

# **BOOK WITH ME HOTEL BOOKING APPLICATION USING ANGULAR AND NODE.JS**

Project report submitted in fulfilment of the requirement for the degree of

Bachelor of Technology in

Computer Science and Engineering by

Akash Marwah (171348)

Under the supervision of

Mr. Bobin Sondhi to

Dr. Rajinder Sandhu



Department of Computer Science & Engineering and Information  
Technology

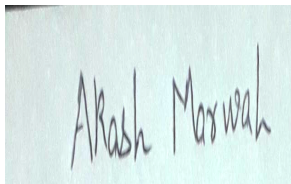
Jaypee University of Information Technology, Wagnaghat,  
Solan-173234, Himachal Pradesh, India

# Certificate

## Candidate's Declaration


I hereby declare that the work presented in this report entitled BOOK WITH ME HOTEL BOOKING APPLICATION USING ANGULAR and NODE.JS in fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted in the department of Computer Science & Engineering and Information Technology, Waknaghat is an authentic record of our own work carried out over a period from February 2020 to May 2020 under the supervision of Mr. Bobin Sondhi.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

A photograph of a handwritten signature in black ink on a light-colored background. The signature reads "Akash Marwah".

Akash Marwah  
171348

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

A photograph of a handwritten signature in blue ink on a light-colored background. The signature reads "Sondhi".

Mr. Bobin Sondhi  
Dated: 22-05-2021

## **Acknowledgement**

It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behaviour and acts during the course of study.

We are thankful to **Mr. Bobin Sondhi** for his support, cooperation, and motivation provided to us during the training period.

I am also thankful to every member of the Paxcom India Pvt. Ltd. organization for ingraining in me the essential information about the fields that incredibly profited me while learning and implementing the technical tasks.

I likewise thank my companions and friends who broadened their assistance and backing at whatever point required. Their commitment has consistently been huge.

## **Abstract**

Our project is a single page hotel room booking application built using angular and node.js that is a single page application utilizing node and angular architecture delivers most of the administrative function on the client side, which will allow the client to insert HTML pages and continuously update the item on that page, without updating it, as the client interacts with the program and helps individuals in booking and listing hotel rooms on the web and hence, this web application will help people to find, rent and lease out hotel rooms online.

In this web application we have basically 5 modules namely Auth, Booking, Payment, Rental and Routing modules. The Auth module consists of registration and login components. The Booking module consists of booked properties components. The Payment module consists of paid hotel rooms and bookings. The rental module lists out all the properties available for renting. The Routing modules consist of routing configuration paths which exist under the application. The Node.js architecture follows the MVC i.e Models, View and Controllers Logic. It has models and controllers for booking, user, rental, review and payments. We have also used AWS S3 for image uploading and Stripe.js for payment handling.

Keywords: Hotel Booking, Single Page application, Angular, Components, Directives, Modules, Routing, Node.js, MVC

# TABLE OF CONTENTS

<b>1. INTRODUCTION</b>	1-2
1.1 What is E-commerce	
1.2 Rise of E-commerce	
1.3 Shopping Online	
<b>2. LITERATURE REVIEW</b>	2-5
2.1 Advantage of Online Shopping	
2.2 Growth of web-based retailing in India	
<b>3. METHODOLOGY</b>	6-7
3.1 Project Design	
3.2 Problem Statement	
3.3 Objectives	
3.4 Field of observation	
<b>4. WEB-DEVELOPMENT</b>	8-20
4.1 Website	
4.2 Web-page	
4.3 Modern web application	
4.3.1 Single page application	
4.3.2 Multi page application	
4.3.3 SPA or MPA?	
4.3.4 Traditional vs Single page application lifecycle	
4.3.5 Should you go for SPA?	
4.4 Angular Framework	
<b>5. SOFTWARE REQUIREMENT AND SPECIFICATIONS</b>	21
5.1 Hardware Requirements	
5.2 Software Requirements	
5.3 Angular Modules Requirements	
<b>6. UML Diagrams</b>	22-25
6.1 DATA FLOW DIAGRAM	
6.2 USE CASE DIAGRAM	
6.3 ACTIVITY DIAGRAM	

# TABLE OF CONTENTS

<b>7. PROJECT</b>	26
7.1 Project Description	
7.2 Project Modules and Components	
7.3 Technical details	
<b>8. PROJECT'S USER INTERFACE</b>	27-31
<b>9. CODE IMPLEMENTATION</b>	32-36
<b>10. FUTURE SCOPE AND FUTURE ENHANCEMENT</b>	37
<b>11. CONCLUSION</b>	38
<b>12. REFERENCES</b>	39
<b>13. PROJECT REPORT PLAGIARISM RESULT</b>	

# **CHAPTER-1**

## **INTRODUCTION**

### **(1.1) What is E-Commerce?**

The traditional e-commerce business Electronic Website selling objects or associations, for example through into the Internet, thru computer systems. Electronic wallets start, for example, dynamic companies, electronic accounts, telephone boarding, website displays, planning for internet trade, electronic data trading, construction areas controlled and automated data edition systems. Today, great deals do use the World Wide Web to transact in every manner oblivious of how they might use movement of the earth, such as WhatsApp.

- Electronic businesses can use the following couples or relationships:
- Customer targeted web business. Web industries.
- Online purchase and sale.
- Collect and use site communication and application management category info.
- Exchange of customer electronic data
- E-mail or fax advertisement for coordinated and growing clients (for example, brochures)

### **(1.2) Rise of E-commerce**

India was indeed genuinely recognised for its high experience. The country's financial condition has enhanced postoperative inequalities, and India is emerging already as one of the world's leading nations. Moreover, it may appear differently from the motivational object, with more than 150 crore people and a production rate of more than 6%. There will be plenty of insights from this stage on why webcasting is growing even more quickly in India. As growth in remote towns expanded and more jobs have been created for unemployed youth, an increasing number of people were paid for and buy money to buy splendour item on the internet.

### **(1.3) Shopping Online**

Online shopping is that how clients purchases directly from stocks and companies on the Internet without a referee. This online ventures software is designed to create an electronic

interface for investors to create and sell web goods, the web is a simple and intuitive standalone website, and shopping functionality is now convenient for consumers.

It began with only two lakhs and now costs over 3000 thousand. Data indicate that the Flipkart Indian agency provided \$67 million to the Cooperation Department for the year 2012-2013. The compensation is forecast at 450 million dollars for the 2013-14 financial year.

A whole new universe of possibility was opened more by progress. Not only can we speak to a person we love in minutes, we have to take several simple tasks into perspective in a relaxed atmosphere, such as cooking, shopping, mixing and other matters. The concept is to develop an online business plan on one site, using a corning structure to move the majority of user-sized administrative functions, allowing the user to insert a single HTML page and to refresh it on that page on a frequent basis, without changing it, while the customer associates with and supports buyers. And on the website selling things.



## **CHAPTER-2**

### **LITERATURE REVIEW**

When buying clothes, in India there is still little spread and partition of electronic shopping between customers. The above problem was factored into the equation by a guardian's assessment of electronic buying behaviors. Based on structural characterization, the software regulatory regime was explicitly focused on four daunting psychographic measures, how, inspiration, character, and confidence. The models for online outcome were arranged after a true study of all four aspects. The solution is to manufacture one page web technologies use corner design, what transfer the vast majority of customer handling jobs, that will allow the client to stack and update the substance in a simple HTML page without reestablishing it once the retailer acts along with the application and help individuals in auctioning shops to require the buyer to stack and update a common HTML page without reviving it.

#### **(2.1) Advantages**

##### **1) Saving Time:**

It saves a ton of time for us. Both our helped services can be provided from our home and the train stations do not need to be used.

##### **2) Is profitable:**

Articles can be purchase by online shopping at any cost. This is because online shops offer insane limitations and remuneration for the inclusion of any single item.

##### **3) Additional:**

Tasks are received uninhibited, relocated to our homes with no shipping costs. In addition, we can return them without purchase everywhere at date, if they are not appropriate to our inspiration

##### **4) Shop around the world anywhere:**

Not everyone will be geographically obligated any more. At each venue on the country, multiple vendors don't have actual stores. Because when you shopping on the internet, everybody can get it with their side space from some unknown market.

##### **5) Legitimately sell blessings:**

You may mail rewards or even blessing boxes exclusively to their claimants upon request on the site. This means that you can save significant queues at the postal service and your blessing should be sent faster. Often shops sell red cards and prayers to cooperate with your present.

**6) Find some things you won't find potentially in shops:**

Since small sales are gravity, traders do not necessarily express the goods that we offer. On the Internet, alongside other colours, labels but even personalised products, you can access your whole stock.

**7) Web server runs 24/7:**

Purchase at any time signature stage. There's no worry, then, the web server never slows down. Online banking is great for people at night or someone who is too engaged to care of making a ride to the supermarket mostly in week. The display of the purchase of goods from markets rather than shops undoubtedly eliminates a broad attempt to mix up the Vietnamese customer's shopping point of view in the development, by the corner architecture of the internet, of an application using the vast majority of management work on the customer's side, as a customer, they work for the software, and support people to buy and sell online products.

After some time in the Near east the replay of BBC exchange has undeniably spread. There are a number of shopping areas where a conclusive flood is seen in busy weather (this declaration is being made while keeping in mind the presentation of shopping destinations over the last five years). In addition, they have been perfect steps towards advances and flags of multiple brands with a progress in the number of visitors to these places.

Digitized purchase certainly takes place over a timeframe that requires, for one, increasing numbers of internet services and mental incomes. Various products have been made acceptable for internet customers and one of them is a smartphone. Smartphones are available in beneficial quantities in various locations, free of the mark. The agreements that come from this online stage are very well for all of the other vital players in Communications correspondence, FMCG, organisations, convenience etc. The income generated is usually used effectively in order to boost basic research and development environment.

In comparison, customers have several outstanding benefits separated from the companies that make admirable advancement. Some of them use the unconditional products with any additional fee, for example, but many include rather prudent and mild, unusual deals.

**(2.2) Internet business growth in India**

India's online market has increased annually by 40% AZ09 and is dependent on the forecast of BY60 of \$38 billion entry.

M. Value Research Center revelations: Indian Electronic-commerce, TipToe of the Antarctic, split over 300 million ordinary jobs, an increasingly adaptable takeover and low levels of online market growth by the M. Value Research Centre.

"India's web-based industry is also in its early stages, with just a 1.6% of GDP compared to a further 2-4% for various countries, with just 15% of India's web-based executive networks compared to the further 54% in the US and more than half in China," Atula and Nitina said in their report. In addition, this enhancement will be augmented by free posting partnerships. In the Indian setting, only two web-based organisations, like Infoedge, and is the largest operation entrance to India, are registered.

India's online company possibility is also rare. Travel has a large share of the web-based market in India and yet electronic tracking has been the fastest to reach 19% of the business finance at a 69% level between AZ10 and 15M. It was originally from the East, the Indian internet company and now we recognise that it may be a more notable thing in the entire shopping landscape than it was in the East.

We all recognise that web marketing will become the fundamental strategy for certain young Indians to look for the term of their lives. Whatever huge open entrances, e.g. electronics, many marketers are fighting to be unquestionable leaders for brand acknowledgement and consumer experience and many times these players will change their strategies on the road to creating with the industry, "The paper said. That report.

## **CHAPTER-3**

### **METHODOLOGY**

#### **(3.1) Project Design**

Meta-Analysis Design is the sort of structure used for doing this. Contextual is a diagnostic technique designed to methodically evaluate, summarise, and extend the general example scale of the expert and his capacity to understand the repercussions of the quality from different individual inquiries.

#### **(3.2) Problem Statement**

The production of a single website business app that uses a corner infrastructure that pushes the overwhelming amount of user-side handling work to attract more customers to stack a single HTML page and gradually change the substance on that page through reviving it, while the customer works with it and lets people buy and sell online products.

A single page website programme that uses angular architecture to transfer the overwhelming amount of the customer handling jobs, which allows the customer to pick and change a solitary HTML page without re-living it, as a customer works with the platform and lets people rent and lease hotel rooms on the Internet.

#### **(3.3) Objectives**

1. The production of a single website business app that uses a corner infrastructure that pushes the vast bulk of user-side handling work to allow the customer to mount a single HTML page and gradually change the substance on that webpage without reviving it, while the product works with it and lets booking and leasing hotel rooms.
2. Implementing verification with Authorization from Node.js server.
3. Implementing login as a visitor is useful for mysterious clients to visit our application.
4. Creation of database using Mongo DB Atlas.
5. Implementing REST APIS for structured application.

### **(3.4) Field of observation**

1. Locating qualities and shortcomings of single page application.
2. Locating quantity of hotel rooms booking.
3. Locating consumer loyalty.

## **CHAPTER-4**

### **WEB-DEVELOPMENT**

Online development is an extensive concept used to create a web site for the web or intranet Web production (a private framework network). It can hit the most bewildering on the web apps, electronic partnerships and casual associations, by operating on the least volatile static single page of a simple matter. The web construction, website organisation, new content creation, client touch, customer side-by-side scripting, web servers and the system safety scheme and web company enhancement may be an indiscriminately comprehensive once again of the projects web advancement means continuously. "Web-app enhancement" generally among web specialists. The most recent migration to the Web has assumed that managers' programmes are creating substances. This system creates the web application using an angular architecture that moves the vast majority of the customer-free handling work, allowing clients to set up a solitary HTML page and gradually update the content on that page, with reviving, as the buyer works with the website and helps people to buy and sell items

For larger affiliations and partnerships, web forward meetings will hold several individuals and maintain standard methods, such as Agile systems, while local builders form a strong page Internet business application, using an architecture the moves the overwhelming majority of controlling work to a plaintiff's side.

A lonely or unalterable architect or an assisted company undertaking, such as the planner or master of information technology, can merely be required to join Tinier affiliations. Instead of being a dished off workplace, web improvement can be a community effort among divisions. There are three types of web technicians, namely the initiator, the architect and the engineer.

#### **(4.1) WEB-SITE**

A webpage is a mixture of linked sites, including media contents, usually designated by a general area name, and sent to a single web server on every occasion. For instance, the Internet, or a private network region (LAN), a web page may be opened using a standardised resource locator (URL), which receives the site, through means of accessible Internetx Protocols (IP) settings.

Destinations have different restrictions and can be found under different materials; a domain can either be an individual site, an agency website, or an affiliate site that is not a benefit. Locals are usually dedicated to a specific point or cause which reaches out from cheap interaction to news and preparation. All the perceived uninhibiting destinations include the World Wide Web, however private locals, with example, for its delegates, are still a bit intranet.

Pages, which are structural squares of destinations, are records, usually in simple words, combined with principles of orchestration, like the Hypertext Markup Language (HTML, XHTML). Segments from different locations with appropriate marking captures may be united.

Therefore, build a single page web business application using angular architecture where moves the vast amount of online work, allowing customers to stack a lonely HTML page and update the substance without reviving the page, as the customer collaborates with the app and helps individuals buy and sell items on the web. It builds a single page internet business app using hierarchical architecture and transfers much of the work to the client, helping clients to stack a single HTML page and update the substance gradually on that page through reviving it.

## **(4.2) WEB-PAGE**

A blog or a page on the site is a WWW or Site program-sensitive report. On a screen or cell phone, a browser application displays a website page. The link is what exists, but in the same way the word means a PC record, usually written in Zip or the language of increase equivalent. Online programmes arrange for something like the page's formats, substances and images to display the characteristic internet web segments.

Natural search engines offer multiple links, as far as possible implied as connections, toward a tutorial bar or a dashboard tab with other website.

In one way, a single-page web based business interface can be created that uses cubic architecture to transfer the vast bulk of the management work to the client's side, which allows the user to stack another solitary Style sheet and refresh the substance gradually, without revitalising it.

## **(4.3) MODERN WEB APPLICATION**

They unintentionally replace the old programme of the workspace as promote happiness to use, they are hard to recover and not limited to one gadget. In addition, while customers are tenderly scalable, there is now a huge concern for sloppy and sophisticated implementations. If you're thinking about making my angular application, you've already heard the two main better comprehend are available for MPA and SPA web services. Of course, there are both similarities and drawbacks in all methods.

Before you start translating the idea into a modelling tool for a single web page using angular structure which moves most customer work, enabling the patient to rack a solitary Web pages and steadily updating the substance on that page by reviving it, because the customer works together and helps the application

There are various advantages and drawbacks of singleton apps, as well as micro apps, which enable the consumer to package an HTML page alone and to upgrade the substance slowly, without reviving the page, as a customer collaborates with and allows people to buy and sell products on the internet.

### **(4.3.1) SINGLE PAGE APPLICATION**

A single-page application is a native application which works with the web programme to replace the original strategy for storing entirely various pages by capable of re-doing the current website page with updated information from the web server. The intention is to speed up the development of the site as a neighbouring application.

The theoretical foundations and the main HTML, JavaScript and CSS code are recovered either by a single page loaded programme or by intensive storage and insertion of the necessary tools into the page as important, usually in view of the customer's work. The server does not restart or transfer power of any page at any time, regardless of how the area hash or the History API may be used to provide recognition and navigation to the free authoritative sites within the application.

The following, The one-tone application is a program-like app that does not need to refresh a tab during navigation. Nearly everyday, you use this kind of job. For example: Gmails, Google Street view, Twitter or Youtube.

They are associated with serving a shocking UX by trying to mimic a software integrated with "trademarks." It's just one website you access and you stack all the other material through JavaScript — which they really depends on.

SPA wants unreserved marking which results, and makes a single page web application using an angular architecture that takes the vast majority of controlling task to the customer's side, which permits the customer to pick up a Style sheet and to change it gradually without rejuvenating it as the buyer cooperates with the project and supports others in that process.

#### **Benefits:**

- It is expensive, since most services are loaded in the future of use. Just data is being navigated.
- The progress is smoothed and simplified. There is no persuasive reason for code generation. It's much less hard to start because you can normally start from an Ivar record, even without server.
- They certainly aren't difficult to identify with Browser, because you can monitor masterpiece drills, look at page sections and details.



- This is the perfect approach for customers who need a visual guide to match the app. Powerful, almost no route dependent on route is a basic element of the traditional application.
- They are both perfect and easy for the board to optimise its legal search engine. It provides more possibilities for different stock phrases as a single catchphrase implementation for each part can be improved.

### **Loss:**

- They are particularly risky and do not make the search engines easy to optimise. The Async-await JavaScript is stacked by default, a method to share and revitalise data and devices without trying to revive the page.
- It is delayed for downloading since the client will be supplied with striking customer systems.
- You need a JavaScript converter to be opened and activated. That's almost impossible if a customer cannot deliver the software and exercise correctly.
- This is less secure. This is less protect. Through the Sql injections (cross-site script), it enables consumers to insert side materials in software applications of various customers.
- SPA transmission of data makes the incredibly wonderful approach easy to explain.

### **(4.3.2) MULTI-PAGE APPLICATION**

The regular apps that insert the entire website and view the update when a customer is connected to the website are tri apps. Another on-line on-server tab is identified as the data is sent to and fro on the Internet explorer. This method attempts to create the server pages, send them to system and expose they in the app, which affects the customer's features.

Whatever, AJAX made it necessary for the submission to only be created in a certain chapter, but this makes the process of progression considerably more troubling and difficult. for now.

### **Benefits:**

- This is the perfect approach for customers who need a visual guide to match the app. Powerful, almost no route dependent on route planning is a basic element of the traditional application.
- They are both perfect and easy for the board to optimise its legal search function. It provides more possibilities for different catchphrases as a single guiding principle implementation for each feature can be improved.

### **Loss:**

- There is no solution to the compact component counterpart backend.
- The progression on the frontend and backend is strongly linked.
- It turned out that the progress is really stupid. For customers and back-end developers, the programmer wants to use applications. The results have been drawn to further use.

### **(4.3.3) Which one to choose?**

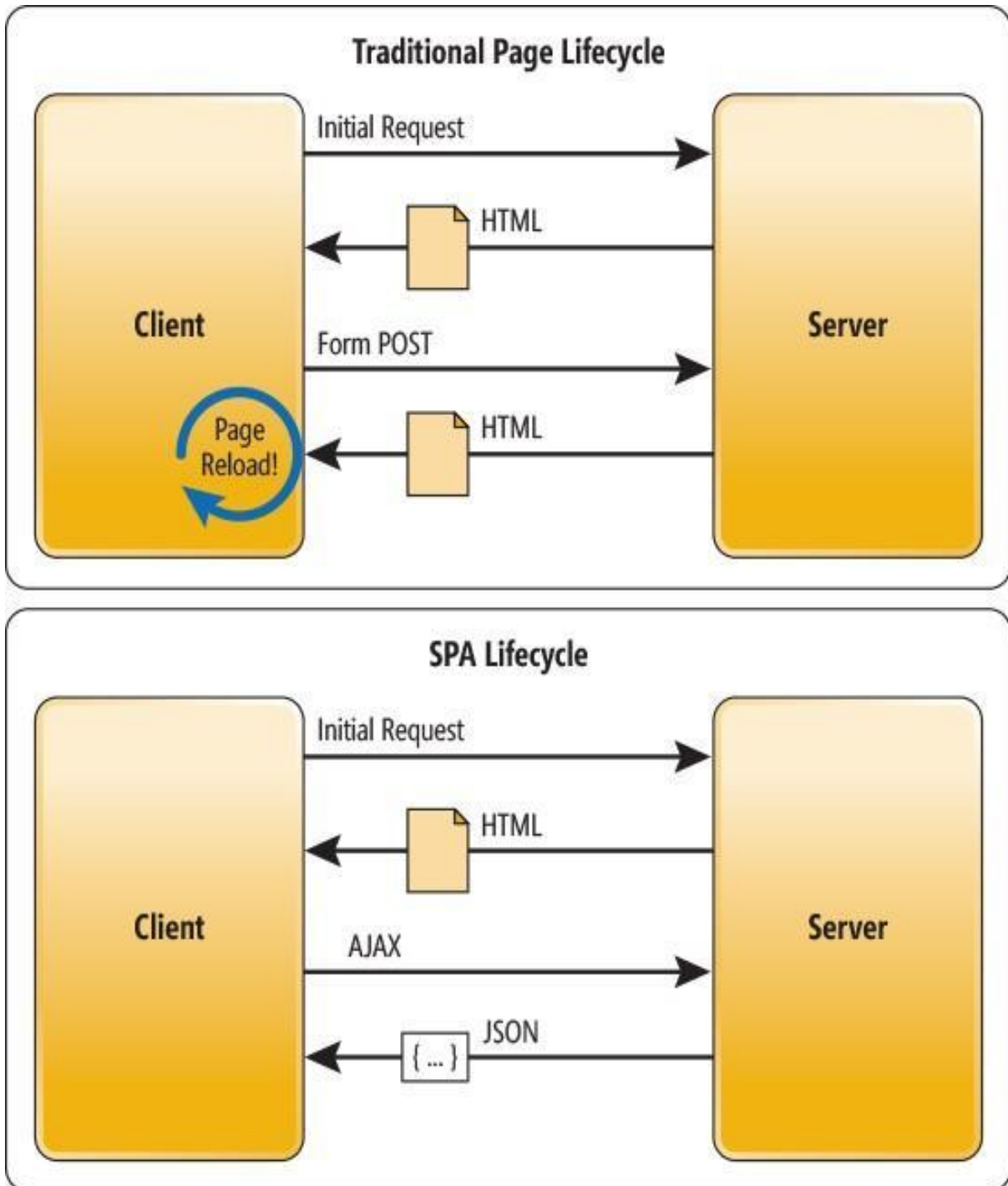
#### **SPA or MPA?**

About the fact that an implementation method can be used by all kinds of web applications on a single website, here are a few things to remember before you let it go.

It is essential that you mention in your web application the elements and capacities you require. Does an SPA have the equity and highlights you need? For instance, does your website gradually increase content or will it be focused on illustrations? Or both of them on the other hand? Can all patterns and resources be fitted on one page?

Does the company control or manage explicitly or provide food with a variety of products or leaderships? If your company is able to handle a lonely object, nothing but a solo page submission will be excellent. If your company has a range of administrators or products, you must certainly sell Importance for each by using a multi-page programme to bifurcate them in different pages. This will also allow you to link growing page in depth.

#### (4.3.4) SPA vs Traditional app

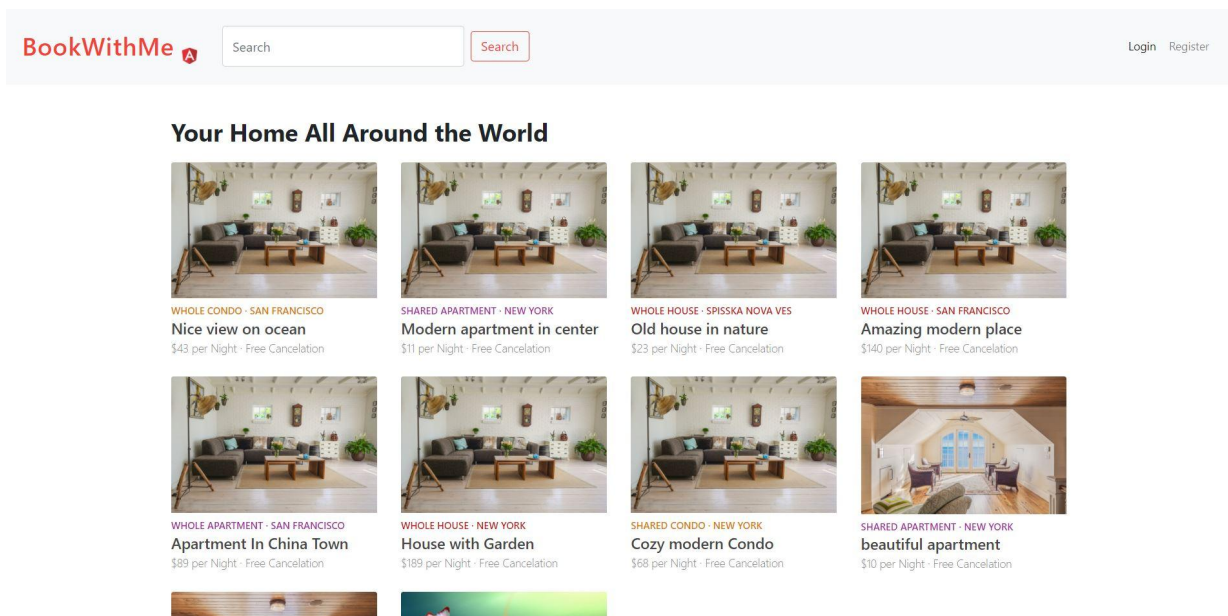


### (4.3.5) So, should you go for the SPAs?

About the fact that an implementation method can be used by all kinds of web applications on a single website, here are some other things to remember before you let it go.

It is essential that you mention in your website the elements and capacities you require. Does an SPA have the equity and highlights you need? For instance, does your website gradually increase content or will it be focused on graphics? Or both of them on the other hand? Can all layouts and materials be fitted on one page?

Does the company control or run explicitly and also provide food with a variety of products or appointees? If your company is able to handle a lonely object, and nothing more than a solo page submission will be excellent. If a number of administrators or things are covered in your market, then you must undoubtedly give importance to each one of them via an intra application websites. This will also allow you to link each page in depth.



*(Book With Me — Single Page Application)*

If you want to re-enact your customer continuing without even a page? An ongoing automotive configurator, for instance.

Today's web apps have rich and provide a wonderful client interface regardless of whether they are extremely advanced SPAs or MPAs. But you should consider the needs of certain company and the customer in order to use these conventional technologies most severe.

## (4.4) ANGULAR FRAMEWORK

It is the platform and architecture for the application of extremely compact, modular applications of one-ton paying.

### Prerequisites

You must be familiar with the accompanying structure to use the angular structure:

- JavaScript
- HTML
- CSS

TypeScript information is helpful, but not necessary.

### Node.js

Ensure you have node.js and a manager of the npm package.

Exactly needs the latest, dynamic LTS or LTS node.js interpretation support. See the engine key in refer to attached data for the specific rendition requirements.

Run `hub - v` on a terminal / convince window to monitor the adaptation.

Go to [nodejs.org](http://nodejs.org) to get Node.js.

Precisely, the Angular CLIs and Angular modules use repositories accessible in the form of npm packages for highlights and utility. You can have a npm package supervisor to obtain & introduce npm packages.

The framework between customer lines npm and naturally Node.js.

### Step 1: Install the Angular CLI

You use the Angular CLI for the development, processing, packaging and shipping of programme and library technology and a number of advancing businesses.

Present the Angular CLI in.

Open the terminal / help application and access the corresponding computer to implement CLI using npm:

```
npm install- g @angular/cli
```

## Step 2: Create a workspace and beginning application

With respect to angular workspaces you build programmes.

1. To build a different workspace and initial use:  
Bootstrap ngenew operating system and call it my script, as seen here:  
Ng fresh my request
2. The latest order asks you to recall details on excerpts for the submission.  
Enter the Return key to recognise the defaults.

## Step 3: Run the application

The Angular CLI has a server, so you can support your user locally with next to no stretch form. Go to the coordinator of the workshop (my-application). Start the server using the ngxserve CLI command. Ng servve command sends the server, monitors the documentation as you make the updates to the logs and modifies the programme.

The options —open (or only -o) open naturally to <http://localhost:4200/> for your programme.

### (4.4.1) Hierarchy of Angular application

Every Angular application consist of:

1. A root modules referred to as AppModule which work as container in the following structures for standard functionality, directives, pipes, utilities and many features:

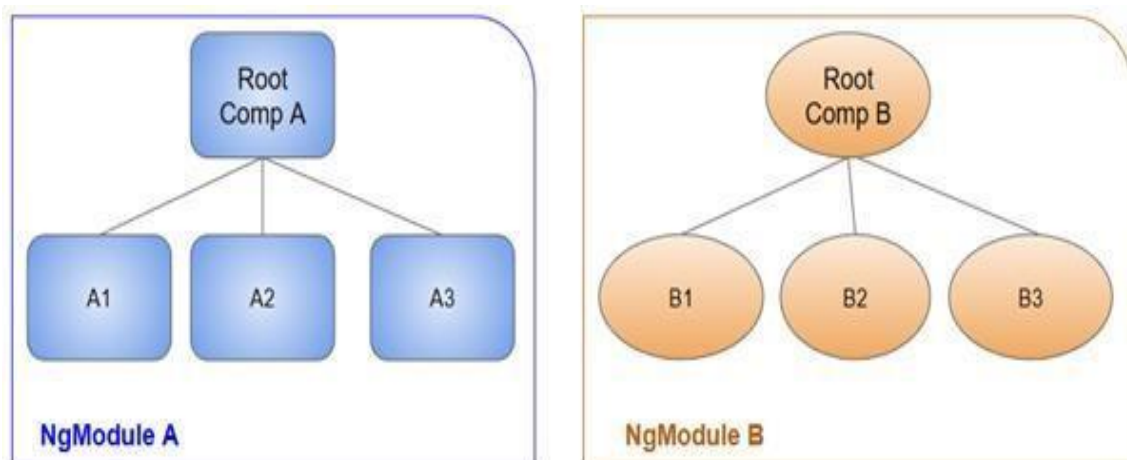


Fig 4.4.1 (a) Hierarchy of angular application modules

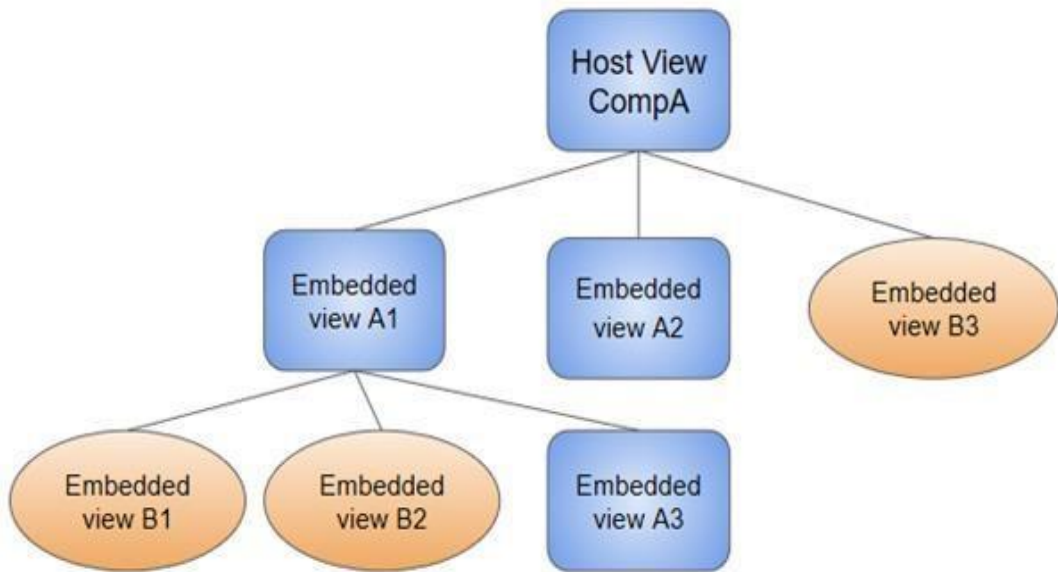


Fig 4.4.1 (b) Root Modules Heirarchy

2. A root component known as AppComponent which contains the app view logic and others modules act as child components in a tree structure as follows:

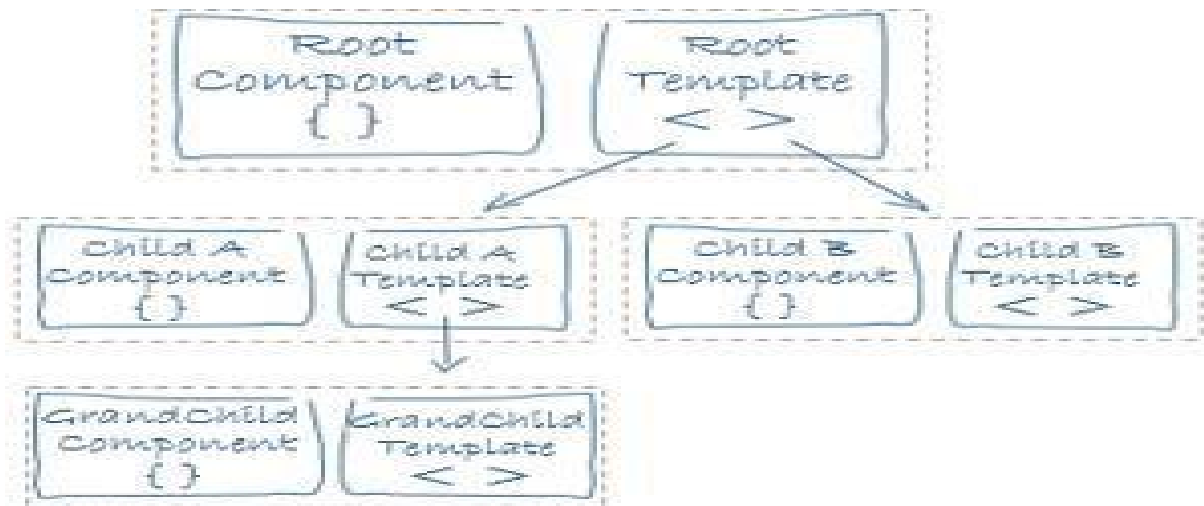


Fig 4.4.2 Child Component Heirarchy

3. A service file that contains application business logic and are injected into components class constructor as as dependency through Angular's Dependency Injection system demonstrated as follows:

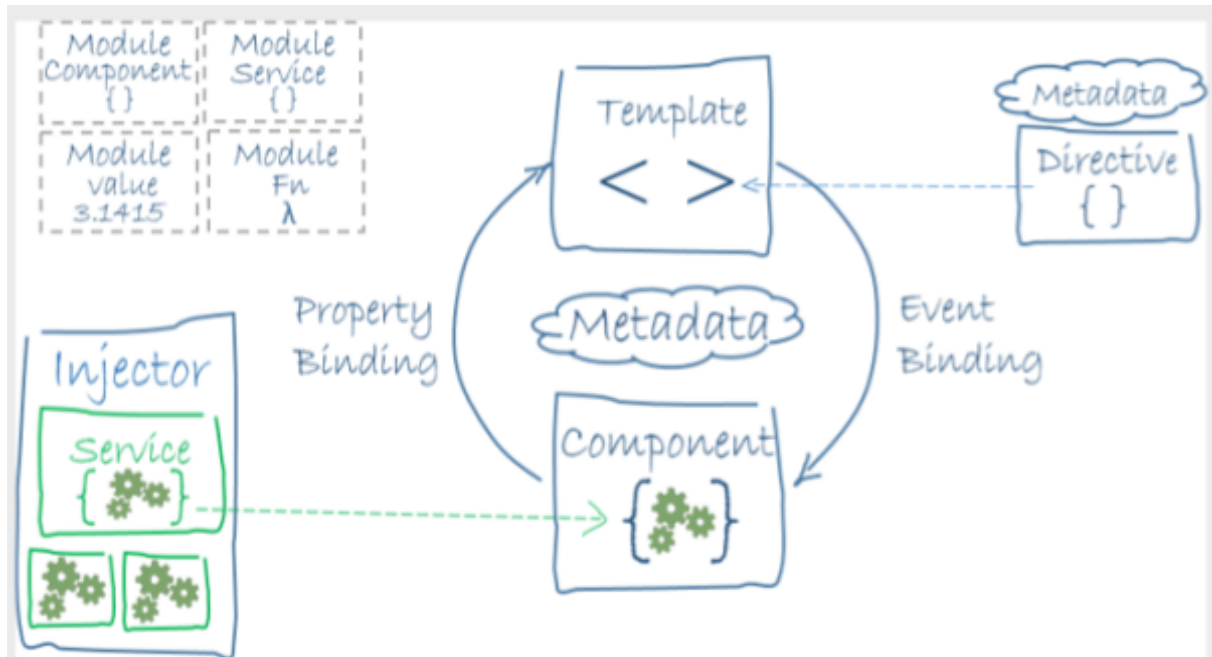


Fig \_ 4.4.2 Basic Structure of angular application

(4.4.2) Structure of Angular application

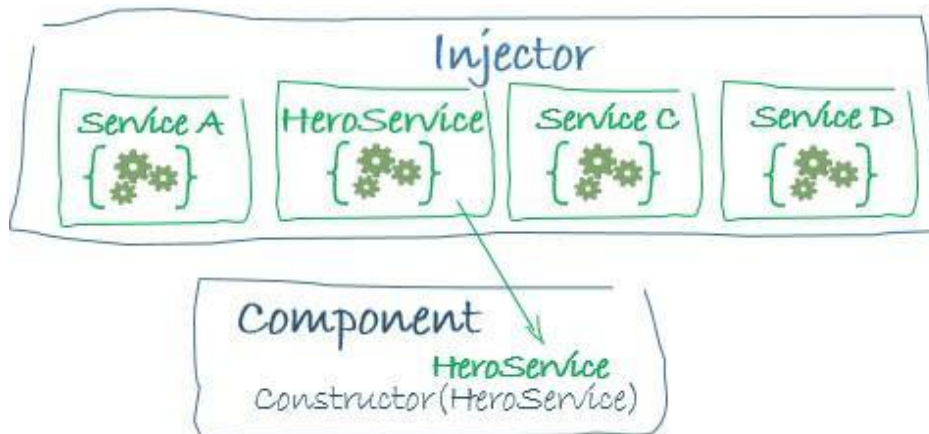


Fig 4.4.3 Dependency Injection

## (4.5) NODE.JS

Node.js is a Google V8 JavaScript programming tool also recognized as the Node only. In JavaScript, you code Node.js, but then create V8 to run it. Some or even any Node.js Application Code, including web server and server scripts as well as other web platform supports, will be written. You will write. The webpage and web service scripting languages



supported by the same web server ensure a better communication seen between web server as well as the scripts. Only a few reasons are below why Node.js is a great framework:

JavaScript End-to-End: Perhaps one Node.js' biggest strengths is to allow you to write either client and server scripting to JavaScript. If reasoning could be inserted into Client Scripts and Server Scripts was also difficult to decide. You could write JavaScript upon on client's server and easily adapt it along with Node.js and conversely. An additional advantage would use the same language with customers and server programmers.

Event-built scalability: Node.js applies to web requests an unique logic. Node.js manages them on the very same thread that used a basic event model rather than make multiple threads awaiting on web requests. This allows Node.js not to scale traditional web servers.

Extensibility: Node.js has an extensive community of successful development with follow-up. New modules are provided to constantly increase the capacity of Node.js. New Node.js modules can be easily installed and connected and a Node.js project can be expanded with new functionality in minutes. Fast implementation: setting up and growing Node.js is super fast. Node.js and a working web server are installed in some time.

For a wide range of software node.js may be used. As the code for traffic control is heavily rooted in V8 and optimal in HTTP it has been used most often as a web server. However, with many other websites, for instance, node.js is also used:

- I. Online tools APIs include real-time REST games
- II. Web backend functions such as server calls, cross-domain
- III. Internet applications
- IV. Collaboration with multiclients like IM.

Node.js has several integrated plugins, which are discussed in the following components:

- Buffer: Enables TCP stream connectivity as well as system files functionality.
- C/C++ additional: allows use, as with every other Node.js unit, of C or C++ programming.
- Processes for children: Help develop baby processes.
- Cluster: Enables to be using multi-core equipment.
- Command-line choices: From a console, issue instructions to Node.js.
- Server: Provides the client with a monitoring console.
- Crypto: Allows implementation of custom encryption.
- Debugger: Makes testing file Node.js.
- DNS: Allows links to DNS service.
- Errors: Allows the handling of errors.
- Events: Allows treatment of asynchronous events.
- File System Makes both the I/O file and the I/O file approaches asynchronous.
- Globals: Makes available modules widely used without having to include one first.
- HTTP: Allows supporting some HTTP features.
- HTTPS: TLS/SSL Web prompts.
- Modules: Includes the Loading Structure of Node.js programme.
- Net: Enables the growth of servers and customers.
- OS: Provides OS control where Node.js is running.
- Path: Allows entry to the path file and folder.
- The method collects information and makes the current Node.js procedure monitoring easier.

- Query Threads: Includes the parsing and formatting of URL queries.
- Return: Allows a reading interface for a stream of data.
- REPL: Aid people to create a command shell.
- Stream: offers a stream interface abstraction development API.
- String decoder: Has an API encoding buffer element in strings.
- Timers: Allows to call or schedule upcoming projects.
- SSL/TLS: Supports TLS and SSL protocols.
- URL: Enables URL resolving and decoding.
- Utilities: supports various modules and structures.
- V8: Opens the Node.js APIs V8 version.
- VM: Allows the V8 virtual machine to run and compile javascript.
- ZLIB: Gzip or inflated compression are supported.

#### (4.5.1) Loop of events in Node.js

The first thing the node has, as most languages, packages and plugins. There are libraries which can be loaded and is used by npm through our code (node package manager). A number of basic plugins have now been enabled if a node is created on your server. That's how, but later, we create a simple server.



Figure 4.5.1 Loop of Event in Node.js

They recognize that it is indeed single threaded but asynchronous because you know JavaScript. The single thread is indeed the case loop which performs every task and query. The asynchronous behaviour is extremely important when using a node because it means that the event loop will never be interrupted by any synchronous action.

And when only a case loop exists, the loop will pass the demand to an asynchronous task execution feature when the request is set. After the function has been completed and the response returned, the event loop may be returned towards the receiver and submitted to the recipient via callback.

If the functions are synchronised, the cases loop is lock and responded to by one consumer, with all customers waiting for the customer. With JavaScript asynchronous, node-based

programmes can concurrently handle many queries. This illustrates that you still have to take account of the fact that now the functions published in Node.js really aren't synchronous when programming. Therefore, capturing server errors returning toward the client is extremely necessary. This prevents errors which might crash the device and affect all users.

### (4.5.1) Disadvantages of Node.js

Since the app uses a lot of computer technology or numerous algorithms, the event loop would slow down. It supposes that any user is frustrated when they all try for stop or trigger the event loop on the same computer. The hell callback is another problem that can occur. That happens if the code is incomprehensible, or difficult to handle, because too many calls are nestled in each other. It can be avoided, so it takes time to understand how to think asynchronously. The packages from npm are also another challenge. Because Node.js is fairly new packages and modules, many websites and apps are compatible.

### (4.6) MongoDB

NoSQL databases are unstable and retain rather than related data databases (also regarded as not just SQL). The data sources of NoSQL are built on a variety of forms of the data model. Text, main sense, long rows and diagram are the main features. It has flexible systems and large data loads are easily measured.

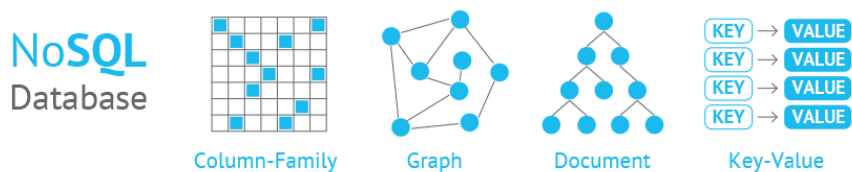


Fig. 4.6.1 NoSQL DB Types

#### (4.6.1) NoSQL DB Types

What kinds of NoSQL databases are there?

Four major types of NoSQL databases, i.e. text databases, crucial databases, broadcast storage and graph databases, are being developed over time. Want to see some shape.

The JSON (JavaScript Entity Notation) storage descriptions of document documents recording artefacts. Each text uses pairs of fields and values. Values will usually include strings, numbers, tables, structures, and entity representations, usually comply with creators of programming. The values are different. Due to their different field values and effective query languages, the paper repositories would be used as a database to development purposes, are perfect for many purposes.

Key-value registry is a simplistic database type so each object contains key and value. A value is normally only restored by referring the key, so it is generally easy to find a certain pair of key values. When large amounts of data are processed, the key data bases are ideal to be used, but advanced searches are not enough to find. Popular cases of usage require user settings to be saved or cached. Redis or DynamoDB are the most important popular databases.

### (4.6.1) Working of NoSQL DB

One approach to understand the NoSQL database attraction from the architecture perspective is to see if SQL and NoSQL data structures in an unnecessarily simplified situation may be viewed with address data.

The SQL's event. In a SQL database, the database initialization begins when the structure is created logically and the document is assumed to be almost unchanged. After evaluation of the query patterns, the SQL database will allocate storage in the second, one for essential information and one for consumer applications. The last name becomes the key to all tables. For each column within every table there is a single customer, and with each row there are set attributes:

Each NoSQL database format is designed to represent a specific customer circumstances and so every application category is provided with technical details. The shortest concept is the repository of documents in which detailed information and user data are naturally combined in one JSON database. In this case, the consumer data is the record knowledge for the fields in each SQL column parameter but also for the values from each field.

### (4.6.2) Disadvantages of NoSQL DB

NoSQL has the following disadvantages.

Narrow focus – NoSQL databases are highly focused and most of them are storage optimised, but have very minimal functionality. Interaction servers are a better alternative to NoSQL in the field of transaction management.

Open Source – NoSQL is the open database. The NoSQL specification is also not consistent. Or most other terms, two database architectures should be consistent.

Challenge of management – The aim of big data applications is to make navigating a large amount of data as simple as possible. Regrettably, it's not that quick. The management of NoSQL data seems to be more complicated than a relational table.

MongoDB doesn't have direct data retrieval route.

Wide text size – JSON information is saved by such database structures like MongoDB and CouchDB. Documents (BigData, bandwidth, accelerations) are massive, which only damage expressive key names by increasing text size.

Table 4.6 Mongo DB and RDBMS Differences

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysql/Oracle	mongod
mysql/sqlplus	mongo

## CHAPTER-5

### SRS-SOFTWARE REQUIREMENT AND SPECIFICATION

#### a. Hardware Requirements

The assurance of equipment is huge in the nearness and fitting working of any item.

While picking equipment, the size and need are also critical.

Processor	Intel core i7
RAM	16 GB
HDD	1TB

#### b. Software Requirements

Windows	7, 8, 10
UI development languages	HTML, CSS, BS4, Angular Material, MDC
Scripting languages	JavaScript, TypeScript
UI Framework	Angular 2+
Sever	node json-server
Database	JSON {key:value} collection pair
Tools	MS Visual Studio, Angular CLI, node package manager

#### c. Angular Modules Requirements

Module name	Installation command
Social login module	npm i --save <u>angularx-social-login</u>
Pagination module	npm i ngx-pagination

# CHAPTER-6

## UML (UNIFIED MODELING LANGUAGE) DIAGRAMS

### (6.1) Data flow Diagrams

Visualizations show the data exchange in the setup from analog circuitry, then the computer's surroundings. Flow chart for data There are four icons to draw a DFD:

- I. Place that represents our sense of understanding.
- II. The principal features that define the oval.
- III. Designation definition of the rectangle.
- IV. Disk image which identifies the ultimate essence information.

The figures below are the data flow diagrams of the SPA Web Server Platform. Any step in the structure first emerges as a tier 1 DFD, then a minor DFD type element.

#### CONTEXT LEVEL DIAGRAM



Fig 6.1

#### First Level DFD

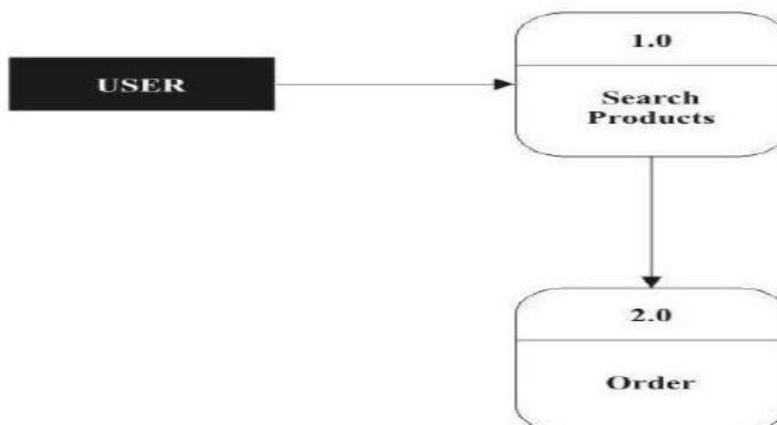


Fig 6.1.1

### SECOND LEVEL DFD

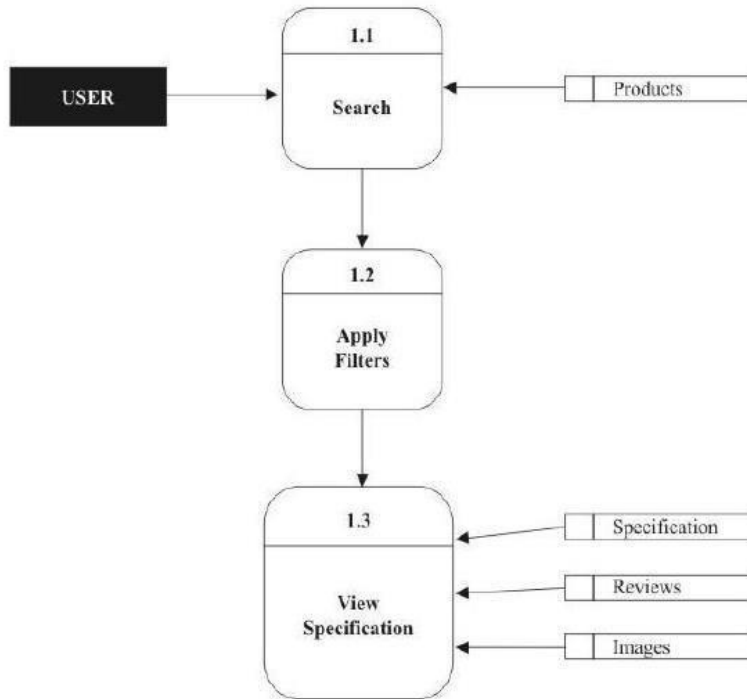


Fig 6.2.1

### THIRD LEVEL DFD

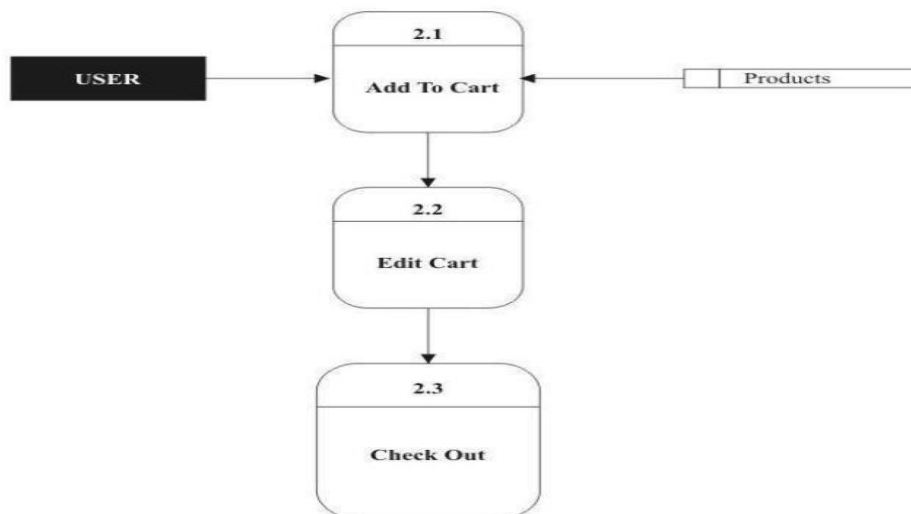


Fig 6.2.2

## (6.2) Use case Diagram

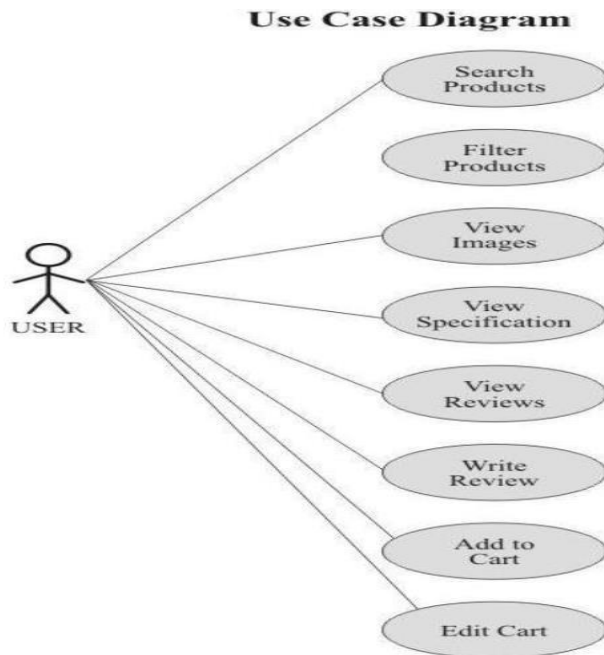
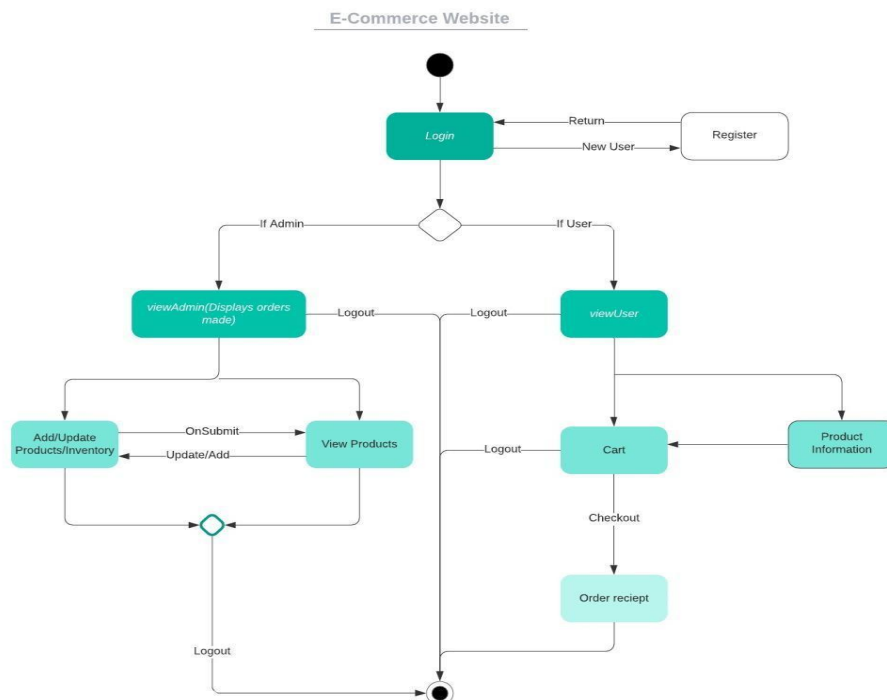


Fig 6.3.1

## (6.3) Activity Diagram





# **CHAPTER-7**

## **PROJECT**

### **7.1 PROJECT DESCRIPTION**

**Name:** BOOK WITH ME HOTEL BOOKING APPLICATION USING ANGULAR AND NODE.JS

**Technologies Used:** HTML, CSS, BS, JS, TS, Angular, Angular Material, MDC, Node.js, MongoDB, AWS S3, Stripe.js

**Tools Used:** MS Visual Studio code, Angular CLI, Node package manager

**Server:** Node Server

**Database:** Mongo Atlas DB

**Operating System:** Windows

### **7.2 PROJECT MODULES AND COMPONENTS**

#### **1. Auth Module**

- 1.1. Registration component
- 1.2. Login (Manual/Google) component

#### **2. User View Module**

- 2.1. Toolbar component
- 2.2. Sidebar component
- 2.3. View Hotel Listings component

#### **3. Cart Module**

- 3.1. Checkout component
- 3.2. Booked Hotel room component

#### **4. Routing Module**

### **7.3 TECHNICAL DETAILS:**

□ JS, Angular, Corner and MDC data. Ajax and XHR used to load Html in an on-line compiler, so the browser understands and thereby makes client-side script transferring TypeScript to JavaScript.

Database is made on MongoDB Atlas.

Reactive forms are made using Angular.

- 
- 
- The whole web application is a single page application build using Angular (JavaScript client-side frame) and server built on node

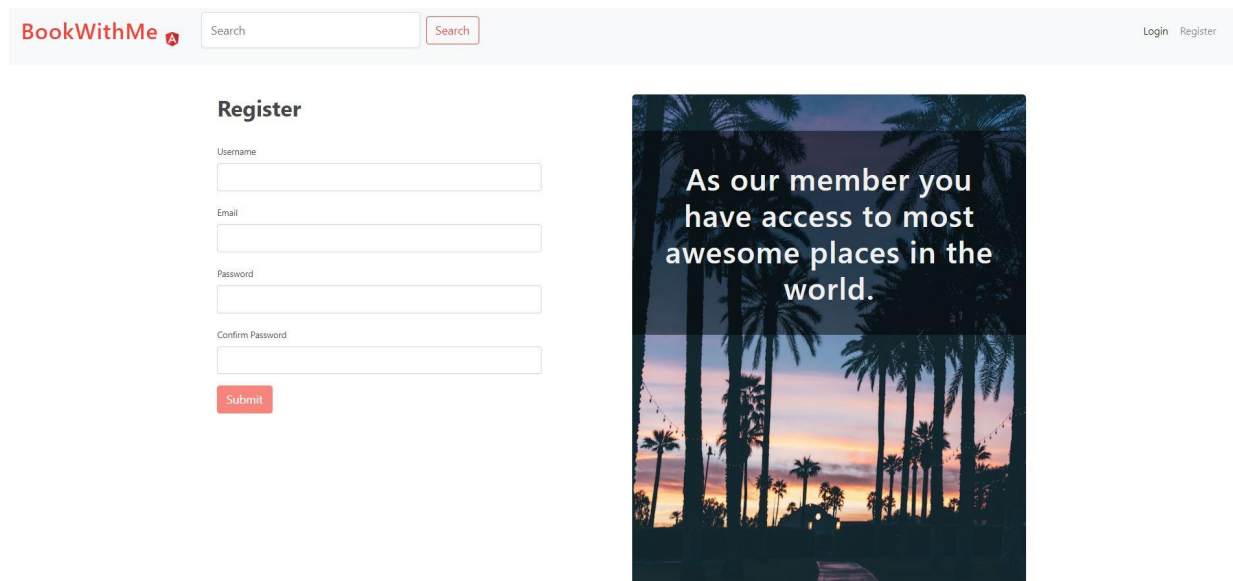


Fig 8.1 Registration view

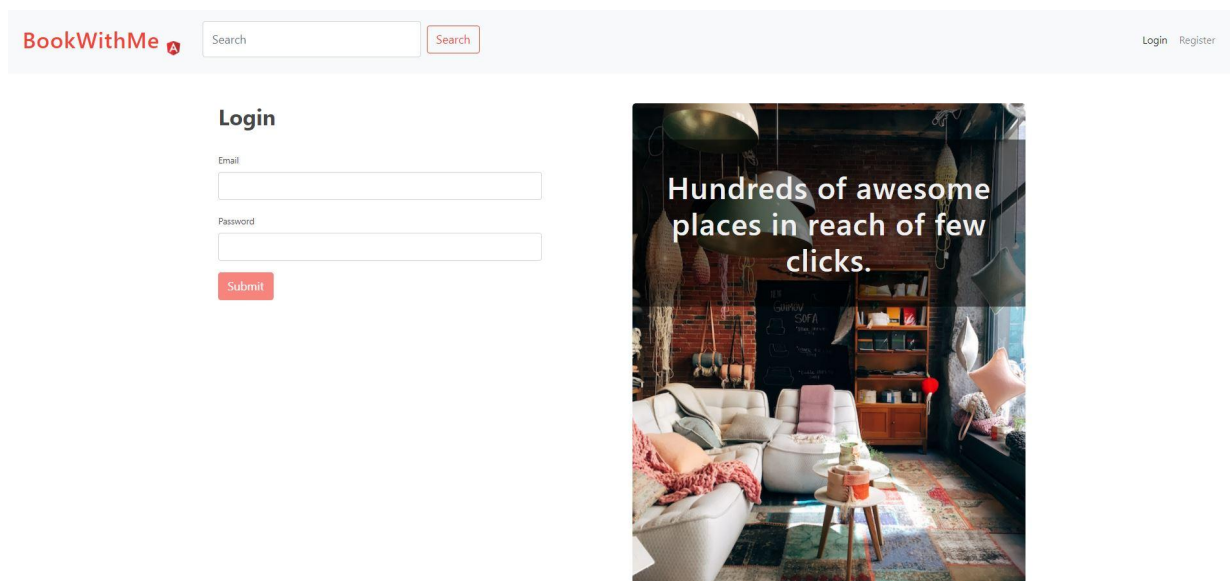


Fig 8.2 Login view

**Your Home All Around the World**

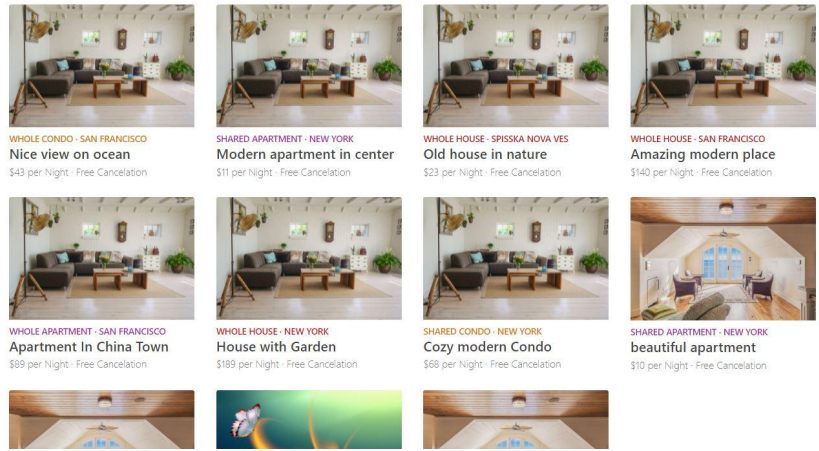
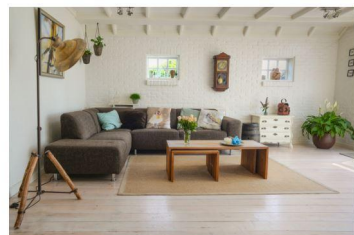


Fig 8.4 Home view



WHOLE CONDO  
**Nice view on ocean**  
 San Francisco

4 bedrooms 8 guests 6 beds

Very nice apartment in center of the city.

0 / 5

**Assets**

- ★ Cooling
- 🗑 Working area
- 🔥 Heating
- 🧺 Washing machine
- 🔪 Iron
- 🧼 Dishwasher

\$43 per night


Test User

[Login and book this place today](#)


People are interested into this house  
 More than 500 people checked this rental in last month.

Fig 8.5 Rental view


### Your Home in New York




SHARED APARTMENT - NEW YORK  
**Modern apartment in center**  
\$11 per Night - Free Cancellation




WHOLE HOUSE - NEW YORK  
**House with Garden**  
\$189 per Night - Free Cancellation



SHARED CONDO - NEW YORK  
**Cozy modern Condo**  
\$88 per Night - Free Cancellation




SHARED APARTMENT - NEW YORK  
**beautiful apartment**  
\$10 per Night - Free Cancellation



SHARED APARTMENT - NEW YORK  
**beautiful**  
\$10 per Night - Free Cancellation

Fig 8.6 Search View



SHARED APARTMENT  
**Modern apartment in center**  
New York  
1 bedrooms 5 guests 3 beds  
Very nice apartment in center of the city.  
0 / 5

**Assets**

- Cooling
- Heating
- Iron
- Working area
- Washing machine
- Dishwasher

**Reviews**

#### Confirm Booking

2021/05/21 to 2021/05/28

7 nights / 11 per Night  
Guests: 5  
Price: 77\$

Card Number: 1234 1234 1234 1234  
Expiry Date: MM / YY  
CVC: CVC

Do you confirm booking for selected days?

1 per night

2021/05/21 - 2021/05/28

Guests: 5

People are interested into this house  
More than 500 people checked this rental in last month.

Fig 8.7 Payment View

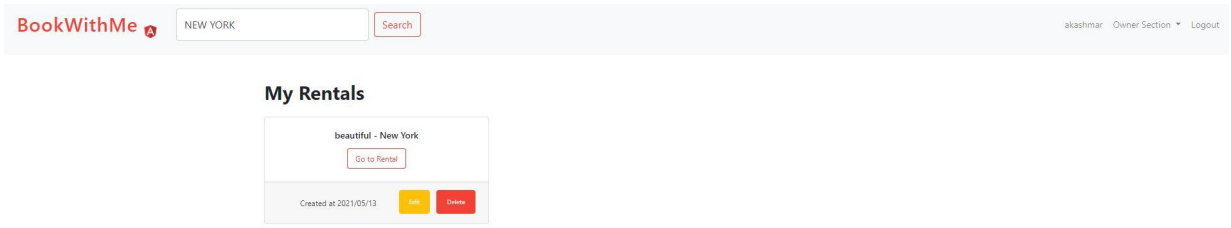


Fig 8.7 Manage Rental View

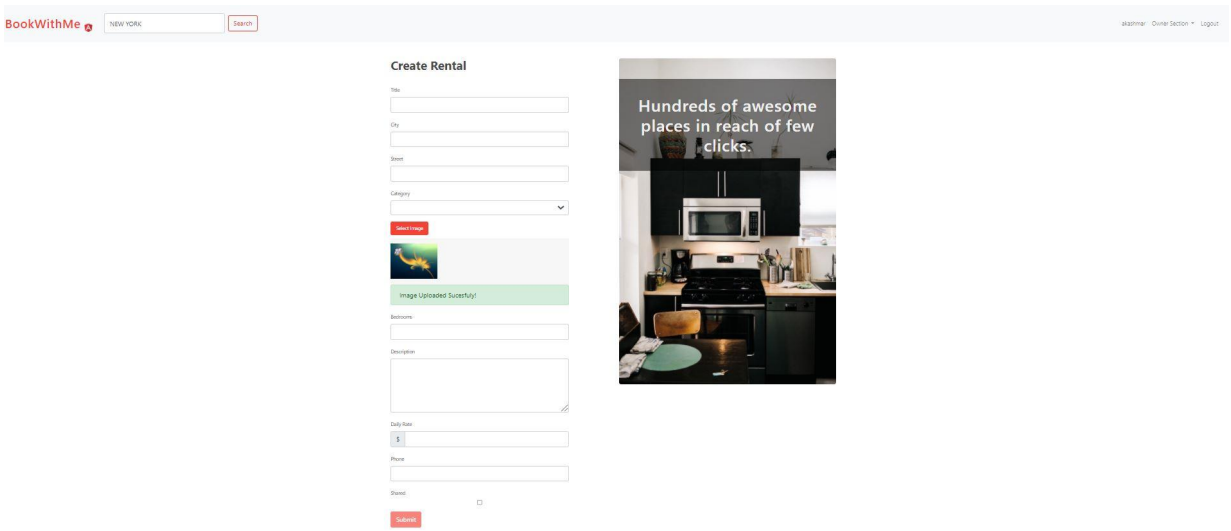


Fig 8.7 Create Rental View

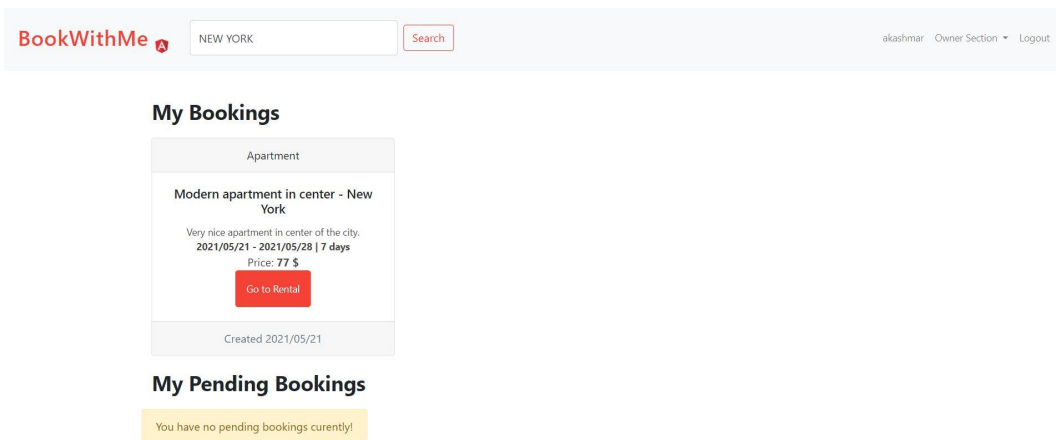
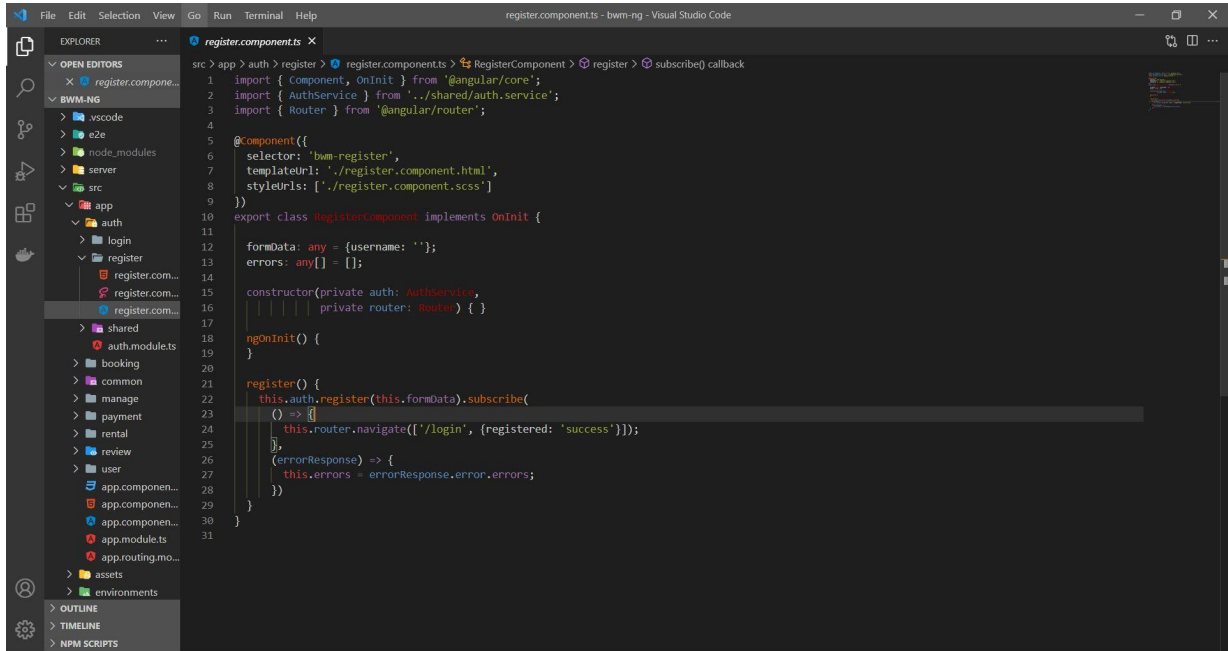


Fig 8.8 Booking View

# CHAPTER-9

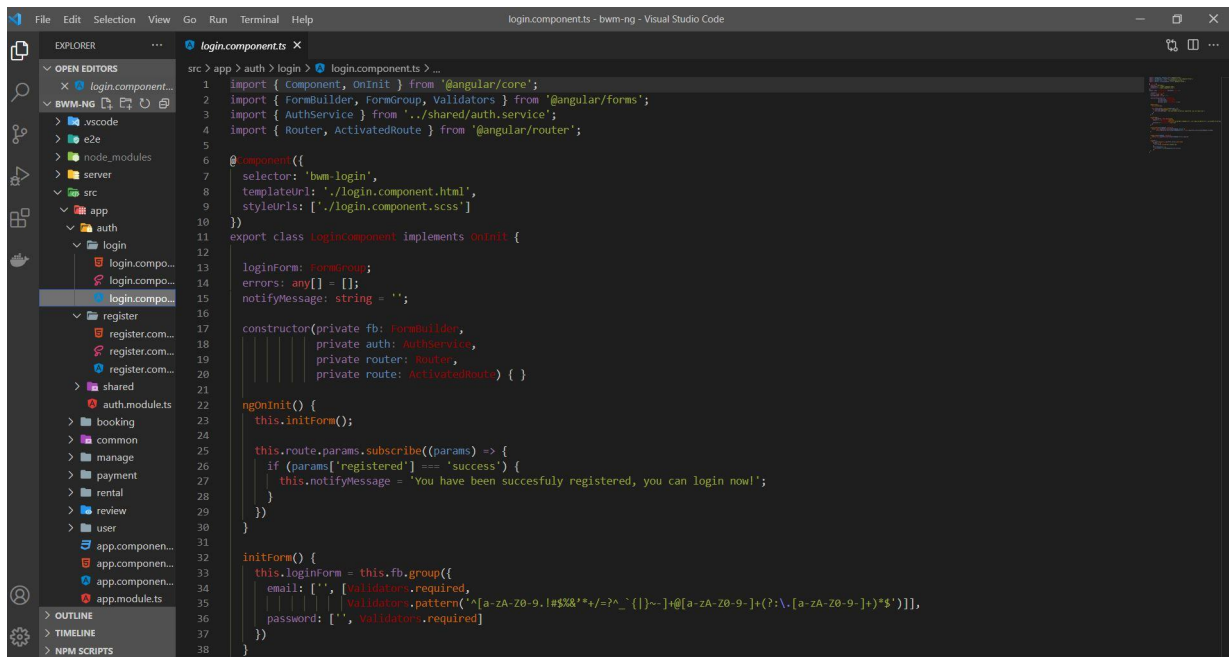
## CODE IMPLEMENTATION

### I. Register Component



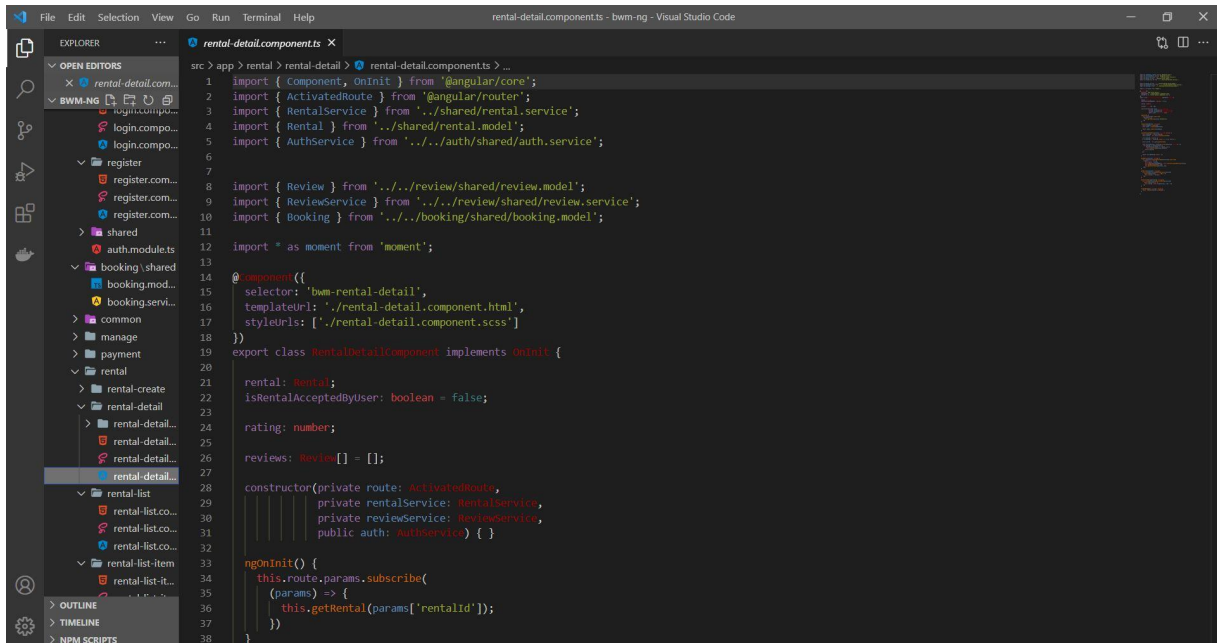
```
src > app > auth > register > register.component.ts > RegisterComponent > register > subscribe() callback
1 import { Component, OnInit } from '@angular/core';
2 import { AuthService } from '../shared/auth.service';
3 import { Router } from '@angular/router';
4
5 @Component({
6   selector: 'bwm-register',
7   templateUrl: './register.component.html',
8   styleUrls: ['./register.component.scss']
9 })
10 export class RegisterComponent implements OnInit {
11
12   formData: any = { username: '' };
13   errors: any[] = [];
14
15   constructor(private auth: AuthService,
16               | private router: Router) { }
17
18   ngOnInit() {
19   }
20
21   register() {
22     this.auth.register(this.formData).subscribe(
23       () => {
24         this.router.navigate(['/login'], {registered: 'success'});
25       },
26       (errorResponse) => {
27         this.errors = errorResponse.error.errors;
28       }
29     )
30   }
31 }
```

### II. Login Component



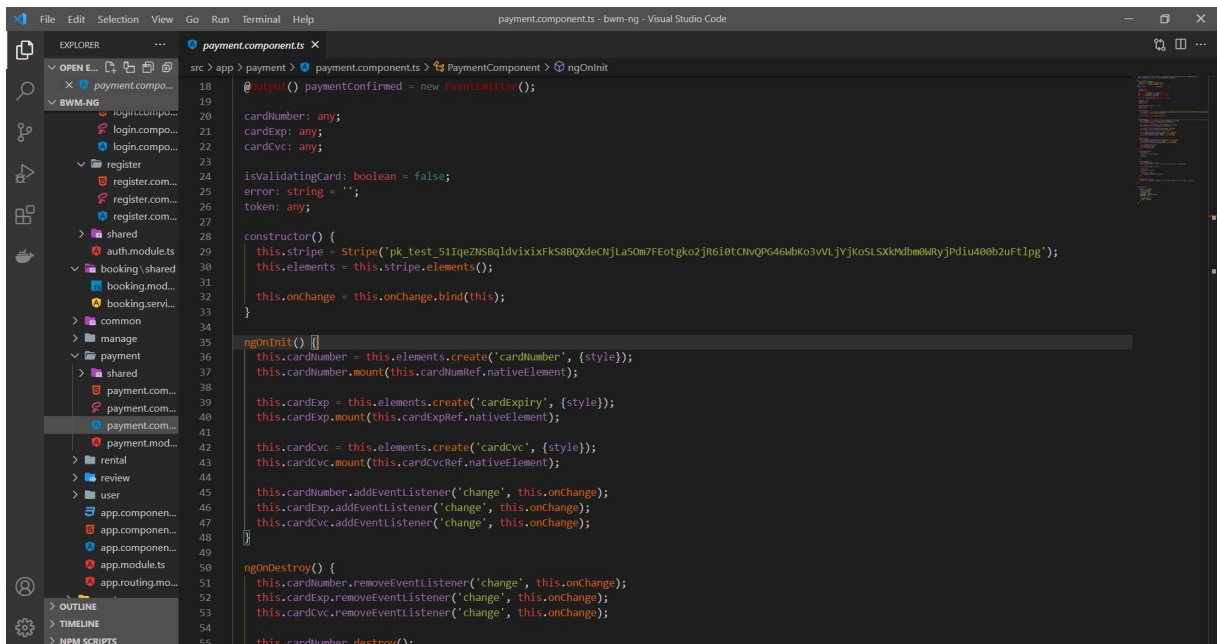
```
src > app > auth > login > login.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { FormBuilder, FormGroup, Validators } from '@angular/forms';
3 import { AuthService } from '../shared/auth.service';
4 import { Router, ActivatedRoute } from '@angular/router';
5
6 @Component({
7   selector: 'bwm-login',
8   templateUrl: './login.component.html',
9   styleUrls: ['./login.component.scss']
10 })
11 export class LoginComponent implements OnInit {
12
13   loginForm: FormGroup;
14   errors: any[] = [];
15   notifyMessage: string = '';
16
17   constructor(private fb: FormBuilder,
18               private auth: AuthService,
19               private router: Router,
20               private route: ActivatedRoute) { }
21
22   ngOnInit() {
23     this.initForm();
24
25     this.route.params.subscribe((params) => {
26       if (params['registered'] === 'success') {
27         this.notifyMessage = 'You have been successfully registered, you can login now!';
28       }
29     })
30   }
31
32   initForm() {
33     this.loginForm = this.fb.group({
34       email: ['', [Validators.required,
35                 Validator.pattern(/^[a-zA-Z0-9.!#$%&'*/=?^_{}|~]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+$/)]],
36       password: ['', Validators.required]
37     })
38 }
```

### III. Rental Component



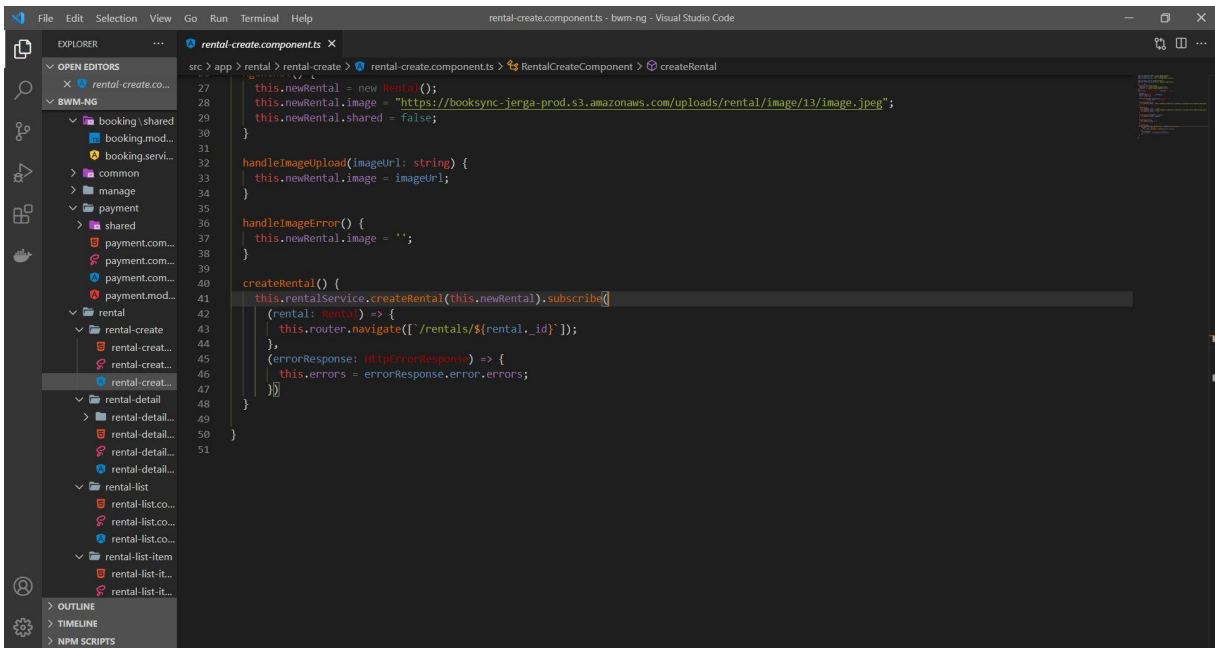
```
src > app > rental > rental-detail > rental-detail.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3 import { RentalService } from '../shared/rental.service';
4 import { Rental } from '../shared/rental.model';
5 import { AuthService } from '../../auth/shared/auth.service';
6
7
8 import { Review } from '../../review/shared/review.model';
9 import { ReviewService } from '../../review/shared/review.service';
10 import { Booking } from '../../booking/shared/booking.model';
11
12 import * as moment from 'moment';
13
14 @Component({
15   selector: 'bwm-rental-detail',
16   templateUrl: './rental-detail.component.html',
17   styleUrls: ['./rental-detail.component.scss']
18 })
19 export class RentalDetailComponent implements OnInit {
20
21   rental: Rental;
22   isRentalAcceptedByUser: boolean = false;
23
24   rating: number;
25
26   reviews: Review[] = [];
27
28   constructor(private route: ActivatedRoute,
29               private rentalService: RentalService,
30               private reviewService: ReviewService,
31               public auth: AuthService) { }
32
33   ngOnInit() {
34     this.route.params.subscribe(
35       (params) => {
36         this.getRental(params['rentalId']);
37       }
38     )
39   }
40 }
```

### IV. Payment Component



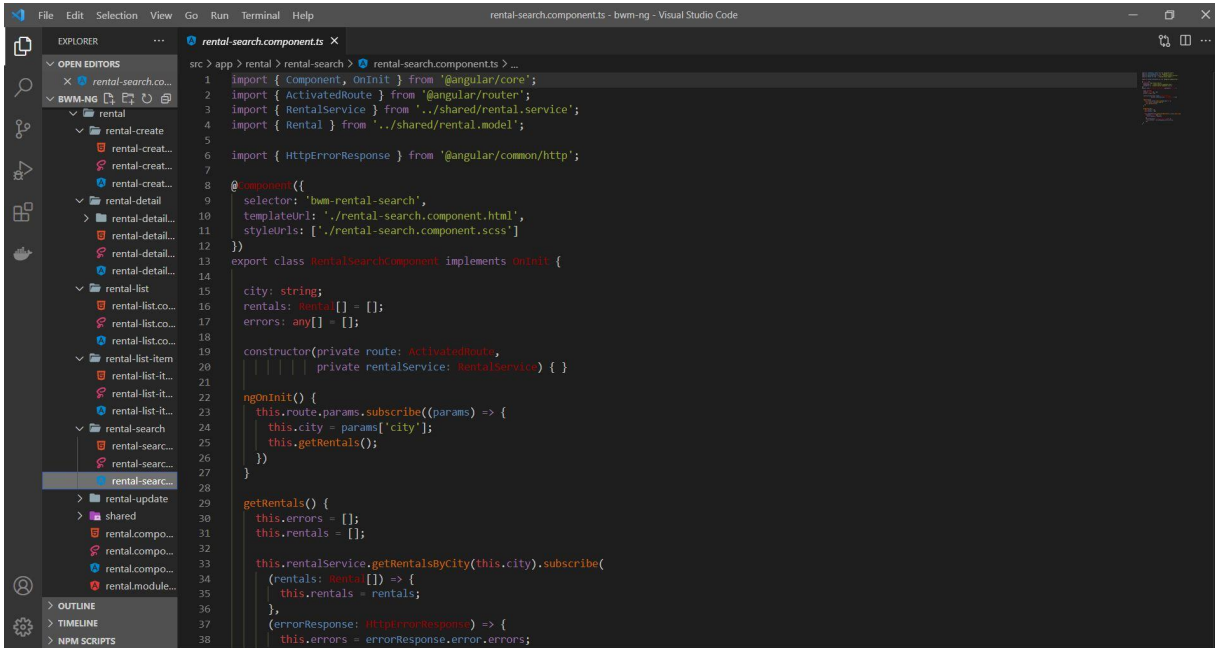
```
src > app > payment > payment.component.ts > PaymentComponent > ngOnInit
18 @Output() paymentConfirmed = new EventEmitter();
19
20 cardNumber: any;
21 cardExp: any;
22 cardCvc: any;
23
24 isValidatingCard: boolean = false;
25 error: string = '';
26 token: any;
27
28 constructor() {
29   this.stripe = Stripe('pk_test_51IqeZNSBqldvixiFks8BQXdeCnJLa5om7FEotgko2jRei0tCNVqG46Wko3vVLjYjkoSLXkPdbmWRyJpdIu400b2uFtlpg');
30   this.elements = this.stripe.elements();
31   this.onChange = this.onChange.bind(this);
32 }
33
34 ngOnInit() {
35   this.cardNumber = this.elements.create('cardNumber', {style});
36   this.cardNumber.mount(this.cardNumRef.nativeElement);
37
38   this.cardExp = this.elements.create('cardExpiry', {style});
39   this.cardExp.mount(this.cardExpRef.nativeElement);
40
41   this.cardCvc = this.elements.create('cardCvc', {style});
42   this.cardCvc.mount(this.cardCvcRef.nativeElement);
43
44   this.cardNumber.addEventListener('change', this.onChange);
45   this.cardExp.addEventListener('change', this.onChange);
46   this.cardCvc.addEventListener('change', this.onChange);
47
48   ngOnDestroy() {
49     this.cardNumber.removeEventListener('change', this.onChange);
50     this.cardExp.removeEventListener('change', this.onChange);
51     this.cardCvc.removeEventListener('change', this.onChange);
52
53     this.cardNumber.destroy();
54
55   }
56 }
```

## V. Create Rental Component



```
src > app > rental > rental-create > RentalCreateComponent > createRental
27
28   this.newRental = new Rental();
29   this.newRental.image = "https://booksync-jerga-prod.s3.amazonaws.com/uploads/rental/image/13/image.jpeg";
30   this.newRental.shared = false;
31 }
32
33 handleImageUpload(imageUrl: string) {
34   this.newRental.image = imageUrl;
35 }
36
37 handleError() {
38   this.newRental.image = '';
39 }
40
41 createRental() {
42   this.rentalService.createRental(this.newRental).subscribe(
43     (rental: Rental) => {
44       this.router.navigate(['/rentals/${rental.id}']);
45     },
46     (errorResponse: HttpResponse) => {
47       this.errors = errorResponse.error.errors;
48     })
49 }
50
51 }
```

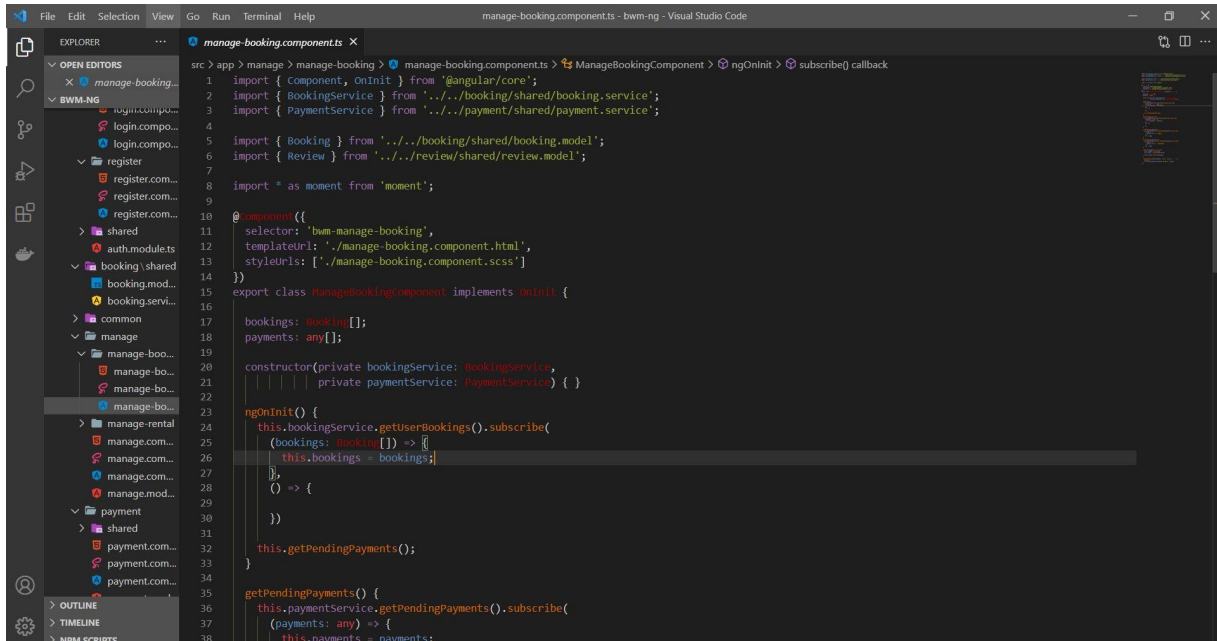
## VI. Search Rental Component



```
src > app > rental > rental-search > rental-search.components.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3 import { RentalService } from '../shared/rental.service';
4 import { Rental } from '../shared/rental.model';
5
6 import { HttpResponse } from '@angular/common/http';
7
8 @Component({
9   selector: 'bwm-rental-search',
10  templateUrl: './rental-search.component.html',
11  styleUrls: ['./rental-search.component.scss']
12 })
13 export class RentalSearchComponent implements OnInit {
14
15   city: string;
16   rentals: Rental[] = [];
17   errors: any[] = [];
18
19   constructor(private route: ActivatedRoute,
20               private rentalService: RentalService) {}
21
22   ngOnInit() {
23     this.route.params.subscribe((params) => {
24       this.city = params['city'];
25       this.getRentals();
26     })
27   }
28
29   getRentals() {
30     this.errors = [];
31     this.rentals = [];
32
33     this.rentalService.getRentalsByCity(this.city).subscribe(
34       (rentals: Rental[]) => {
35         this.rentals = rentals;
36       },
37       (errorResponse: HttpResponse) => {
38         this.errors = errorResponse.error.errors;
39       }
40     );
41   }
42 }
```

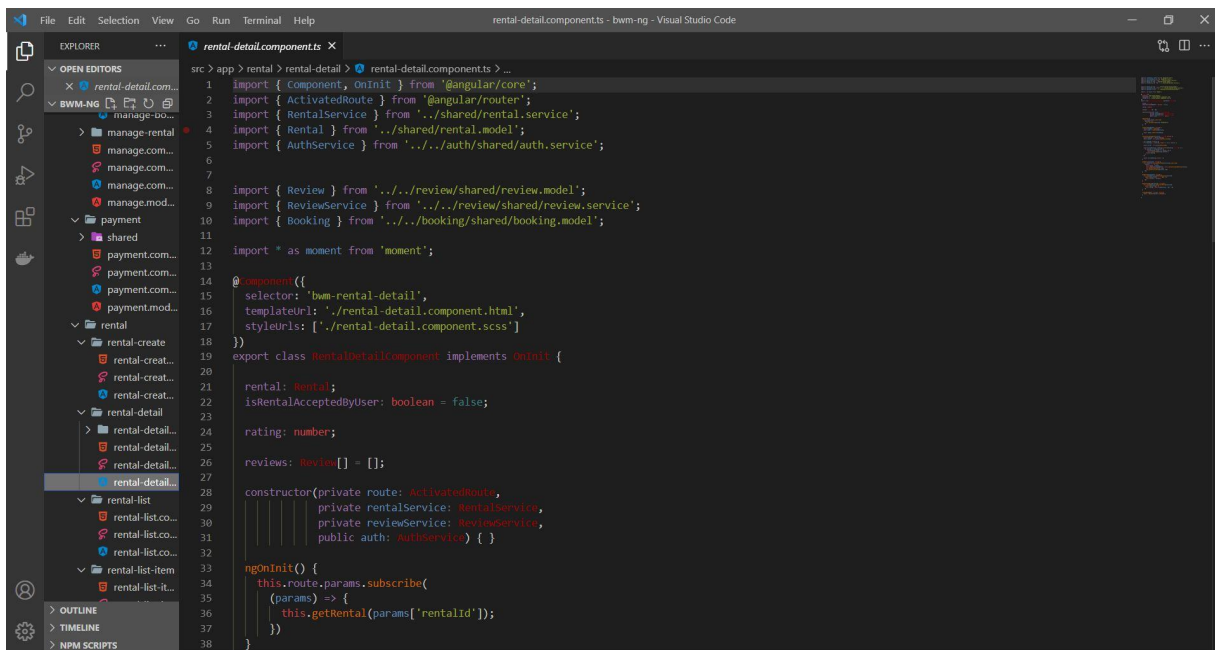


## VII. Manage Booking Component



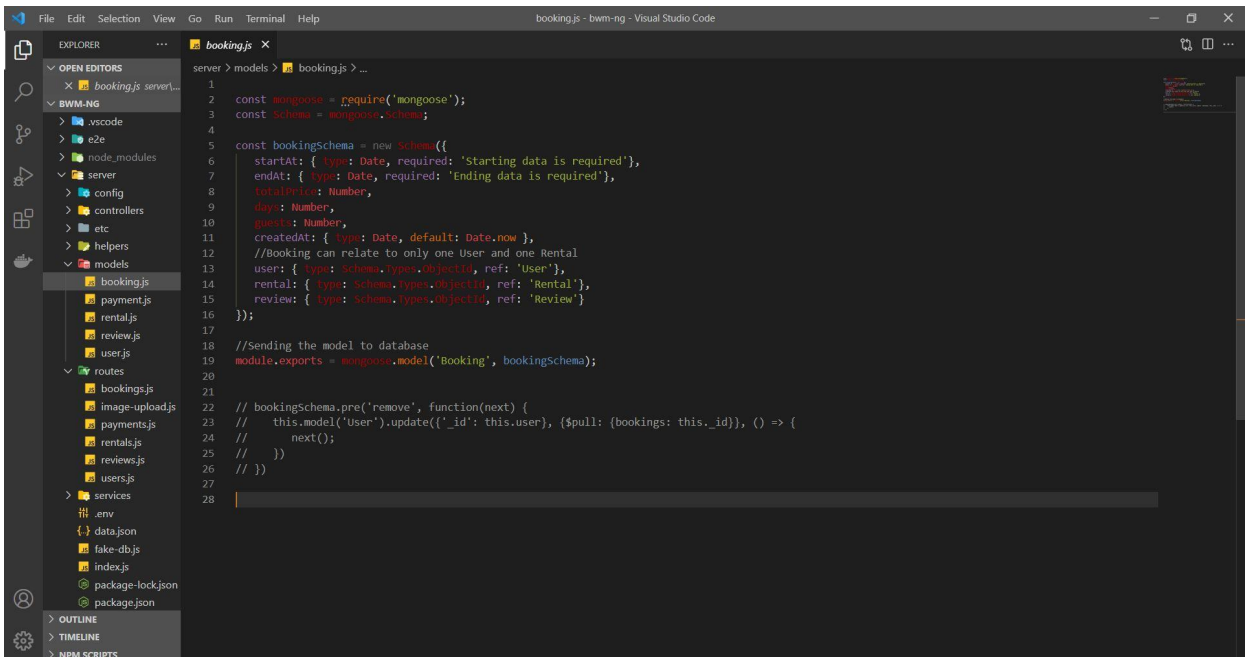
```
1 import { Component, OnInit } from '@angular/core';
2 import { BookingService } from '../../booking/shared/booking.service';
3 import { PaymentService } from '../../payment/shared/payment.service';
4
5 import { Booking } from '../../booking/shared/booking.model';
6 import { Review } from '../../review/shared/review.model';
7
8 import * as moment from 'moment';
9
10 @Component({
11   selector: 'bwm-manage-booking',
12   templateUrl: './manage-booking.component.html',
13   styleUrls: ['./manage-booking.component.scss']
14 })
15 export class ManageBookingComponent implements OnInit {
16
17   bookings: Booking[];
18   payments: any[];
19
20   constructor(private bookingService: BookingService,
21              private paymentService: PaymentService) { }
22
23   ngOnInit() {
24     this.bookingService.getUserBookings().subscribe(
25       (bookings: Booking[]) => {
26         this.bookings = bookings;
27       }
28     );
29     this.getPendingPayments();
30   }
31
32   getPendingPayments() {
33     this.paymentService.getPendingPayments().subscribe(
34       (payments: any) => {
35         this.payments = payments;
36       }
37     );
38   }
39 }
```

## VIII. Rental Detail



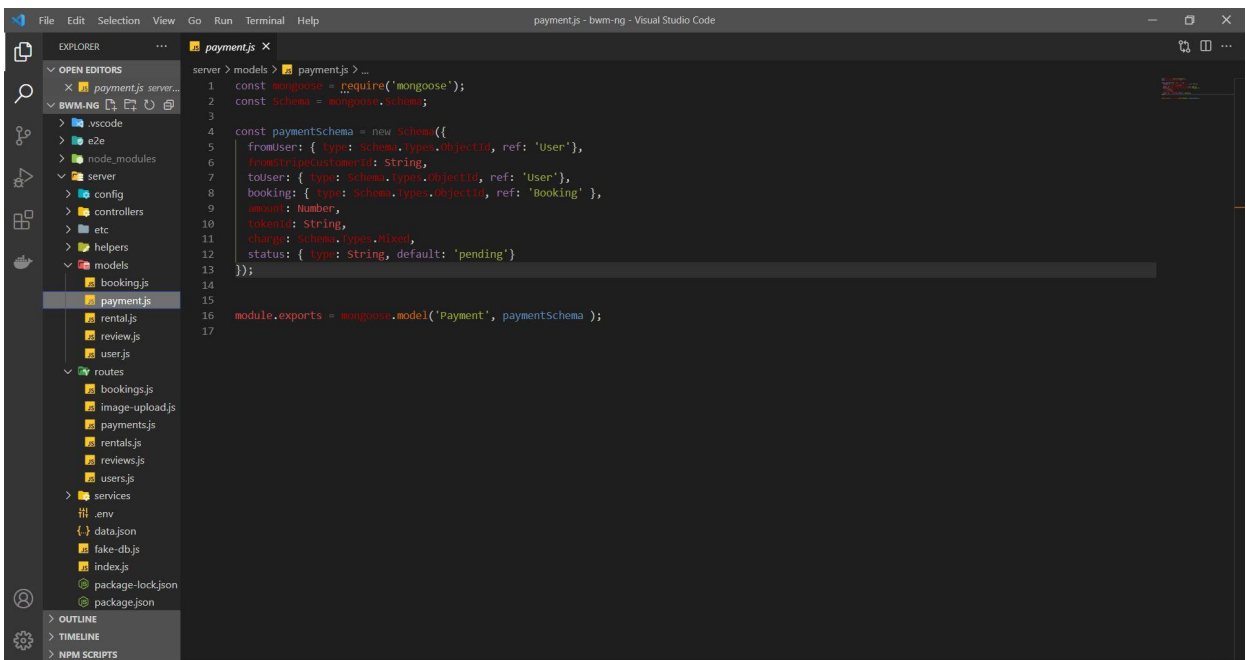
```
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute } from '@angular/router';
3 import { RentalService } from '../shared/rental.service';
4 import { Rental } from '../shared/rental.model';
5 import { AuthService } from '../auth/shared/auth.service';
6
7 import { Review } from '../../review/shared/review.model';
8 import { ReviewService } from '../../review/shared/review.service';
9 import { Booking } from '../../booking/shared/booking.model';
10
11 import * as moment from 'moment';
12
13 @Component({
14   selector: 'bwm-rental-detail',
15   templateUrl: './rental-detail.component.html',
16   styleUrls: ['./rental-detail.component.scss']
17 })
18 export class RentalDetailComponent implements OnInit {
19
20   rental: Rental;
21   isRentalAcceptedByUser: boolean = false;
22
23   rating: number;
24
25   reviews: Review[] = [];
26
27   constructor(private route: ActivatedRoute,
28              private rentalService: RentalService,
29              private reviewService: ReviewService,
30              public auth: AuthService) { }
31
32   ngOnInit() {
33     this.route.params.subscribe(
34       (params) => {
35         this.getRental(params['rentalid']);
36       }
37     );
38   }
39 }
```

## IX. Booking Model



```
server > models > booking.js > ...
1  const mongoose = require('mongoose');
2  const Schema = mongoose.Schema;
3
4
5  const bookingSchema = new Schema({
6    startAt: { type: Date, required: 'Starting data is required'},
7    endAt: { type: Date, required: 'Ending data is required'},
8    totalPrice: Number,
9    days: Number,
10   guests: Number,
11   createdAt: { type: Date, default: Date.now },
12   //Booking can relate to only one User and one Rental
13   user: { type: Schema.Types.ObjectId, ref: 'User' },
14   rental: { type: Schema.Types.ObjectId, ref: 'Rental' },
15   review: { type: Schema.Types.ObjectId, ref: 'Review' }
16 });
17
18 //Sending the model to database
19 module.exports = mongoose.model('Booking', bookingSchema);
20
21
22 // bookingSchema.pre('remove', function(next) {
23 //   this.model('User').update({'_id': this.user}, {$pull: {bookings: this._id}}, {} => {
24 //     next();
25 //   })
26 // })
27
28
```

## X. Payment Model



```
server > models > payment.js > ...
1  const mongoose = require('mongoose');
2  const Schema = mongoose.Schema;
3
4  const paymentSchema = new Schema({
5    fromUser: { type: Schema.Types.ObjectId, ref: 'User' },
6    fromStripeCustomerId: String,
7    toUser: { type: Schema.Types.ObjectId, ref: 'User' },
8    booking: { type: Schema.Types.ObjectId, ref: 'Booking' },
9    amount: Number,
10   taxamt: String,
11   changed: Schema.Types.Mixed,
12   status: { type: String, default: 'pending' }
13 });
14
15
16 module.exports = mongoose.model('Payment', paymentSchema);
17
```

## XI. Rental Model

```

server > models > rental.js > ...
4 const rentalSchema = new Schema({
5   title: { type: String, required: true, max: [128, 'Too long, max is 128 characters']},
6   city: { type: String, required: true, lowercase: true },
7   street: { type: String, required: true, min: [4, 'Too short, min is 4 characters']},
8   category: { type: String, required: true, lowercase: true },
9   image: { type: String, required: true },
10  bedrooms: Number,
11  shared: Boolean,
12  description: { type: String, required: true },
13  phone: {
14    type: String,
15    min: [4, 'Too short, min is 4 characters'],
16    max: [32, 'Too long, max is 32 characters']
17  },
18  dailyRate: Number,
19  createdAt: { type: Date, default: Date.now },
20  user: { type: Schema.Types.ObjectId, ref: 'User' },
21  bookings: [{ type: Schema.Types.ObjectId, ref: 'Booking' }]
22 });
23
24
25 module.exports = mongoose.model('rental', rentalSchema);
26

```

## X. User Model

```

server > models > user.js > userSchema > password
1 const bcrypt = require('bcrypt');
2 const mongoose = require('mongoose');
3 const Schema = mongoose.Schema;
4
5 const userSchema = new Schema({
6   username: {
7     type: String,
8     min: [4, 'Too short, min is 4 characters'],
9     max: [32, 'Too long, max is 32 characters']
10  },
11  email: {
12    type: String,
13    min: [4, 'Too short, min is 4 characters'],
14    max: [32, 'Too long, max is 32 characters'],
15    unique: true,
16    lowercase: true,
17    required: 'Email is required',
18    match: [/^[\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/]
19  },
20  password: {
21    type: String,
22    min: [4, 'Too short, min is 4 characters'],
23    max: [32, 'Too long, max is 32 characters'],
24    required: 'Password is required'
25  },
26  stripeCustomerId: String,
27  revenue: Number,
28  rentals: [{ type: Schema.Types.ObjectId, ref: 'Rental' }],
29  bookings: [{ type: Schema.Types.ObjectId, ref: 'Booking' }]
30 });
31
32 userSchema.methods.hasSamePassword = function(requestedPassword) {
33   return bcrypt.compareSync(requestedPassword, this.password);
34 }
35
36
37
38 userSchema.pre('save', function(next) {

```

## XI. User Controller

```

server > controllers > userjs > authMiddleware > authMiddleware
1 const User = require('../models/user');
2 const { normalizeErrors } = require('../helpers/mongoose');
3 const jwt = require('jsonwebtoken');
4 const config = require('../config');
5
6 exports.getUser = function(req, res) {
7   const requestedUserId = req.params.id;
8   const user = res.locals.user;
9
10  if (requestedUserId === user.id) {
11    User.findById(requestedUserId, function(err, foundUser) {
12      if (err) {
13        return res.status(422).send({errors: normalizeErrors(err.errors)});
14      }
15
16      return res.json(foundUser);
17    })
18  } else {
19    User.findById(requestedUserId)
20      .select('-revenue -stripeCustomerId -password')
21      .exec(function(err, foundUser) {
22        if (err) {
23          return res.status(422).send({errors: normalizeErrors(err.errors)});
24        }
25
26        return res.json(foundUser);
27      })
28  }
29 }
30
31 exports.auth = function(req, res) {
32   const { email, password } = req.body;
33
34   if (!password || !email) {

```

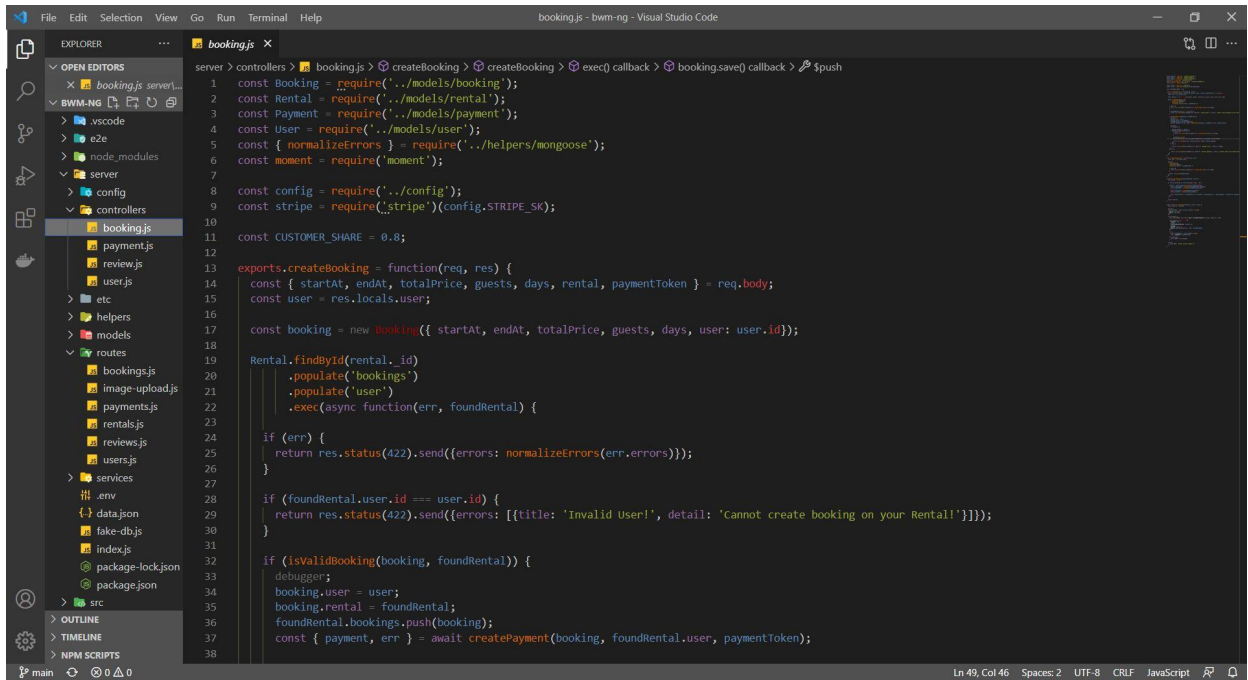
## XII. Payment Controller

```

server > controllers > paymentjs > stripe
1 const Rental = require('../models/rental');
2 const Payment = require('../models/payment');
3
4 const { normalizeErrors } = require('../helpers/mongoose');
5
6 const config = require('../config');
7 const stripe = require('stripe')('sk_test_511qeZNSBqldvixinnXAF1b0KX0RP5367Dk6fT0ifZcgVYsg0JR1Ca2EcBwLEq5vPo7uMsxpk17PLLpsqchDg00r1GfAqna');
8
9 exports.getPendingPayments = function(req, res) {
10   const user = res.locals.user;
11
12   Payment
13     .where({toUser: user})
14     .populate({
15       path: 'booking',
16       populate: {path: 'rental'}
17     })
18     .populate('fromUser')
19     .exec(function(err, foundPayments) {
20       if (err) {
21         return res.status(422).send({errors: normalizeErrors(err.errors)});
22       }
23
24       return res.json(foundPayments);
25     })
26 }
27
28 exports.confirmPayment = function(req, res) {
29   const payment = req.body;
30   const user = res.locals.user;
31
32   debugger;
33   Payment.findById(payment._id)
34     .populate('toUser')
35     .populate('booking')
36     .exec(async function(err, foundPayment) {

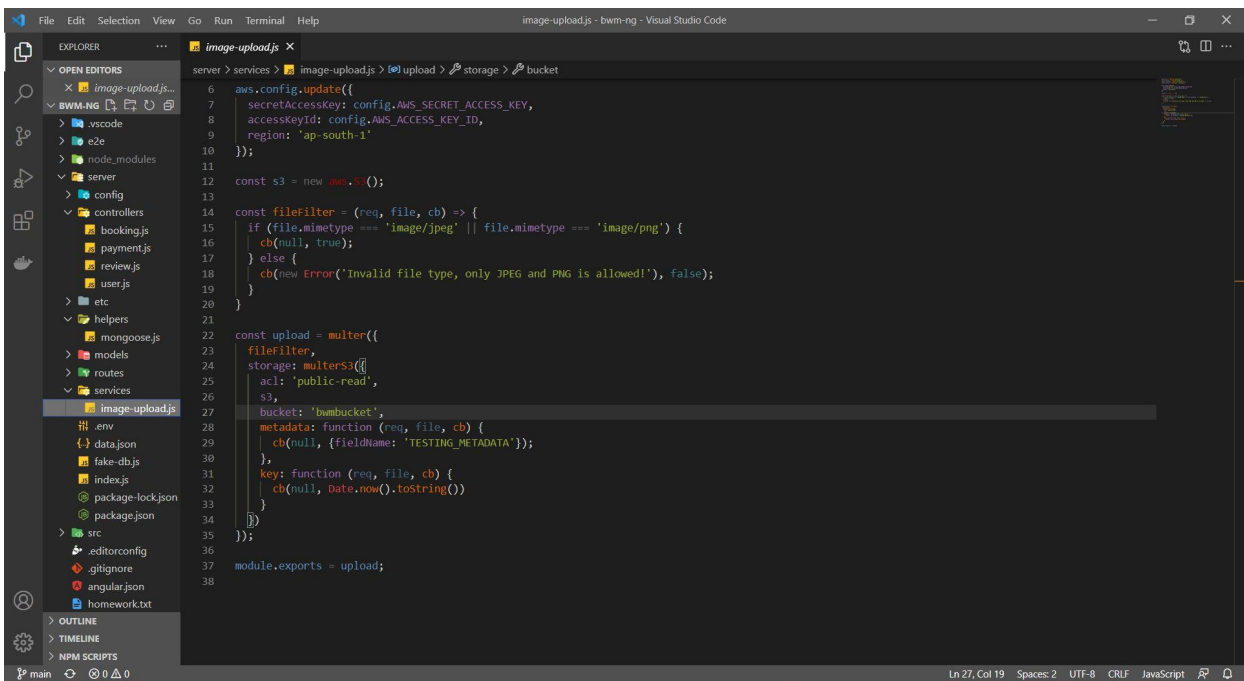
```

### XIII. Booking Controller



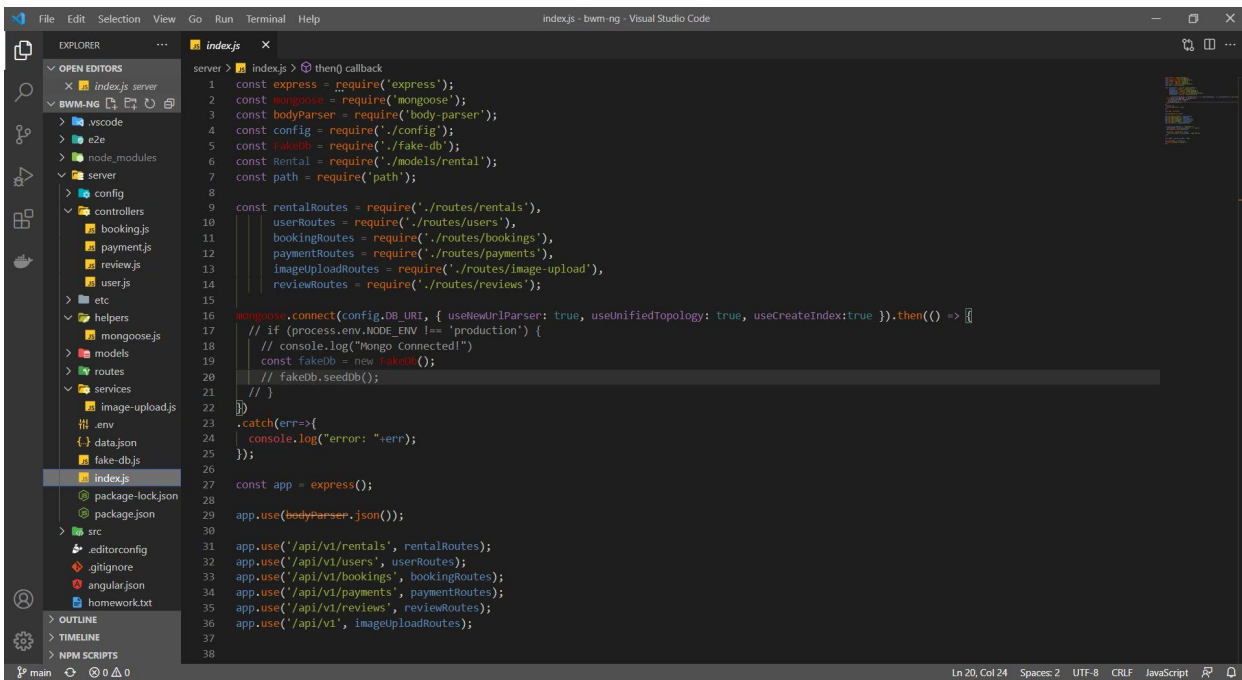
```
server > controllers > booking.js > createBooking > createBooking > exec() callback > booking.save() callback > $push
1 const Booking = require('../models/booking');
2 const Rental = require('../models/rental');
3 const Payment = require('../models/payment');
4 const User = require('../models/user');
5 const { normalizeErrors } = require('../helpers/mongoose');
6 const moment = require('moment');
7
8 const config = require('../config');
9 const stripe = require('stripe')(config.STRIPE_SK);
10
11 const CUSTOMER_SHARE = 0.8;
12
13 exports.createBooking = function(req, res) {
14   const { startAt, endAt, totalPrice, guests, days, rental, paymentToken } = req.body;
15   const user = res.locals.user;
16
17   const booking = new Booking({ startAt, endAt, totalPrice, guests, days, user: user.id });
18
19   Rental.findById(rental.id)
20     .populate('bookings')
21     .populate('user')
22     .exec(async function(err, foundRental) {
23
24       if (err) {
25         return res.status(422).send({errors: normalizeErrors(err.errors)});
26       }
27
28       if (foundRental.user.id === user.id) {
29         return res.status(422).send({errors: [{title: 'Invalid user!', detail: 'Cannot create booking on your Rental!'}]});
30       }
31
32       if (isValidBooking(booking, foundRental)) {
33         debugger;
34         booking.user = user;
35         booking.rental = foundRental;
36         foundRental.bookings.push(booking);
37         const { payment, err } = await createPayment(booking, foundRental.user, paymentToken);
38
```

### XIV. Image Upload Service



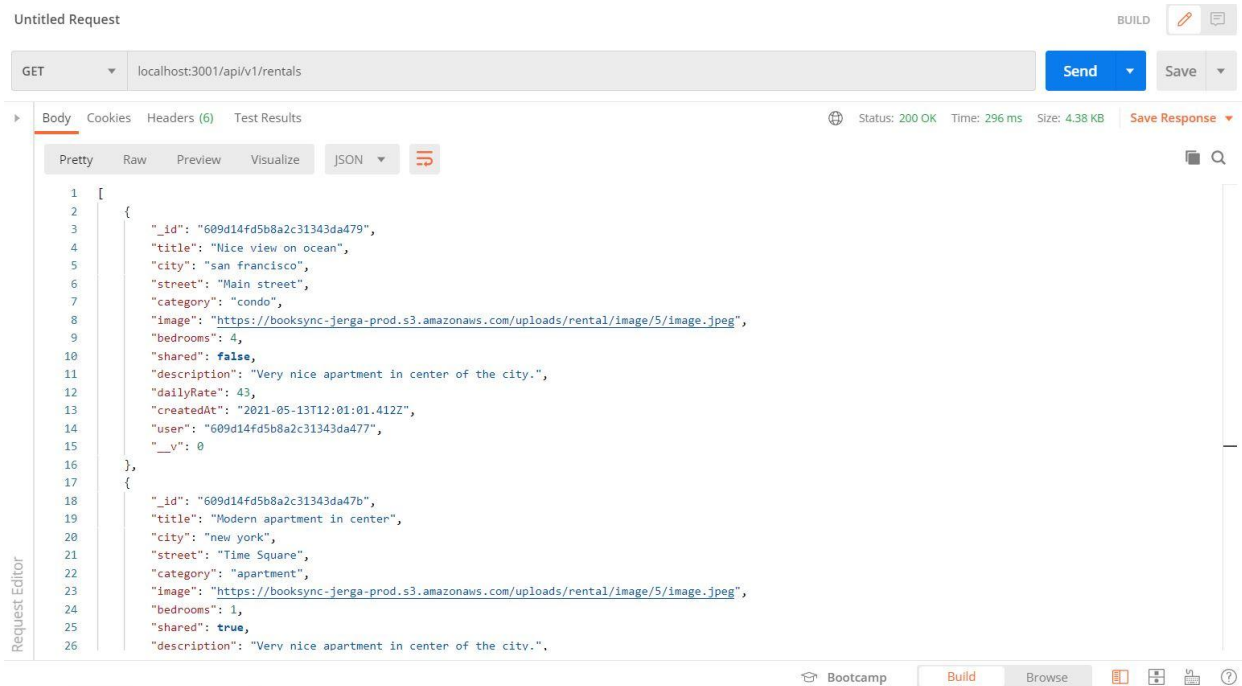
```
server > services > image-uploads.js > upload > storage > bucket
6 aws.config.update({
7   secretAccessKey: config.AWS_SECRET_ACCESS_KEY,
8   accessKeyId: config.AWS_ACCESS_KEY_ID,
9   region: 'ap-south-1'
10 });
11
12 const s3 = new AWS.S3();
13
14 const fileFilter = (req, file, cb) => {
15   if (file.mimetype === 'image/jpeg' || file.mimetype === 'image/png') {
16     cb(null, true);
17   } else {
18     cb(new Error('Invalid file type, only JPEG and PNG is allowed!'), false);
19   }
20 }
21
22 const upload = multer({
23   fileFilter,
24   storage: multerS3({
25     acl: 'public-read',
26     s3,
27     bucket: 'bumbucket',
28     metadata: function (req, file, cb) {
29       cb(null, {fieldName: 'TESTING_METADATA'});
30     },
31     key: function (req, file, cb) {
32       cb(null, Date.now().toString());
33     }
34   })
35 });
36
37 module.exports = upload;
38
```

## XV. Server main implementation



```
server > index.js > then() callback
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const bodyParser = require('body-parser');
4  const config = require('./config');
5  const fakeDb = require('./fake-db');
6  const Rental = require('./models/rental');
7  const path = require('path');
8
9  const rentalRoutes = require('./routes/rentals'),
10     userRoutes = require('./routes/users'),
11     bookingRoutes = require('./routes/bookings'),
12     paymentRoutes = require('./routes/payments'),
13     imageUploadRoutes = require('./routes/image-upload'),
14     reviewRoutes = require('./routes/reviews');
15
16 mongoose.connect(config.DB_URI, { useNewUrlParser: true, useUnifiedTopology: true, useNewUrlParser: true }).then(() => {
17   // if (process.env.NODE_ENV !== 'production') {
18   //   console.log("Mongo Connected")
19   //   const fakeDb = new FakeDb();
20   //   fakeDb.seedDb();
21   // }
22 })
23 .catch(err=>{
24   console.log("error: "+err);
25 });
26
27 const app = express();
28
29 app.use(bodyParser.json());
30
31 app.use('/api/v1/rentals', rentalRoutes);
32 app.use('/api/v1/users', userRoutes);
33 app.use('/api/v1/bookings', bookingRoutes);
34 app.use('/api/v1/payments', paymentRoutes);
35 app.use('/api/v1/reviews', reviewRoutes);
36 app.use('/api/v1', imageUploadRoutes);
37
38
```

## XVI. Rental Response



Untitled Request

GET localhost:3001/api/v1/rentals

Status: 200 OK Time: 296 ms Size: 4.38 KB

Body

```
1  [
2    {
3      "_id": "609d14fd5b8a2c31343da479",
4      "title": "Nice view on ocean",
5      "city": "san francisco",
6      "street": "Main street",
7      "category": "condo",
8      "image": "https://booksync-jerga-prod.s3.amazonaws.com/uploads/rental/image/5/image.jpeg",
9      "bedrooms": 4,
10     "shared": false,
11     "description": "Very nice apartment in center of the city.",
12     "dailyRate": 43,
13     "createdAt": "2021-05-13T12:01:01.412Z",
14     "user": "609d14fd5b8a2c31343da477",
15     "__v": 0
16   },
17   {
18     "_id": "609d14fd5b8a2c31343da47b",
19     "title": "Modern apartment in center",
20     "city": "new york",
21     "street": "Time Square",
22     "category": "apartment",
23     "image": "https://booksync-jerga-prod.s3.amazonaws.com/uploads/rental/image/5/image.jpeg",
24     "bedrooms": 1,
25     "shared": true,
26     "description": "Very nice apartment in center of the city."
27   }
28 ]
```

## **CHAPTER-10**

### **FUTURE SCOPE & FUTURE ENHANCEMENT**

Some of the advances that can be done in our project are:

1. Frontend modifications
2. Dockerizing for scaling
3. Deployment to a domain
4. Payment-gateway implementation.
5. CI/CD implementation.

## **CHAPTER-11**

### **CONCLUSION**

We have effectively executed our project named 'BOOK WITH ME HOTEL BOOKING APPLICATION USING ANGULAR and NODE.JS' utilizing angular framework till what we have chosen to work in phase 1. Now, in the phase 2 we will:

- We will host the platform on AWS EC2 a scalable cloud service.
- We will implement CI/CD using Travis CI.
- We will Dockerize the project into containers using Docker for scalability.
- We will try to implement microservice architecture for scalability.



## **CHAPTER-12**

### **BIBLIOGRAPHY/REFERENCES**

- [1] Angular Documentation Accessed On: 15 May, 2021 [Online] Available:  
<https://www.angular.io>
- [2] Node.js Documentation Accessed On: 15 May, 2021 [Online] Available:  
<https://nodejs.org/en/docs/>
- [3] Bootstrap Documentation Accessed On: 15 May, 2021 [Online] Available:  
<https://www.getbootstrap.com>
- [4] Debugging and Testing Accessed On: 15 May, 2021 [Online] Available:  
<https://www.stackoverflow.com>

## akash\_marwah\_major

### ORIGINALITY REPORT

<b>10</b> %	<b>1</b> %	<b>0</b> %	<b>9</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	Submitted to Jaypee University of Information Technology Student Paper	<b>9</b> %
<b>2</b>	www.informit.com Internet Source	<b>&lt;1</b> %
<b>3</b>	Submitted to University of Hong Kong Student Paper	<b>&lt;1</b> %

Exclude quotes  On

Exclude matches  Off

Exclude bibliography  On