manage Inventory (Add, Edit, Delete)
- Manage Departments (Add, Edit, Delete)
- Set Rates

4.1 Project Flow

The project is built in bottom to top approach, the flow is explained in below image-

Backend:

Main Micro-services -4 (spring boot applications) - Owner Boot on 8081, Manager Boot on 8082, Guest Boot on 8083, JMS on 8085.

Eureka (discovery client) and Api gateway are spring boot applications running on ports 8761, 9191. Each Micro-service is connected with different repository to different collection of mongo data base.

Jwt authentication and authorization is done and uses a user repository pointing to a MySQL database on port 3306 to fetch user data. Additionally, an authentication node on port 3001 is connected for better authorization at frontend pointing to same MySQL database (3306). Swagger UI is integrated for the documentation of backend to design UI.

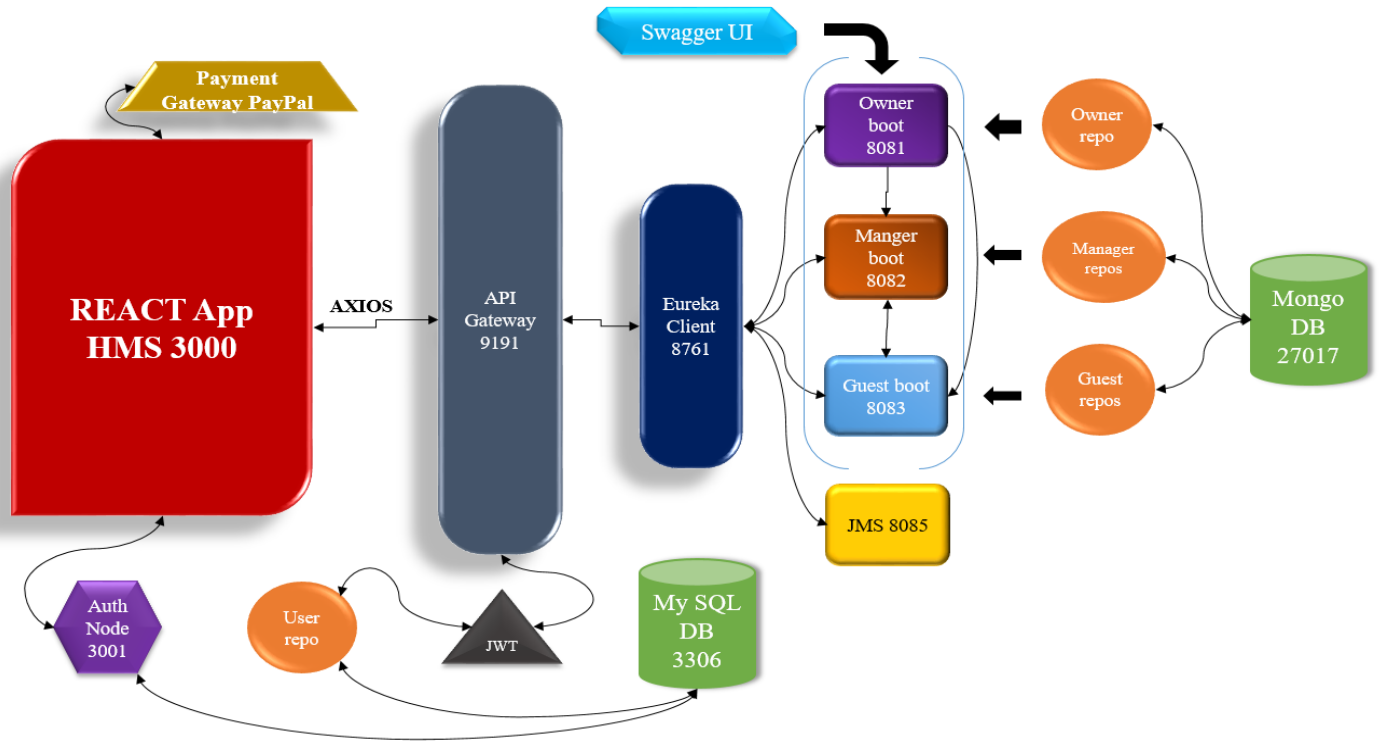


Figure 4.1 Project (block) Flow

Frontend:

React app **HMS** running on port 3000 integrates with backend through axios service Api calls. Payment gateway PAYPAL is attached to HMS for payment processing.

4.2 Backend Implementation

In this part of the project, Middleware and the Backend of the project is Implemented using Spring Boot in IntelliJ. All the Micro-Services are implemented in their respective individual spring boot application and then integrated with Rest Api (Application Programming Interface), Api gateway and discovery client. Each spring boot application has different configuration packages to perform certain operations coded inside.

They are Main class, Controller, Model, Service, Repository, exception, util, filter, config, resources, tests. We can see the package distribution in figure 4.2, the java class files are distributed among different packages. Let us look into each package in detail.

Package Description:

Controller: Rest controller- create and handles all restful web service requests.

Model: A model class is typically used to "model" the data in application. For example, we could write a Model class that mirrors a database or a JSON.

Service: service layer contains business logic for some of the controllers.

Repository: bares as connection in between spring application and database.

Exception: exception handlers, exception models are present.

Config: have main security configuration under which all Jwt handlers are present.

Util: generate Jwt tokens and sessions.

Filter: filter class used to act as filter to Jwt requests.

Resources: have application properties and yaml files.

Tests: Contain all integration tests of all packages.

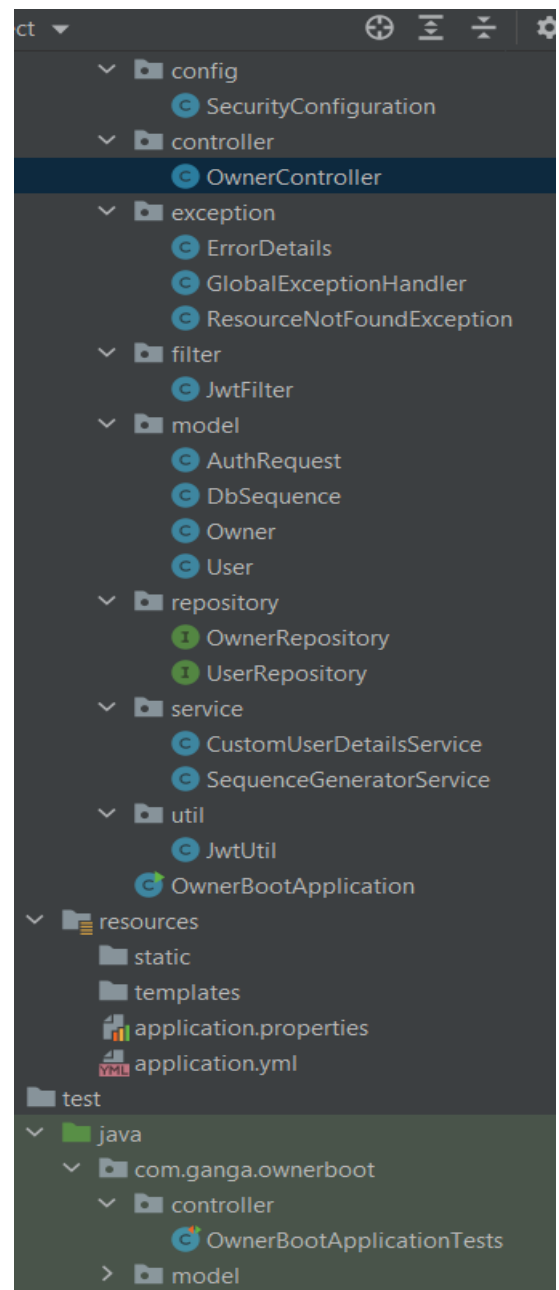


Figure 4.2 package description

Let us look into each of packages in detail for a single spring boot application owner boot.

Owner Controller:

As it is a rest controller, it contains different request mappings (POST, PUT, GET, DELETE). Each restful web request has a different Api call as shown /addManager, /editManager etc., The method addManager contains a sequence generator call to auto-generate id, the insert operation is done to mongo db.

```

54  /*Handles the HTTP POST request matched with given URI expression*/
55  @PostMapping("/addManager")
56
57  /*Method to create a new manager*/
58
59  /*@Request body enables automatic deserialization of the inbound HttpRequest body onto a Java object*/
60  @
61  public String addManager(@RequestBody Owner manager){
62
63      /*Setting Id by sequence generator*/
64      manager.setManagerId(service.getSequenceNum(Owner.sequenceName));
65
66      /*inserting manager model variables into collection*/
67      repository.insert(manager);
68
69      /*displays Manager added*/
70      return "Added Manager with id: "+manager.getManagerId();
71  }
72
73  /*Handles the HTTP PUT request matched with given URI expression*/
74  @PutMapping("/editManager/{managerId}")
75
76  /*Method to update a specified manager provided with manager Id*/
77  @
78  public ResponseEntity<Owner> updateManager(@PathVariable int managerId,
79      @Validated @RequestBody Owner managerDetails) throws ResourceNotFoundException {
80
81      /*exception handling if id not found*/
82      Owner manager = repository.findById(managerId)
83          .orElseThrow(() -> new ResourceNotFoundException("Manager not found for this id :: " + managerId));
84
85      /*if id is found updating old with new */
86      manager.setManagerName(managerDetails.getManagerName());
87      manager.setDepartmentName(managerDetails.getDepartmentName());
88      manager.setManagerMailId(managerDetails.getManagerMailId());
89      manager.setManagerContact(managerDetails.getManagerContact());
90      manager.setManagerSalary(managerDetails.getManagerSalary());

```

Figure 4.3 owner controller

Owner Model: This contains all model attributes, constructors, getters and setters of all variables.

```

import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Transient;
import org.springframework.data.mongodb.core.mapping.Document;

/*getters, setters, all arguments and no argument constructors by annotations*/
@Data
@AllArgsConstructor
@NoArgsConstructor

/*@document- to specify custom property values*/
/*Defining collection name of mongodb to the model class*/
@Document(collection = "Owner")
public class Owner {

    /*This helps ignoring the field by not mapping it to the database*/
    @Transient
    public static final String sequenceName="userSequence";

    /*Providing all variables required */
    /*Setting up the below variable as primary key in the collection*/
    @Id
    private int managerId;
    private String departmentName;
    private String managerName;
    private String managerContact;
    private String managerMailId;
    private String managerGender;
    private int managerSalary;
    private int staffSalary;
}

```

Figure 4.4 owner model

Owner Repository:

It extends mongo repository to configure the model class to mongo db.

```
package com.ganga.ownerboot.repository;

import com.ganga.ownerboot.model.Owner;
import org.springframework.data.mongodb.repository.MongoRepository;

/*MongoRepository T=> model name ID=> Primary key*/
public interface OwnerRepository extends MongoRepository<Owner, Integer> {
}
```

Figure 4.5 owner repository

Exception Handler:

Exception handler handles the exceptions when called anywhere.

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;

/*handle exceptions across the whole application in one global handling component*/
@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<?> resourceNotFoundException(ResourceNotFoundException ex, WebRequest request) {
        ErrorDetails errorDetails = new ErrorDetails(ex.getMessage(), request.getDescription(false));
        return new ResponseEntity<>(errorDetails, HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<?> globalExceptionHandler(Exception ex, WebRequest request) {
        ErrorDetails errorDetails = new ErrorDetails(ex.getMessage(), request.getDescription(false));
        return new ResponseEntity<>(errorDetails, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Figure 4.6 exception handler

Security Configuration:

Consists of methods that helps in configuring certain Api calls restricted and allowed to only certain roles and authorities (login users). Authentication manager helps in generating tokens and look as they are valid for particular session.

```

@Autowired
private CustomUserDetailsService userDetailsService;

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(encoder());
}

@Bean
public PasswordEncoder encoder() { return new BCryptPasswordEncoder(); }

@Bean(name = BeanIds.AUTHENTICATION_MANAGER)
@Override
public AuthenticationManager authenticationManagerBean() throws Exception {
    return super.authenticationManagerBean();
}

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.cors();
    http.csrf().disable().authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
        .antMatchers( ...antPatterns: "/owner/addManager",
            "/owner/editManager/**", "/owner/findManagers", "/owner/findManager/**", "/owner/deleteManager/**")
        .hasAnyRole( ...roles: "owner").antMatchers( ...antPatterns: "/owner/authenticate").permitAll().anyRequest()
        .authenticated().and().exceptionHandling().and().sessionManagement() SessionManagementConfigurer<HttpSecurity>
        .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
}
}

```

Figure 4.7 Security configuration

The configure file need certain extended services like util, filter, custom service etc., are listed below- Jwt Util,

```

public Date extractExpiration(String token) { return extractClaim(token, Claims::getExpiration); }

public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
    final Claims claims = extractAllClaims(token);
    return claimsResolver.apply(claims);
}

private Claims extractAllClaims(String token) {
    return Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody();
}

private Boolean isTokenExpired(String token) { return extractExpiration(token).before(new Date()); }

public String generateToken(String username) {
    Map<String, Object> claims = new HashMap<>();
    return createToken(claims, username);
}

private String createToken(Map<String, Object> claims, String subject) {

    return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
        .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 10))
        .signWith(SignatureAlgorithm.HS256, secret).compact();
}

public Boolean validateToken(String token, UserDetails userDetails) {
    final String username = extractUsername(token);
    return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
}

```

Figure 4.8 Jwt util class

Jwt Filter

```
@Override
protected void doFilterInternal(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, FilterChain filterChain) throws ServletException, IOException {

    String authorizationHeader = httpServletRequest.getHeader("Authorization");

    String token = null;
    String username = null;

    if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {
        token = authorizationHeader.substring(7);
        username = jwtUtil.extractUsername(token);
    }

    if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

        UserDetails userDetails = service.loadUserByUsername(username);

        if (jwtUtil.validateToken(token, userDetails)) {

            UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken =
                new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
            usernamePasswordAuthenticationToken
                .setDetails(new WebAuthenticationDetailsSource().buildDetails(httpServletRequest));
            SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);
        }
    }
    filterChain.doFilter(httpServletRequest, httpServletResponse);
}
```

Figure 4.9 Jwt Filter

Custom service

```
@Service
public class CustomUserDetailsService implements UserDetailsService {

    @Autowired
    private UserRepository repository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        List<SimpleGrantedAuthority> roles = null;

        User user = repository.findByUsername(username);

        if (user != null) {
            roles = Arrays.asList(new SimpleGrantedAuthority("ROLE_" + user.getRole()));
            return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(), roles);
        }
        throw new UsernameNotFoundException("User not found with the name " + username);
    }
}
```

Figure 4.10 customUserDetails service

Main Class

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@SpringBootApplication
@EnableEurekaClient
@EnableSwagger2
public class OwnerBootApplication {

    public static void main(String[] args) { SpringApplication.run(OwnerBootApplication.class, args); }

}
```

Figure 4.11 Main Class

Resources: application yaml file

Server port, name specifications and mongo db, MySQL configuration are present.

```
server:
  port: 8081
spring:
  application:
    name: Owner-boot
  data:
    mongodb:
      host: localhost
      port: 27017
      database: Hotel

  datasource:
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/mydb
    username: root
    password: Amma@2000
  jpa:
    hibernate.ddl-auto: update
    generate-ddl: true
    show-sql: true
```

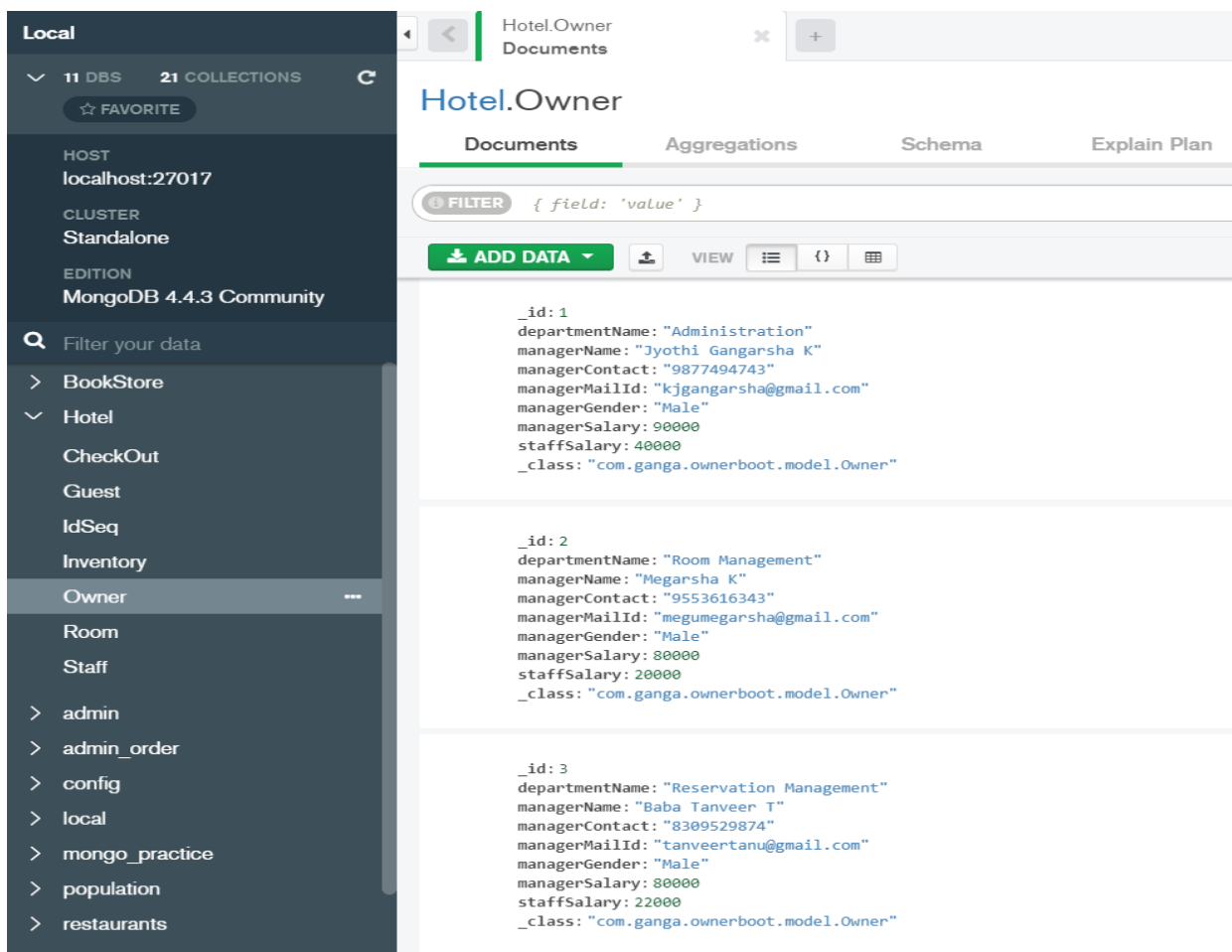
Figure 4.12 application.yaml file

Tests: Controller layer testing

```
@Test
public void testCreateEmployee1() {
    Owner employee = new Owner();
    employee.setDepartmentName("eadmin");
    employee.setManagerName("eadmin");
    employee.setManagerId(2);
    employee.setManagerContact("9845785478");
    employee.setStaffSalary(9521);
    employee.setManagerMailId("Gang@gmail.com");
    employee.setManagerSalary(85514);
    employee.setManagerGender("male");
    ResponseEntity<Owner> postResponse = restTemplate.postForEntity(url.getRootUrl() + "/addManager", employee, Owner.class);
    assertNotNull(postResponse);
    assertNotNull(postResponse.getBody());
}
```

Figure 4.13 Controller testing

Mongo DB: Data is stored in json format in respective collection of the database.



The screenshot shows the MongoDB Compass interface. On the left, the 'Local' sidebar displays the database 'Hotel' and its collections: BookStore, Hotel, CheckOut, Guest, IdSeq, Inventory, Owner (selected), Room, Staff, admin, admin_order, config, local, mongo_practice, population, and restaurants. The main panel shows the 'Hotel.Owner' collection with three documents. Each document contains fields for _id, departmentName, managerName, managerContact, managerMailId, managerGender, managerSalary, staffSalary, and _class.

_id	departmentName	managerName	managerContact	managerMailId	managerGender	managerSalary	staffSalary	_class
1	Administration	Jyothi Gangarsha K	9877494743	kjgangerasha@gmail.com	Male	90000	40000	com.ganga.ownerboot.model.Owner
2	Room Management	Megarsha K	9553616343	megumegarsha@gmail.com	Male	80000	20000	com.ganga.ownerboot.model.Owner
3	Reservation Management	Baba Tanveer T	8309529874	tanveertanu@gmail.com	Male	80000	22000	com.ganga.ownerboot.model.Owner

Figure 4.14 mongo db compass

Postman:

Testing Api calls through postman

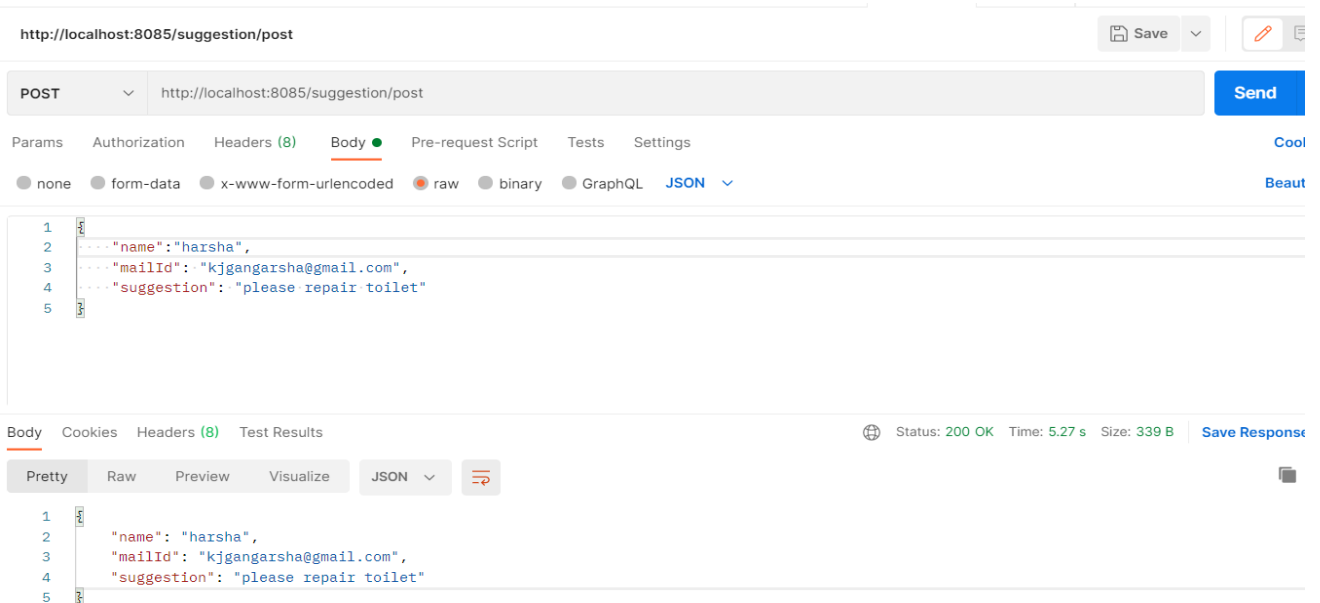


Figure 4.15 postman

MySQL workbench: user credentials and role are stored into database.

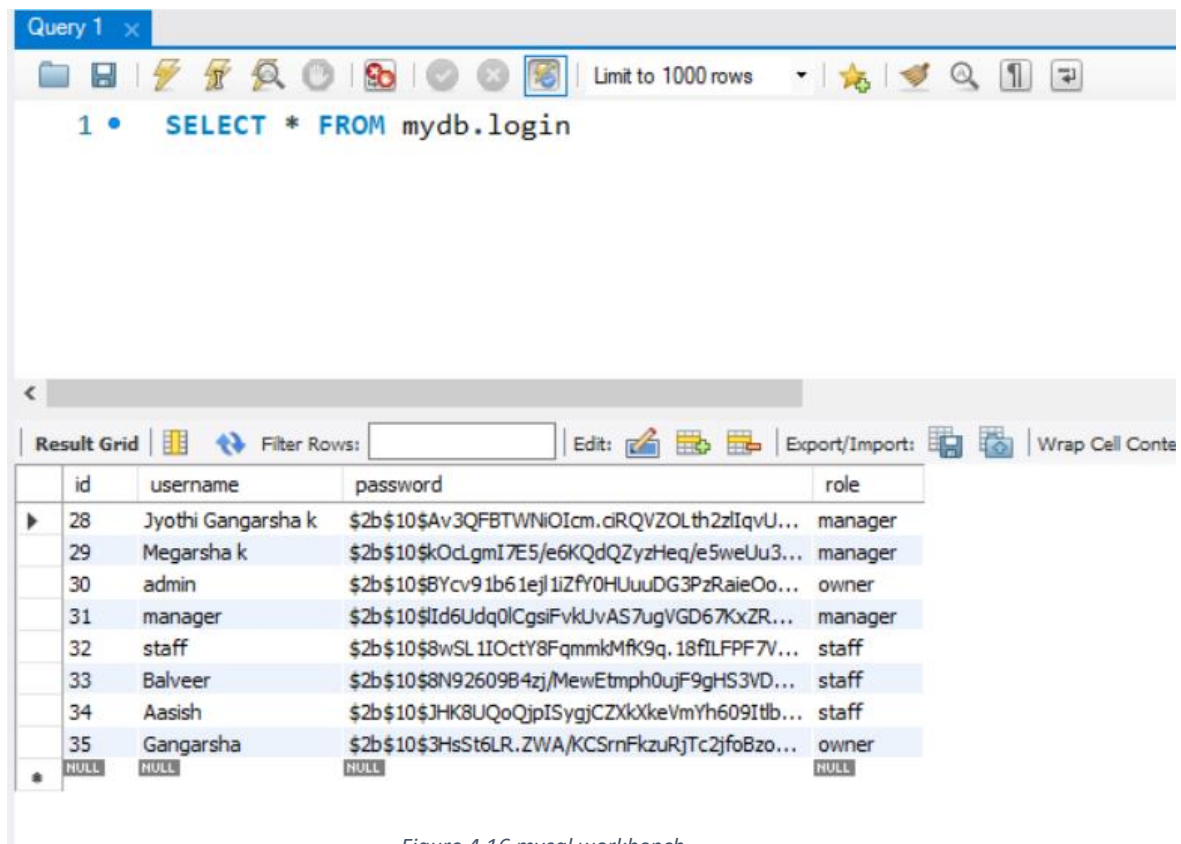


Figure 4.16 mysql workbench

4.3 Frontend Implementation

The whole frontend is done on react app HMS, it has several components like listed.

Every particular component has different js files to perform respective operations.

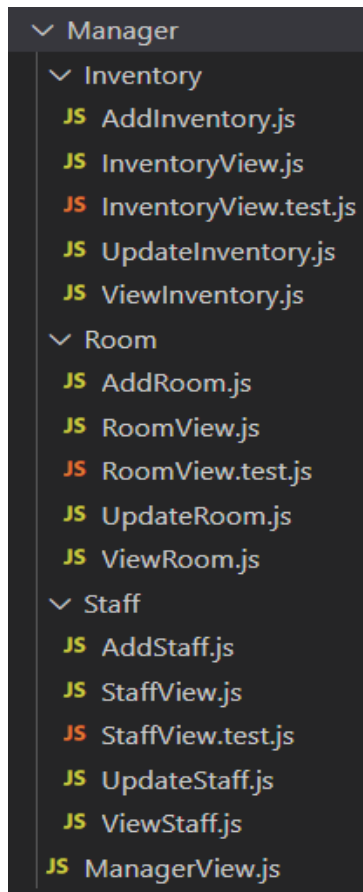


Figure 4.18 manager ops

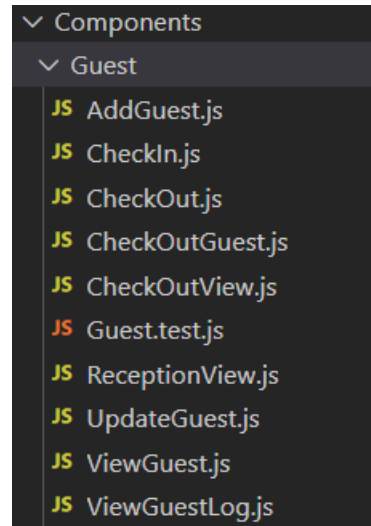


Figure 4.19 Guest ops

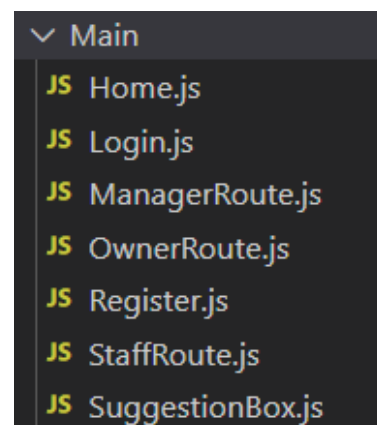


Figure 4.20 other ops files

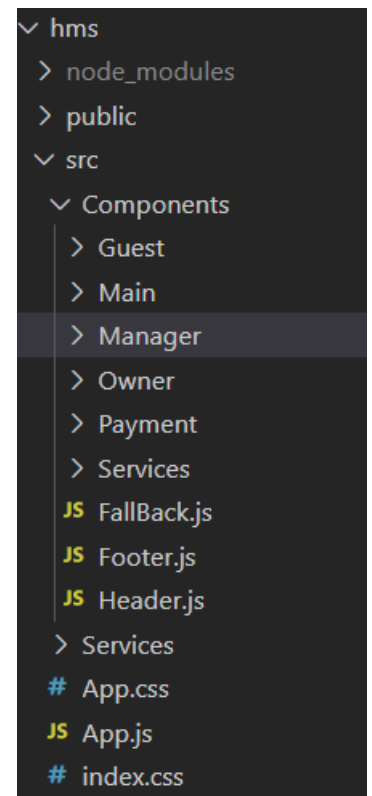


Figure 4.17 react components

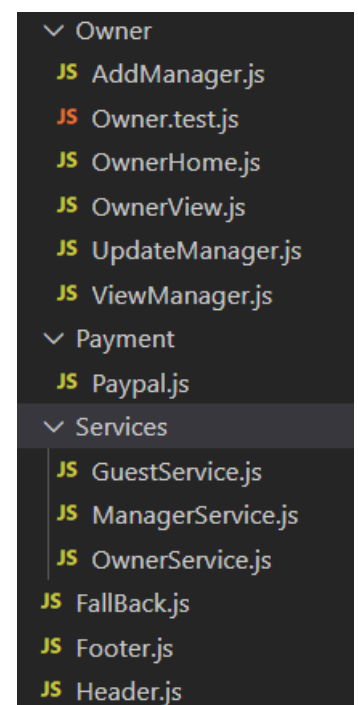


Figure 4.21 Owner, paypal, service files

Each component is exported and imported in app.js file. Here in app.js file routing is done to specifically call a component with a route. Here the route is protected and we can call the route to perform operation of certain component.

App.js

```
<Switch>
  { /* Global routes */ }
  <Route path="/" exact component={Home} />
  <Route path="/home" exact component={Home} />
  <Route path="/login" component={Login} />
  <StaffRoute path="/paypal" component={Paypal} />
  <Route path="/suggest" component={SuggestionBox} />

  { /* Owner routes */ }
  <OwnerRoute path="/owner" component={OwnerHome} />
  <OwnerRoute path="/ownerView" component={OwnerView} />
  <OwnerRoute path="/addUser" component={Register} />

  <OwnerRoute path="/addManager" component={AddManager} />
  <OwnerRoute
    path="/editManager/:managerId"
    component={UpdateManager}
  />
  <OwnerRoute
    path="/viewManager/:managerId"
    component={ViewManager}
  />

  { /* Manager routes */ }
  <ManagerRoute path="/manager" component={ManagerView} />

  { /* Staff Based */ }
  <ManagerRoute path="/staffView" component={StaffView} />
  <ManagerRoute path="/addStaff" component={AddStaff} />
  <ManagerRoute path="/editStaff/:staffId" component={UpdateStaff} />
  <ManagerRoute path="/viewStaff/:staffId" component={ViewStaff} />
```

Figure 4.22 App.js file

Here the routes are specified as owner route, manager etc., only those authorities can access the route paths. The component rendering is done here.


```

const EMPLOYEE_API_GETAVAILABLEROOMS_URL =
  "http://localhost:9191/guest/availableRooms";
const EMPLOYEE_API_GETROOM_URL = "http://localhost:9191/guest/getRoom";
const EMPLOYEE_API_GETROOMBYNUMBER_URL =
  "http://localhost:9191/guest/getRoomByNumber";
const EMPLOYEE_API_ADDGUEST_URL = "http://localhost:9191/guest/addGuest";
const EMPLOYEE_API_ADDCHECKOUT_URL = "http://localhost:9191/guest/checkOut";
const EMPLOYEE_API_VIEWGUEST_URL = "http://localhost:9191/guest/findGuest";
const EMPLOYEE_API_EDITGUEST_URL = "http://localhost:9191/guest/editGuest";
const EMPLOYEE_API_DELETEGUEST_URL = "http://localhost:9191/guest/deleteGuest";

const generateHeader = (backEndToken) => {
  let header = {
    headers: {
      Authorization: "Bearer " + backEndToken,
    },
  };
  return header;
};

class GuestService {
  /* methods wrt axios calls(APIs) */

  getGuests(backEndToken) {
    return axios.get(EMPLOYEE_API_FINDALL_URL, generateHeader(backEndToken));
  }
}

```

Figure 4.23 Guest Service.js

```

changeManagerGenderHandler = (event) => {
  this.setState({ managerGender: event.target.value });
};

/* save employee method */
saveEmployee = (e) => {
  e.preventDefault();
  if (this.validator.allValid()) {
    let employee = {
      departmentName: this.state.departmentName,
      managerName: this.state.managerName,
      managerSalary: this.state.managerSalary,
      staffSalary: this.state.staffSalary,
      managerContact: this.state.managerContact,
      managerMailId: this.state.managerMailId,
      managerGender: this.state.managerGender,
    };

    OwnerService.createEmployee(employee, this.state.backEndToken).then(
      (res) => {
        this.props.history.push("/ownerView");
        alert("New Manager Added");
        window.location.reload();
      }
    );
  } else {
    this.validator.showMessages();
    // rerender to show messages for the first time
    this.forceUpdate();
  }
}

```

Figure 4.24 addManager.js

Home Page

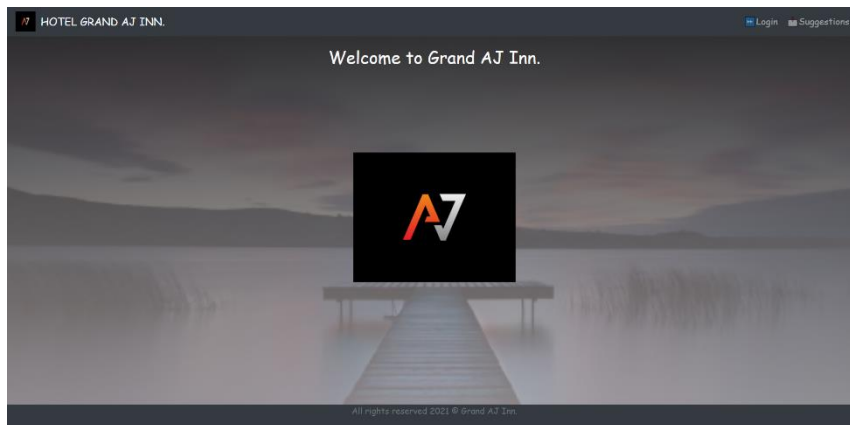


Figure 4.25 Home page

Login Page

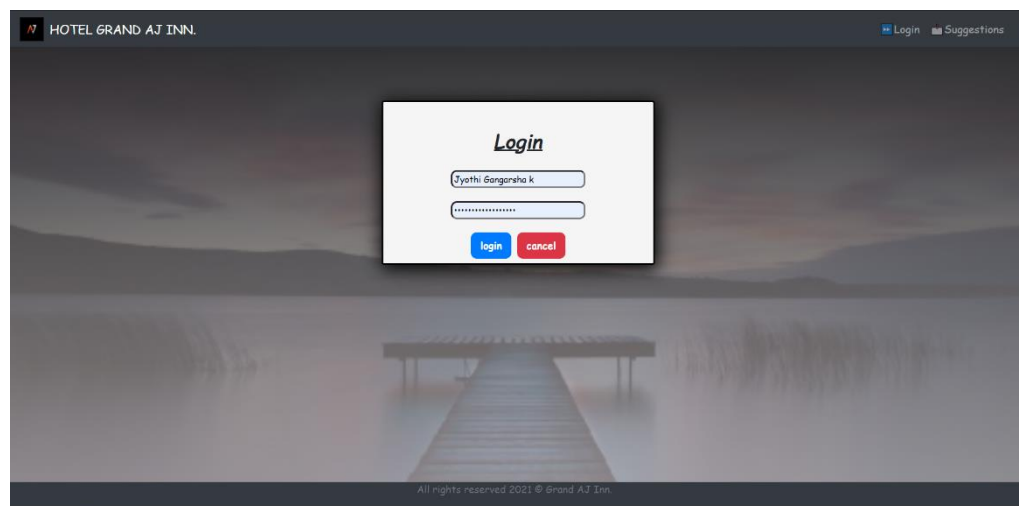


Figure 4.26 Login Page

Owner page

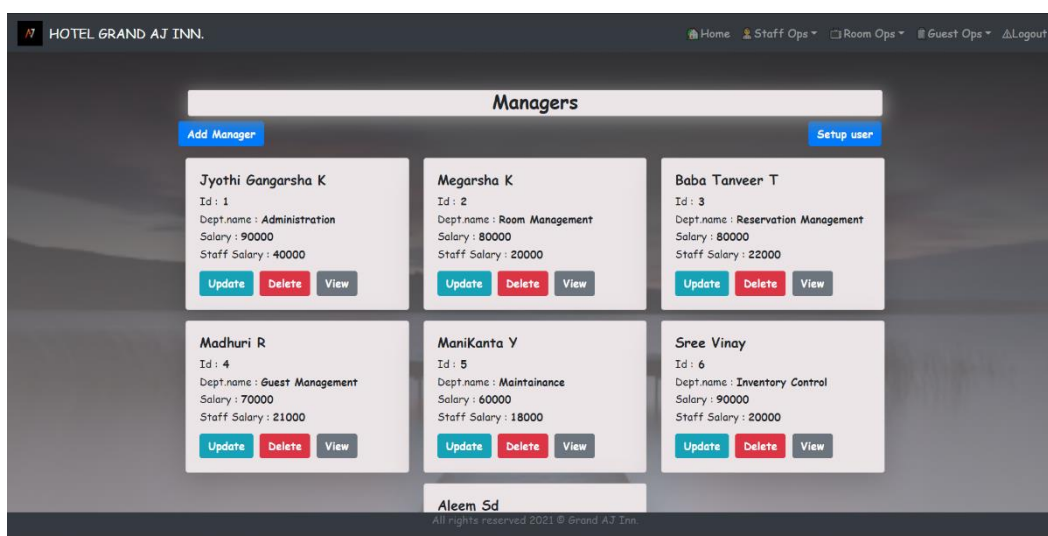


Figure 4.27 owner page

Hotel GRAND AJ INN. Home Staff Ops Room Ops Guest Ops Logout

Add Employee

Manager Name: Manager Salary:

Department Name:

Manager Contact:
The contact must be a valid phone number.

Manager Mail Id:

Manager Gender:

Staff Salary:

All rights reserved 2021 © Grand AJ Inn.

Figure 4.28 Add Manager form

Add Manager Form

Employees page

Hotel GRAND AJ INN. Home staff Rooms Inventory Guest Ops Logout

Employees

Balveer Id : 1 Dept.name : Administration Contact : 9874563210 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	Thahir T Id : 2 Dept.name : Administration Contact : 7412589630 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	Aasish ch Id : 3 Dept.name : Room Management Contact : 9874521741 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>
Rahul s Id : 4 Dept.name : Room Management Contact : 8745693111 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	Swetha Pv Id : 5 Dept.name : Reservation Management Contact : 8745963210 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	Sunitha k Id : 6 Dept.name : Reservation Management Contact : 9877452103 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>
Bhargav phani s Id : 7 Dept.name : Guest Management <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>		

All rights reserved 2021 © Grand AJ Inn.

Figure 4.29 Employees page

Rooms Page

Hotel GRAND AJ INN. Home staff Rooms Inventory Guest Ops Logout

Rooms List

101 Type : Deluxe Suite Price : 800 Status : Booked <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	102 Type : Deluxe Suite Price : 800 Status : Empty <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	103 Type : Deluxe Suite Price : 800 Status : Empty <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>
104 Type : Deluxe Suite Price : 800 Status : Empty <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	105 Type : Deluxe Suite Price : 800 Status : Empty <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	201 Type : Luxury Suite Price : 2000 Status : Booked <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>
202 Type : Luxury Suite Price : 2000 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	203 Type : Luxury Suite Price : 2000 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	301 Type : VVIP Suite Price : 5000 <input type="button" value="Update"/> <input type="button" value="Delete"/> <input type="button" value="View"/>

All rights reserved 2021 © Grand AJ Inn.

Figure 4.30 Rooms page

Room Availability

Room Number	Type	Price	Status	Book
102	Deluxe Suite	800	Empty	Check-In
103	Deluxe Suite	800	Empty	Check-In
104	Deluxe Suite	800	Empty	Check-In
105	Deluxe Suite	800	Empty	Check-In
202	Luxury Suite	2000	Empty	Check-In
203	Luxury Suite	2000	Empty	Check-In
302	VVIP Suite	5000	Empty	Check-In

All rights reserved 2021 © Grand AJ Inn.

Figure 4.31 Rooms Availability

Checked-In guests

Guest Name	Room No.	ID	Check-In Date	Contact	Guest count
raju	101	40	5/18/2021	9050275262	4
vinnu	201	41	5/18/2021	9874563221	2
Ramesh	301	42	5/18/2021	9441305285	3

Checked_In Guests :3

All rights reserved 2021 © Grand AJ Inn.

Figure 4.32 checked-in guests

Guest log

check-Out Id	Guest Id	Room Number	Guest Name	Guest Check-In Date	Check-Out Date	Guest Gender	Guest Contact	Days-in	Payment Method	Bill Amount
1	1	101	Paluri Gowtham	4/20/2021	4/24/2021	Male	9877494745	4	Debit Card	3200
2	2	102	Udhav k	4/22/2021	4/24/2021	Male	8341669785	2	Credit Card	1600
3	3	103	sasirekha pv	4/21/2021	4/24/2021	Female	7346985241	3	BHIM UPI	2400
4	4	201	Harsha Vittal	4/23/2021	4/24/2021	Male	7854963201	1	CASH	2000
5	5	202	pv sindhu	4/23/2021	4/24/2021	Male	9856327410	1	Credit Card	2000
6	6	301	Nanendra singh	4/22/2021	4/24/2021	Male	9876541230	2	Debit Card	10000
7	7	201	subbalakshmi k	4/23/2021	4/24/2021	Male	8521479630	1	Credit Card	2000

All rights reserved 2021 © Grand AJ Inn.

Figure 4.33 Guest log

Check-out

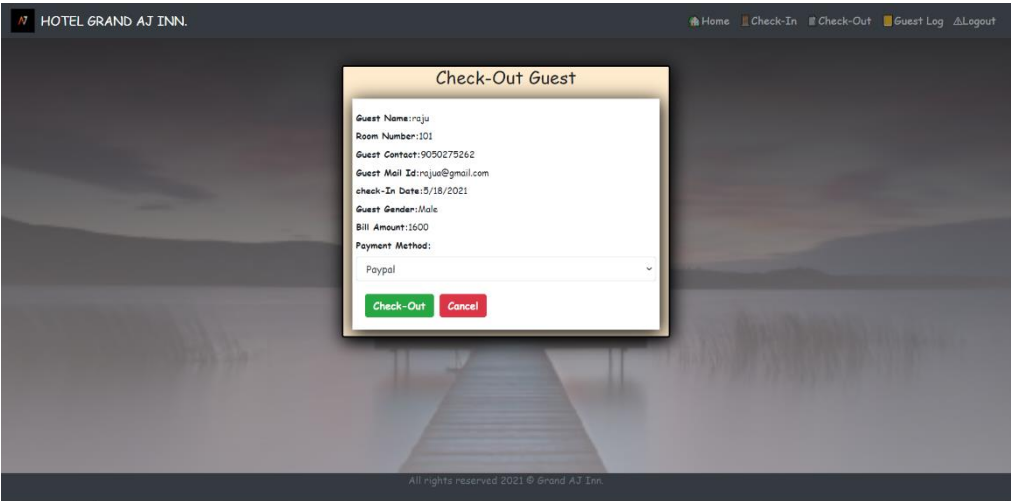


Figure 4.34 Checkout

PayPal

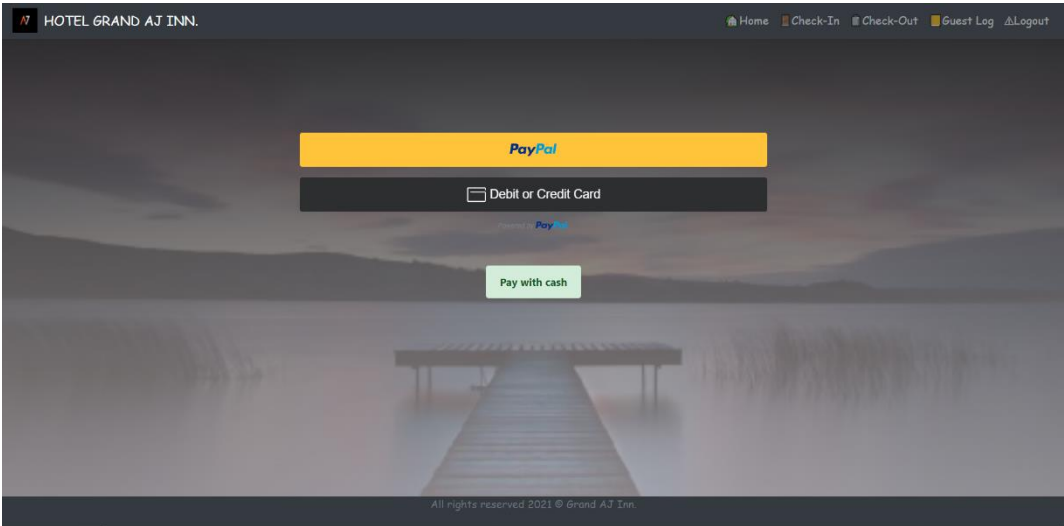


Figure 4.35 PayPal

Payment

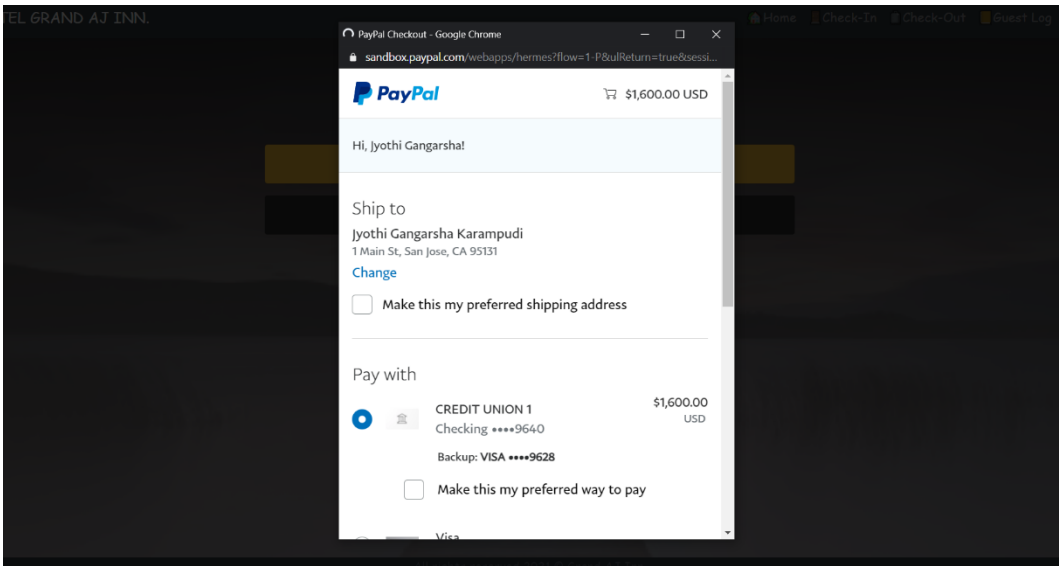


Figure 4.35 Payment

Payment confirmation

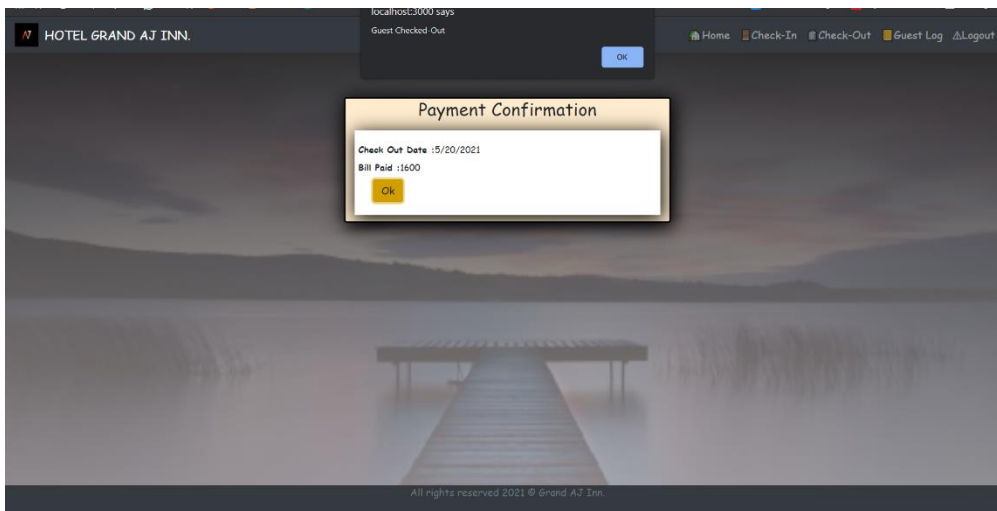


Figure 4.36 Payment confirmation

Suggestion Box

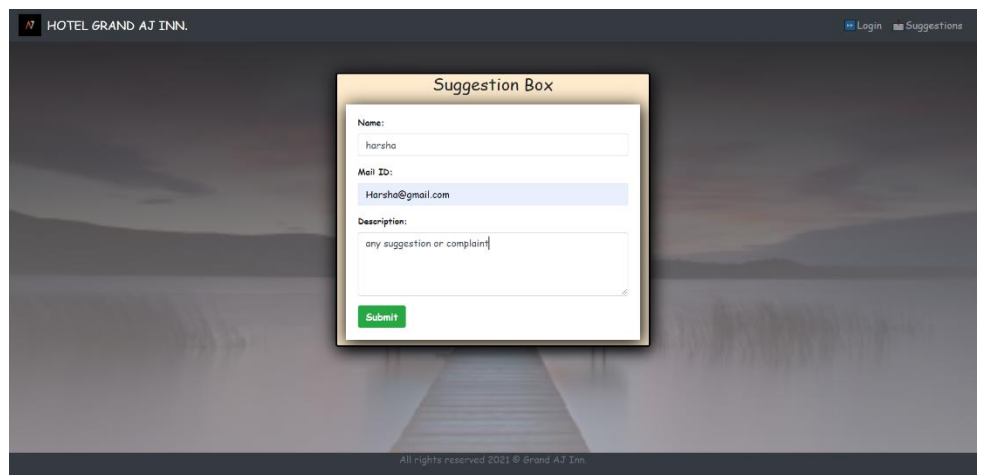


Figure 4.37 Suggestion box

UI Testing

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PASS  src/Components/Manager/Staff/StaffView.test.js
PASS  src/Components/Owner/Owner.test.js
PASS  src/Components/Manager/Room/RoomView.test.js
PASS  src/Components/Manager/Inventory/InventoryView.test.js
PASS  src/Components/Guest/Guest.test.js

Test Suites: 5 passed, 5 total
Tests:       66 passed, 66 total
Snapshots:   0 total
Time:        10.356 s
Ran all test suites.
```

Figure 4.38 UI testing

Backend testing

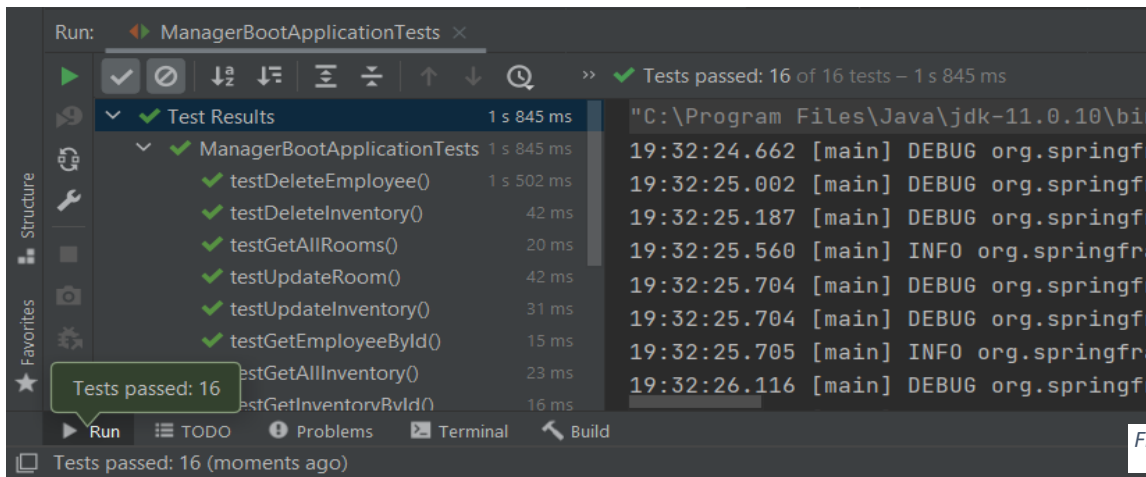


Figure 4.39 Backend testing

Swagger UI

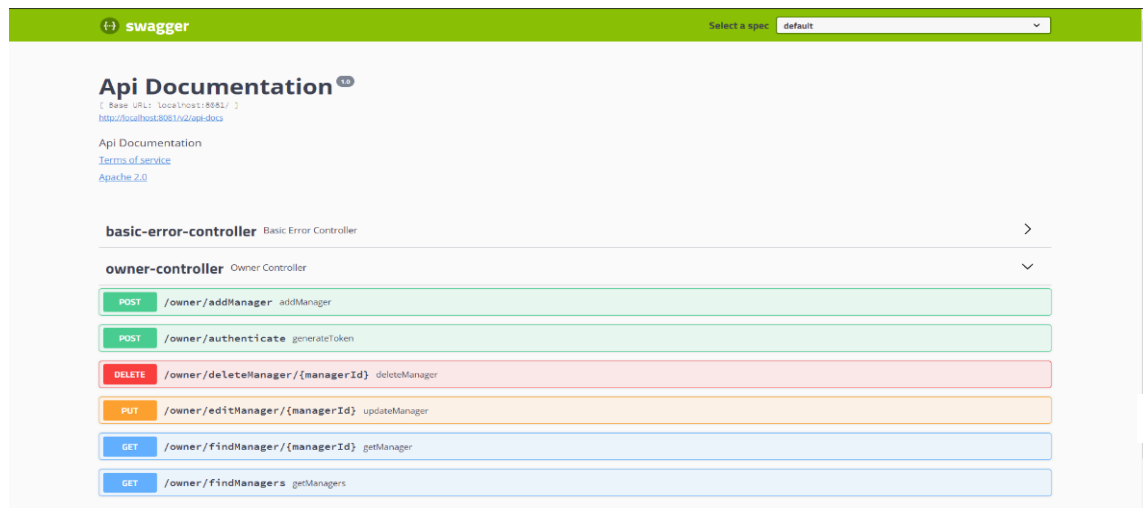


Figure 4.40 Swagger UI

Eureka client

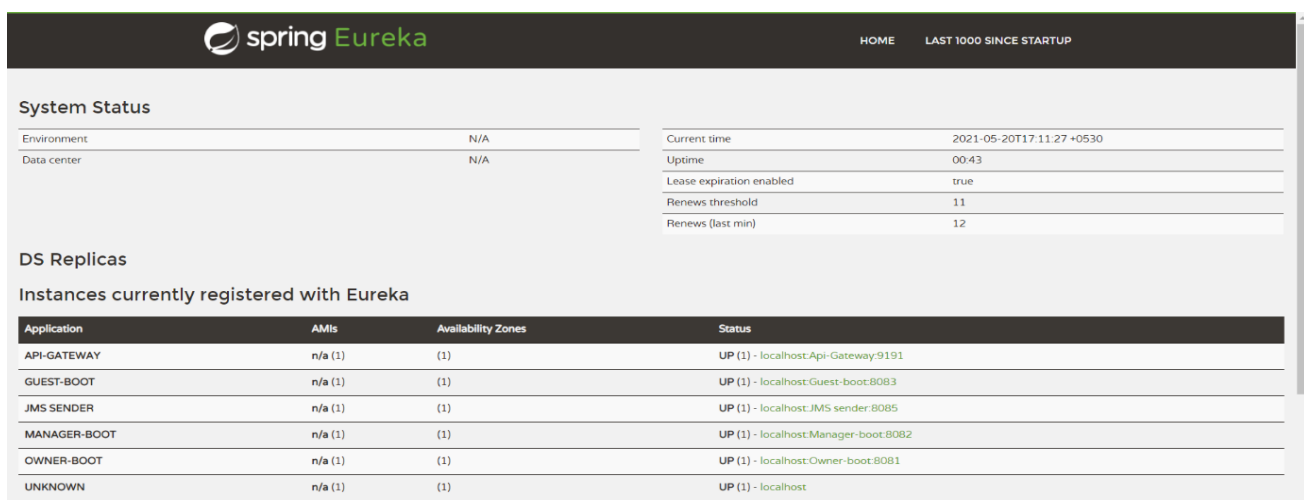


Figure 4.41 Eureka client

Chapter V

Conclusion

The ADAPT training is a self-paced learning module where a student transforms himself with some resources provided and true guidance from mentors and colleagues. This training helped me to excel in a huge bundle of technological stack and to implement them practically. From scratch to a Java full stack developed web application, although faced many challenges but I walked through my first step of achievement in the career path.

References

[FY 2019 results – Capgemini Investors EN](#). Capgemini.

[Capgemini will implement a squeeze-out on Altran shares – Capgemini Worldwide](#). Capgemini.

[Company Profile & Key Figures – Capgemini Worldwide](#) January 2015

[Overview – Capgemini Worldwide](#) 30 May 2018

<http://www.capgemini.com/about/capgemini/history/overview/>

<http://en.wikipedia.org/wiki/Git>

<http://en.wikipedia.org/wiki/GitHub>

<http://en.wikipedia.org/wiki/MongoDb>

<http://en.wikipedia.org/wiki/Java>

<http://en.wikipedia.org/wiki/Spring>

<http://en.wikipedia.org/wiki/HTML>

[http://en.wikipedia.org/wiki/Cascading Style Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets)

<http://en.wikipedia.org/wiki/ECMAScript>

<http://en.wikipedia.org/wiki/JavaScript>

<http://en.wikipedia.org/wiki/AngularJS>

<http://en.wikipedia.org/wiki/React>

<http://en.wikipedia.org/wiki/Rabbitmq>

[http://en.wikipedia.org/wiki/Docker \(software\)](http://en.wikipedia.org/wiki/Docker_(software))

---THANK YOU---