

FULL STACK DEVELOPMENT

Restaurant Website

Summer Internship Report Submitted in partial fulfilment

of the requirement for undergraduate degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

A.kartheek

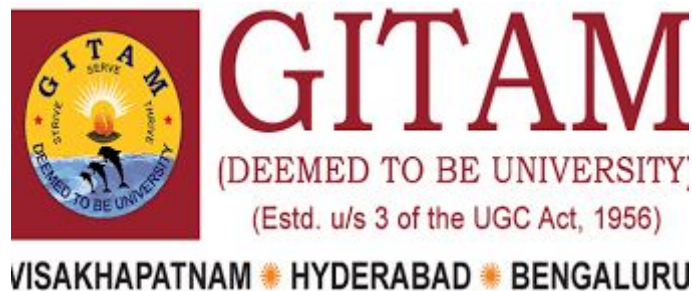
221710301004

<https://github.com/ak950/fullstack>

Under the Guidance of

MS. G. Mounika

Assistant Professor



Department Of Computer Science and Engineering

GITAM School of Technology

GITAM (Deemed to be University)

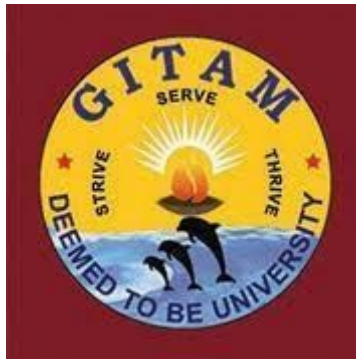
Hyderabad-502329

July 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF TECHNOLOGY
GITAM

(Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS



DECLARATION

I submitted this industrial training work entitled “**Restaurant Website**” to GITAM (Deemed to Be University), Hyderabad in partial fulfilment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science and Engineering**”. I declare that it was carried out independently by me under the guidance of **(MS. G. Mounika)** , Asst. Professor, GITAM(Deemed to Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place:Hyderabad

Date:

Name And signature of the candidate

A.kartheek

(221710301004)

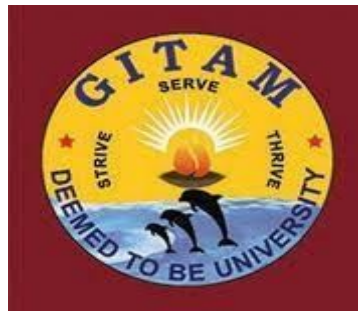
CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS



CERTIFICATE

This is to certify that the Industrial Training Report entitled - "**Restaurant Website**" is being submitted by **A.kartheek(221710301004)**, submitted in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology in Computer Science and Engineering**.

Guided by

Ms. G. Mounika

Head of the Department

Dr. Phani Kumar

Professor & HOD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS



CERTIFICATE



INTERNSHIP CERTIFICATE

This certificate is presented to

A.Kartheek

in recognition of his/her hardwork and dedication in completing **Full Stack Development** Internship project in

Restaurant Website

from 25th May, 2020 to 21st July, 2020 under the sponsorship of Dhyanahitha Educational Society in accordance with all requirements of graduation by GITAM (Deemed to be University), Hyderabad.



ANIL KUMAR M
PROJECT MANAGER
DHYANAHITHA EDUCATIONAL SOCIETY





SURESH N
DIRECTOR OF OPERATIONS
DHYANAHITHA EDUCATIONAL SOCIETY

ACKNOWLEDGEMENT

My project would not have been successful without the help of several people. I would like to thank the personalities who were part of my project in numerous ways, those who gave me outstanding support from the birth of the project.

I would like to thank our honorable Pro-Vice-Chancellor, **Prof. N. Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of my project.

I would like to thank respected **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the project.

I would like to thank respected **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in the completion of this project.

I would like to thank **Ms. G. Mounika**, Assistant Professor in the Computer Science Department for the esteemed guidance, moral support and invaluable advice provided by her for the success of the project.

I would like to thank my parents and friends who extended their help, encouragement and moral support either directly or indirectly in my project work.

Sincerely,

A.kartheek

(221710301004)

ABSTRACT

This project is a web application to a restaurant website. This system provides a reservation table by the customer and explores the menu through the system online. Main objective to build the system is to provide a reservation table to the customer. With the online system, reserving tables will become easier and systematic to replace traditional systems that are still using paper. Furthermore, this system is applicable anytime and where also to the customer.

TABLE OF CONTENTS

SNO	TITLE	PGNO
1	INTRODUCTION	1-2
	1.1 FRONT END	1
	1.2 BACKEND	2
2	FULL STACK DEVELOPMENT	3-6
	2.1 WEB DEVELOPMENT	4
	2.2 WEB SITE	5
	2.3 WEB PAGE	6
3	STEPS TO CREATE A WEBSITE	7-24
	3.1 UI DEVELOPMENT	8
	3.1.1 HTML	8
	3.1.2 CSS	10
	3.1.3 JAVASCRIPT	14
	3.1.4 BOOTSTRAP	15
	3.2 SERVER SIDE AND CLIENT SIDE SCRIPTING	16
	3.2.1 CLIENT SIDE SCRIPTING	16
	3.2.2 SERVER SIDE SCRIPTING	17
	3.2.3 SCRIPTING LANGUAGES	18
	3.3 DATABASE	21

	3.4 MY SQL	24
4	REQUIREMENTS	25
	4.1 HARDWARE REQUIREMENTS	25
	4.2 SOFTWARE REQUIREMENTS	25
5	DATA FLOW DIAGRAMS	26-28
	5.1 DFD-1	26
	5.2 DFD-2	26
	5.3 DFD-3	27
	5.4 USE CASE DIAGRAM	27
	5.5 CLASS DIAGRAM	28
6	PROJECT	29-31
	6.1 SCREENSHOTS	29
	6.1.1 BOOKING TABLE	30
	6.1.2 FEEDBACK TABLE	31
7	MODULES ASSIGNED	32
	7.1 BOOKING PAGE	32
	7.2 NODE JS	32
	7.3 MYSQL	32
8	CODES AND OUTPUTS	33-64
9	MAINTENANCE	65

10	FUTURE SCOPE AND ENHANCEMENT	66
11	CONCLUSION	67

List of Figures

Figure 3.1	steps to create a website	07
Figure 3.1.2.1	syntax of css	10
Figure 3.1.3.1	javascript	14
Figure 3.2	server and client side scripting	16
Figure 3.2.1.1	client side scripting	17
Figure 3.2.2.2	server side scripting	18
Figure 3.2.3.1	uses of nodejs	19
Figure 3.4	sql commands	24
Figure 5.1	context level diagram	26
Figure 5.2	DFD-1	26
Figure 5.3	DFD-2	27
Figure 5.4	use case diagram	27
Figure 5.5	class diagram	28
Figure 6.1	booking table	30
Figure 6.2	feedback table	31
Figure 8.1	connecting to database	58
Figure 8.2	checking for local host port number	58
Figure 8.3	output for booking table	59
Figure 8.4	output for feedback table	60

Figure 8.5	creating database and tables	61
Figure 8.6	output for booking table data	61
Figure 8.7	output for feedback table data	62
Figure 8.8	wrong with inserting into booking table	63
Figure 8.9	wrong with inserting into feedback table	64

CHAPTER – 1

INTRODUCTION

Full Stack Development: It refers to the development of both front end (client side) and back end(server side) portions of web application. Full stack web developers have the ability to design complete web applications and websites. They work on the frontend, backend, database and debugging of web applications or websites.

1.1. FRONT-END

Front end development is mostly focused on what some may coin the "client side" of development. Front end developers will be engaged in analyzing code, design, and debugging applications along with ensuring a seamless user experience. You manage what people first see in their browser. As a front end developer you are responsible for the look, feel and ultimately design of the site.

Front end languages include HTML, CSS, and Javascript. In order to be a front end developer (sometimes even called a Javascript developer) you do not need back end development skills. Sites created by front end developers won't interact with information stored on a database in order to be functional.

1.2. BACK-END

Back end Development refers to the server side of development where you are primarily focused on how the site works. Making updates and changes in addition to monitoring functionality of the site will be your primary responsibility. This type of web development usually consists of three parts: a server, an application, and a database. Code written by back end developers is what communicates the database information to the browser. Anything you can't see easily with the eye such as databases and servers is the work of a back end developer. Back end developer positions are often called programmers or web developers.

Back End developers are working with databases, servers, an application programming interface (API) that creates a structure for component interaction and

the integration of all these processes. Back End developers are using Back End programming languages such as Ruby, Python, PHP, Java,node js Microsoft .

Back end developers are most focused on a site's responsiveness and speed. These languages are used to create dynamic sites which are different from static sites in that these types of websites store database information. Content on the site is constantly changing and updating. Examples of dynamic sites include Facebook, Twitter, and Google Maps.

CHAPTER – 2

FULL STACK DEVELOPMENT

A full-stack developer is a web developer or engineer who works with both the front and back ends of a website or application meaning they can tackle projects that involve databases, building user-facing websites, or even work with clients during the planning phase of projects.

You still may be wondering what a full stack developer is? Well here are some of the skills needed for someone to be considered a full stack developer:

One or more back end languages, HTML, CSS and JavaScript Specialization in one particular back end programming language like Ruby, PHP, or Python.

Rather than having to develop complex proprietary code every time for creating different websites, frameworks have become popular resources to make many processes more efficient and convenient. Libraries like jQuery are extremely popular for front-end developers using javascript, as they can implement various functions that other developers have already cultivated and tested.

Javascript frameworks like AngularJS and emberJS solve many of the challenges faced by front-end developers by developing conventions that can easily be implemented with any website.

On the backend, there are frameworks like Rails for the programming language of Ruby, Django for Python, and CakePHP for working with PHP.

The main purpose of frameworks is to make a developer's job easier by developing a set of conventions that can be adopted for many of the different processes involved in creating a website from how information is displayed to how it is stored and accessed in the database.

2.1 WEB – DEVELOPMENT

Web development refers to building, creating, and maintaining websites. It includes aspects such as web design, web publishing, web programming, and database management. There are three kinds of web developer specialization: front-end developer, back-end-developer, full-stack-developer.

While the terms "web developer" and "web designer" are often used synonymously, they do not mean the same thing. Technically, a web designer only designs website interfaces using HTML and CSS. A web developer may be involved in designing a website, but may also write web scripts in languages such as PHP and ASP. Additionally, a web developer may help maintain and update a database used by a dynamic website.

Web development includes many types of web content creation. Some examples include hand coding web pages in a text editor, building a website in a program like Dreamweaver, and updating a blog via a blogging website. In recent years, content management systems like WordPress, Drupal, and Joomla have also become popular means of web development. These tools make it easy for anyone to create and edit their own website using a web-based interface.

While there are several methods of creating websites, there is often a trade-off between simplicity and customization. Therefore, most large businesses do not use content management systems, but instead have a dedicated Web development team that designs and maintains the company's website(s). Small organizations and individuals are more likely to choose a solution like WordPress that provides a basic website template and simplified editing tool.

2.2 WEB – SITE

A Website is a related collection of World Wide Web (WWW) files that includes a beginning file called a home page. A company or an individual tells you how to get to their Website by giving you the address of their home page. From the home page, you can get to all the other pages on their site. For example, the Web site for IBM has the home page address of <http://www.ibm.com>. (The home page address actually includes a specific file name like `index.html` but, as in IBM's case, when a standard default name is set up, users don't have to enter the file name.) IBM's home page address leads to thousands of pages. (But a Web site can also be just a few pages.)

Websites have many functions and can be used in various fashions; a website can be a personal website, a commercial website for a company, a government website or a non-profit organization website. Websites are typically dedicated to a particular topic or purpose, ranging from entertainment and social networking to providing news and education. All publicly accessible websites collectively constitute the World Wide Web, while private websites, such as a company's website for its employees, and are typically a part of an intranet.

Web pages, which are the building blocks of websites, are documents, typically composed in plain text interspersed with formatting instructions of Hypertext Markup Language (HTML, XHTML). They may incorporate elements from other websites with suitable markup anchors. Web pages are accessed and transported with the Hypertext Transfer Protocol (HTTP), which may optionally employ encryption (HTTP Secure, HTTPS) to provide security and privacy for the user. The user's application, often a web browser, renders the page content according to its HTML markup instructions onto a display terminal.

Hyperlinking between web pages conveys to the reader the site structure and guides the navigation of the site, which often starts with a home page containing a

directory of the site web content. Some websites require user registration or subscription to access content.

2.3 WEB – PAGE

A Web page is a document for the World Wide Web that is identified by a unique uniform resource locator (URL).

A Web page can be accessed and displayed on a monitor or mobile device through a Web browser . The data found in a Web page is usually in HTML or XHTML format. The Web pages usually also contain other resources such as style sheets, scripts and images for presentation. Users may be able to navigate to other pages through hypertext links. Web browsers coordinate the various web resource elements for the written web page, such as style sheets, scripts, and images, to present the web page. Typical web pages provide hypertext that includes a navigation bar or a sidebar menu to other web pages via hyperlinks, often referred to as links.

On a network, a web browser can retrieve a web page from a remote web server. On a higher level, the web server may restrict access to only a private network such as a corporate intranet or it provides access to the World Wide Web. On a lower level, the web browser uses the Hypertext Transfer Protocol (HTTP) to make such requests.

A static web page is delivered exactly as stored, as web content in the web server's file system, while a dynamic web page is generated by a web application that is driven by server- side software or client-side scripting. Dynamic website pages help the browser (the client) to enhance the web page through user input to the server.

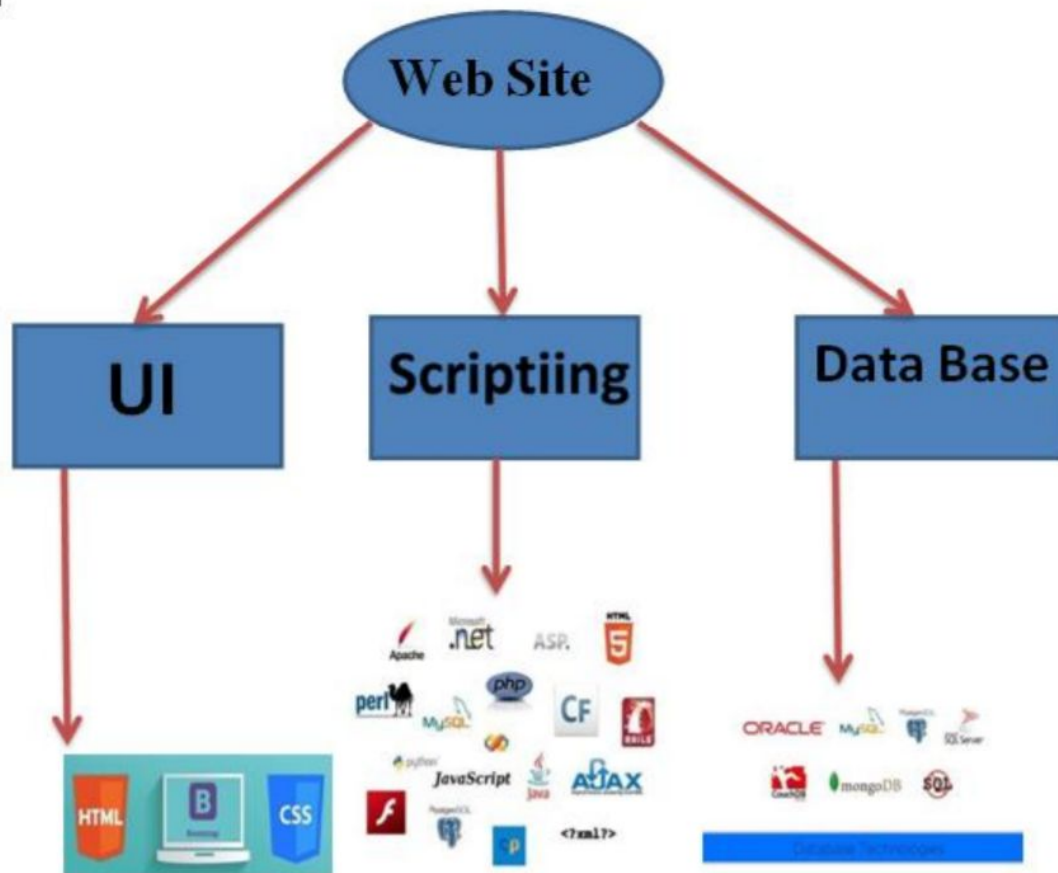
CHAPTER – 3

THE STEPS TO CREATE A WEBSITE

Creating a website includes the following steps:

- Creating a User Interface
- Scripting of both server end and client end
- Creating a backend database

Fig 3.1 steps to create a website



3.1 UI DEVELOPMENT

Technologies that are mostly used to develop a User Interface are:

- HTML
- CSS
- Bootstrap

3.1.1 HTML

HTML stands for Hypertext Markup Language. It allows the user to create and structure sections, paragraphs, headings, links, and blockquotes for web pages and applications. It can be assisted by technologies such as Style Sheets (CSS) and scripting languages such as Javascript.

Web Browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript which affect the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C),

maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

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 commonly 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).

Another important component is the HTML document type declaration, which triggers standards mode rendering.

The following is an example of the classic Hello world program, a common test employed for comparing programming languages, scripting languages and markup languages. This example is made using 9 lines of code.

General Syntax of HTML

```
<!DOCTYPE html>
<html>
<head>
<title>This is a title</title>
</head>
<body>
<p>Hello world!</p>
</body>
</html>
```

The text between `<html>` and `</html>` describes the web page, and the text between `<body>` and `</body>` is the visible page content. The markup text "`<title>This is a title</title>`" defines the browser page title.

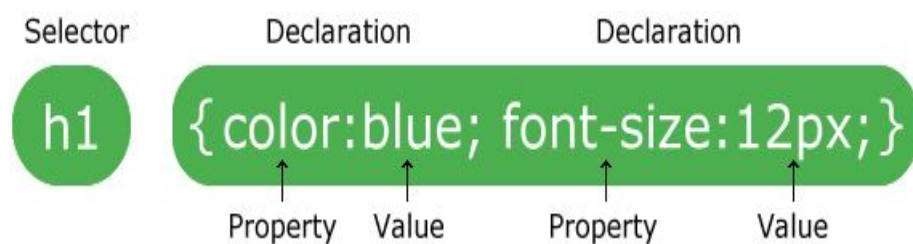
The Document Type Declaration `<!DOCTYPE html>` is for HTML5. If a declaration is not included, various browsers will revert to "quirks mode" for rendering.

3.1.2 CSS

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is one of the core languages of the open Web and is standardized across Web browsers according to the W3C specifications. Previously development of various parts of CSS specification was done synchronously, which allowed versioning of the latest recommendation. You might have heard about CSS1, CSS2.1, CSS3. However, CSS4 has never become an official version. From CSS3, the scope of the specification increased significantly and the progress on different CSS modules started to differ so much that it became more effective to develop and research. Instead of versioning the CSS specification, W3C now periodically takes a snapshot of the latest table space of CSS .

Fig 3.1.2.1 syntax of css



Types of CSS:

- **Inline CSS:**

Inline CSS is used to style a specific HTML element. For this CSS style, you'll only need to add the style attribute to each HTML tag, without using selectors.

This CSS type is not really recommended, as each HTML tag needs to be styled individually. Managing your website may become too hard if you only use inline CSS.

However, inline CSS in HTML can be useful in some situations. For example, in cases where you don't have access to CSS files or need to apply styles for a single element only.

Let's take a look at an example. Here, we add an inline CSS to the <p> and <h1> tag:

```
<!DOCTYPE html>
<html>
<body style="background-color:black;">
<h1 style="color:white;padding:30px;">Hostinger Tutorials</h1>
<p style="color:white;">Something useful here.</p>
</body>
</html>
```

- **Internal CSS:**

Internal or embedded CSS requires you to add <style> tag in the <head> section of your HTML document.

This CSS style is an effective method of styling a single page. However, using this style for multiple pages is time-consuming as you need to put CSS rules to every page of your website.

Here's how you can use internal CSS:

1. Open your HTML page and locate <head> opening tag.
2. Put the following code right after the <head> tag

```
<style type="text/css">
```

1. Add CSS rules on a new line. Here's an example:

```
body {  
    background-color: blue;  
}  
  
h1 {  
    color: red;  
    padding: 60px;  
}
```

Type the closing tag:

```
</style>
```

the HTML file will look like this:

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
<style>  
  
body {  
    background-color: blue;  
}  
  
h1 {  
    color: red;  
    padding: 60px;  
}  
  
</style>  
  
</head>
```

```
<body>
<h1>Hostinger Tutorials</h1>
<p>This is our paragraph.</p>
</body>
</html>
```

- **External CSS:**

With external CSS, you'll link your web pages to an external **.css** file, which can be created by any text editor in your device (e.g., **Notepad++**).

This CSS type is a more efficient method, especially for styling a large website. By editing one **.css** file, you can change your entire site at once.

Follow these steps to use external CSS:

1. Create a new **.css** file with the text editor, and add the style rules. For example:

```
.leftcol {
    float: left;
    width: 33%;
    background:#809900;
}
.middle col {
    float: left;
    width: 34%;
    background:#eff2df;
}
```

1. In the **<head>** section of your HTML sheet, add a reference to your external **.css** file right after **<title>** tag:

```
<link rel="stylesheet" type="text/css" href="style.css" />
```


3.1.3 JAVASCRIPT

Fig 3.1.3 Javascript



JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid email address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Syntax of JS:

```
<script>  
document.write("Basic Print method in JavaScript");  
</script>
```

3.1.4 BOOTSTRAP

The CSS framework in front end development is called Bootstrap. It is free, open source and is mainly designed to develop mobile applications in the front end. The CSS and Javascript templates are used for typing, buttons, navigation and other interacting components. The framework is faster and popular now for developing mobile websites. It is written in CSS, HTML, Less, Sass and JavaScript. Bootstrap is easy to learn and experimenting with the framework hardly takes a month to become an expertise in the process. The framework is actually heavy due to CSS and HTML files.

STEPS TO INSTALL BOOTSTRAP:

Step 1: Use the link below to get bootstrap downloaded. Click on Download, the bootstrap package will be downloaded in a Zip folder. This folder contains the CSS and JS folder.

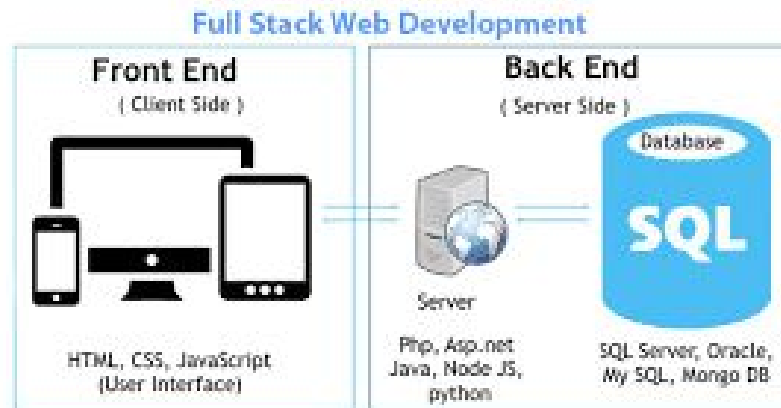
Link:- <https://getbootstrap.com/>

Step 2: Click on download on the top right corner. The new tab will be opened after that click download on that page. Then a zip file will be downloaded.

Step 3: Now extract that zip file then a folder is extracted. Open the folder then we can see css and js files in it .

3.2 SERVER SIDE AND CLIENT SIDE SCRIPTING

Fig 3.2 Server and client side Scripting



3.2.1 Client side scripting

Client-side scripting simply means running scripts, such as JavaScript, on the client device, usually within a browser. All kinds of scripts can run on the client side if they are written in JavaScript, because JavaScript is universally supported.

JavaScript is a client-side scripting language, which means the source code is processed by the client's web browser rather than on the web server. This means JavaScript functions can run after a web page has loaded without communicating with the server.

NEED OF CLIENT SIDE SCRIPTING:

Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser.

Client-side scripting (embedded scripts) is code that exists inside the client's HTML page. This code will be processed on the client machine and the HTML page

will NOT perform a PostBack to the web-server. Traditionally, client-side scripting is used for page navigation, data validation and formatting. The language used in this scripting is JavaScript. JavaScript is compatible and is able to run on any internet browser.

The two main benefits of client-side scripting are:

- The user's actions will result in an immediate response because they don't require a trip to the server.
- Fewer resources are used and needed on the web-server.

Fig 3.2.1.1 Client Side Scripting



3.2.2 SERVER SIDE SCRIPTING

Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page. Scripts can be written in any of a number of server-side scripting languages that are available (see below). Server-side scripting is distinguished from client-side scripting where embedded scripts, such as JavaScript, are run client-side in a web browser, but both techniques are often used together.

Server-side scripting is often used to provide a customized interface for the user. These scripts may assemble client characteristics for use in customizing the

response based on those characteristics, the user's requirements, access rights, etc. Server-side scripting also enables the website owner to hide the source code that generates the interface, whereas with client-side scripting, the user has access to all the code received by the client. A down-side to the use of server-side scripting is that the client needs to make further requests over the network to the server in order to show new information to the user via the web browser. These requests can slow down the experience for the user, place more load on the server, and prevent use of the application when the user is disconnected from the server.

1).

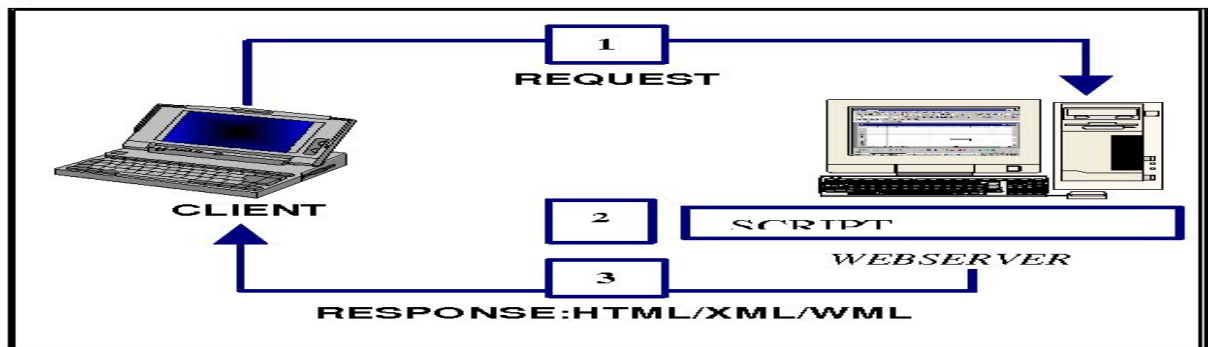


Fig. 1. Server Side Scripting.

Fig 3.2.2.2 Server Side Scripting

3.2.3 SCRIPTING LANGUAGES

DEFINITION: A scripting language is a programming language that is interpreted, meaning it is translated into machine code when the code is run rather than beforehand. JavaScript, Python, and Ruby are all examples of scripting languages.

NodeJs:

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

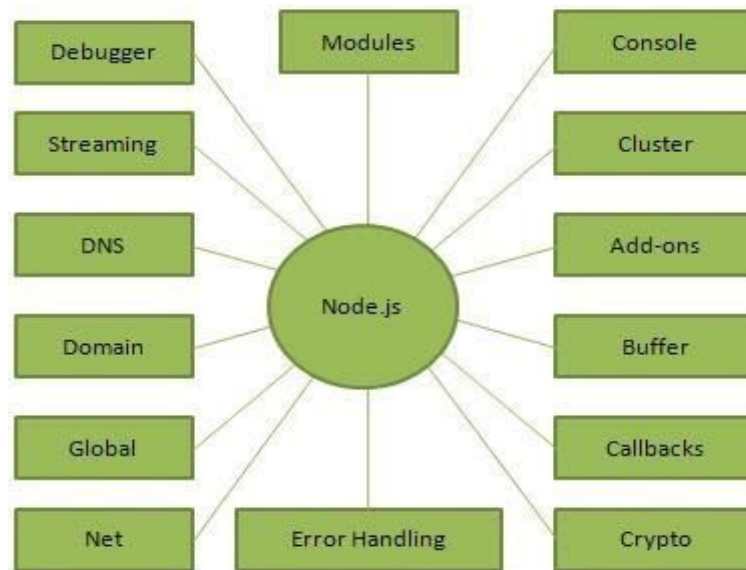


Fig 3.2.3.1 Uses of NodeJS

JavaScript:

JavaScript is a high-level, interpreted programming language that conforms to the ECMAScript specification. It is a language that is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying on these upon the host environment in which it is embedded.

PHP (Hypertext Preprocessor):

PHP (Hypertext Preprocessor) is a server-side scripting language designed for Web development. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter

implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

AJAX :

AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Javascript.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the server using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.

With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server. XML is commonly used as the format for receiving server data, although any format, including plain text, can be used. AJAX is a web browser technology independent of web server software. A user can continue to use the application while the client program requests information from the server in the background. Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger. Data-driven as opposed to page-driven.

JQUERY :

jQuery is a JavaScript library that allows web developers to add extra functionality to their websites. It is open source and provided for free under the MIT license. **jQuery** can also work with Ajax code and scripting languages, such as PHP and ASP to access data from a database. In recent years, jQuery has become the most popular JavaScript library used in web development. To implement jQuery, a web developer simply needs to reference the jQuery JavaScript file within the HTML of a webpage. Some websites host their own local copy of jQuery, while others simply reference the library hosted by Google or the jQuery server. For example, a web page may load the jQuery library using the following line within the <head> section of the HTML:

```
<script type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
```

Once the jQuery library is loaded, a web page can call any jQuery function supported by the library. Common examples include modifying text, processing form data, moving elements on a page, and performing animations. jQuery can also work with Ajax code and scripting languages, such as PHP and ASP to access data from a database. Since jQuery runs on the client side (rather than the web server), it can update information on a webpage in real time, without reloading the page. A common example is "autocomplete," in which a search form automatically displays common searches as you type your query. In fact, this is how TechTerms.com provides search suggestions when you type in the search box.

3.3 DATABASE

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy. Let us discuss a few examples: An online telephone directory uses a database to store data of people, phone numbers, other contact details. Your electricity service provider uses a database to manage billing, client-related issues, handle fault data, etc. Let us also consider Facebook. It needs to store, manipulate, and present data related to members, their

friends, member activities, messages, advertisements, and a lot more. We can provide a countless number of examples for the usage of databases.

The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages.

3.4 SQL

SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.

SQL offers three main advantages over older read-write APIs such as ISAM or VSAM. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, e.g. with or without an index. Finally, SQL uses a human-readable syntax that allows users to be quickly productive without a requirement for long-term, technical training.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements, which may be informally classed as sublanguages, commonly: a data query language (DQL), a data definition language (DDL), a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control. Although SQL is essentially a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages to utilize Edgar F. Codd's relational model. The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then the standard has been revised to include a larger set of features. Despite the existence of standards, most SQL code requires at least some changes before being ported to different database systems. Although there is an ISO standard for SQL, most of the implementations slightly vary in syntax. So we may encounter queries that work in SQL Server but do not work in MySQL.

SQL COMMANDS STATEMENTS :

SELECT - extracts data from a database

UPDATE - updates data in a database

DELETE - deletes data from a database

INSERT INTO - inserts new data into a database

CREATE DATABASE - creates a new database

ALTER DATABASE - modifies a database

CREATE TABLE - creates a new table

ALTER TABLE - modifies a table

DROP TABLE - deletes a table

COMMIT- saves all the work done.

SAVEPOINT-used for saving all the current points in the processing of a transaction.

ROLLBACK-rollback command restores database to original since the last commit.

SET TRANSACTION-used for placing a name on a transaction.

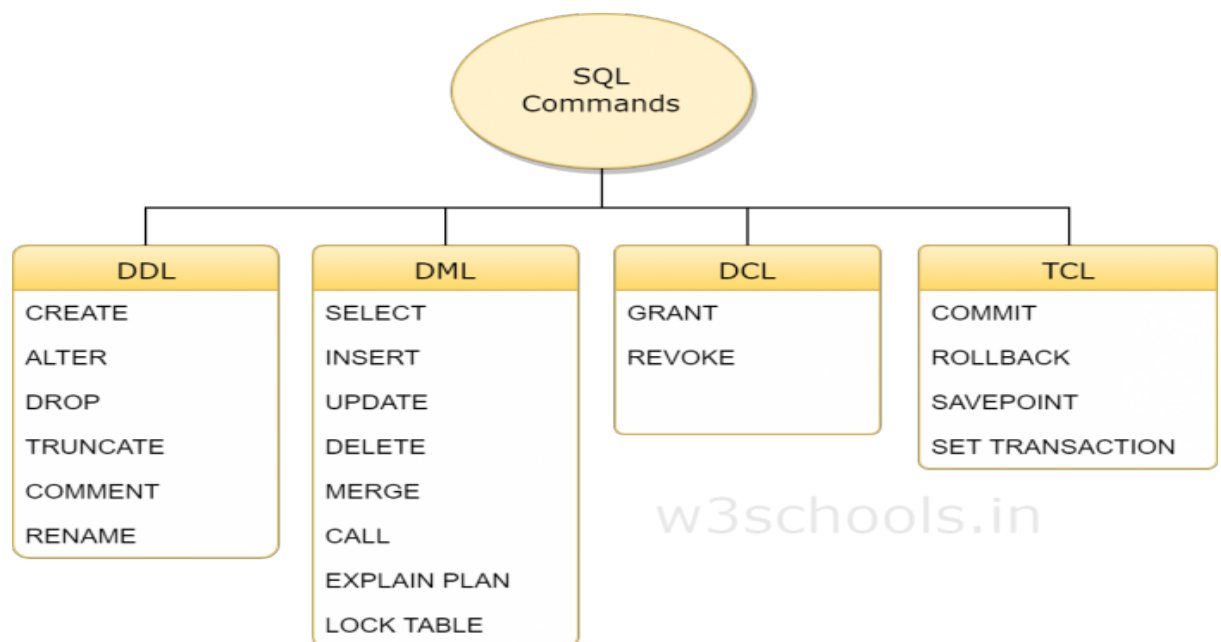


Fig-3.4 sql commands

CHAPTER-4

SOFTWARE REQUIREMENTS

Hardware requirements. The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems.

HARDWARE REQUIREMENTS

PROCESSOR	Intel i5
RAM	8GB
HARD DISK	1TB

SOFTWARE REQUIREMENTS

NUMBER	DESCRIPTION
1	WINDOWS 10
2	HTML,CSS,JS
3	NODEJS,MYSQL(mysql database)
4	COMPILER(VISUAL STUDIO CODE)

CHAPTER-5

DATA FLOW DIAGRAMS

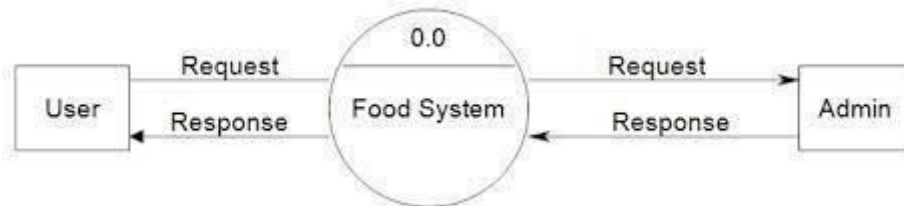


Fig-5.1 context level diagram

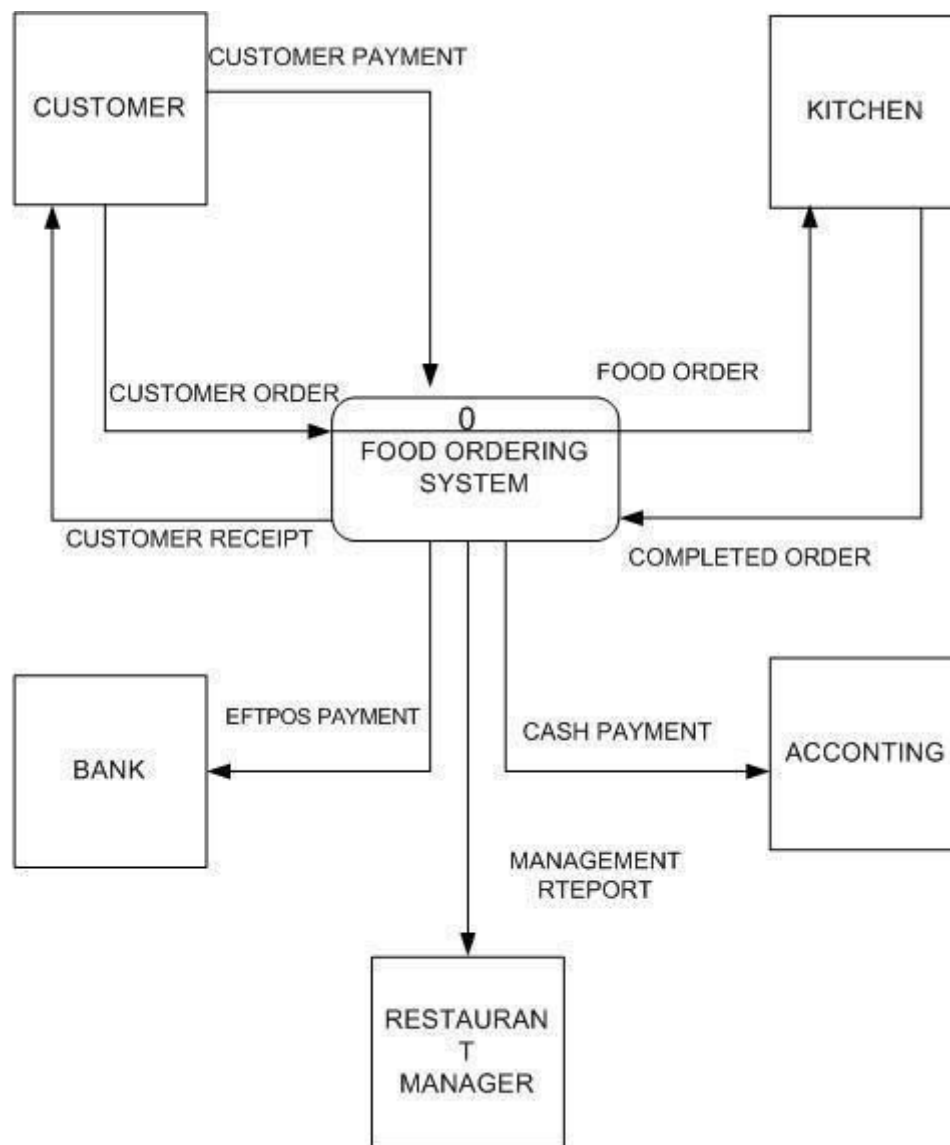


Fig 5.2- DFD-1

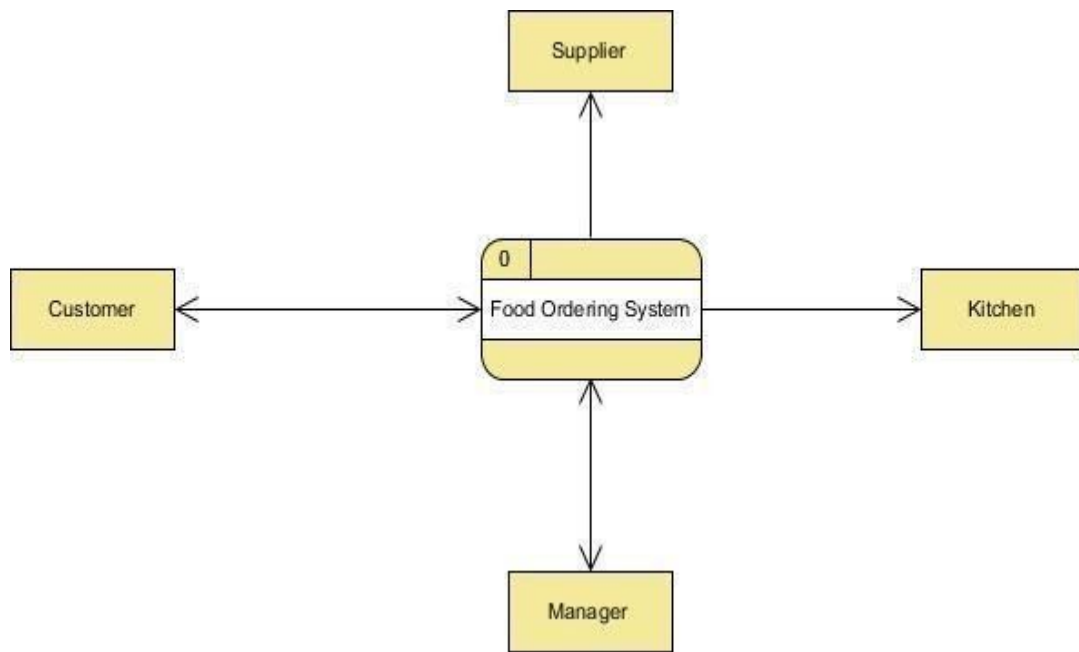


Fig 5.3 DFD-2

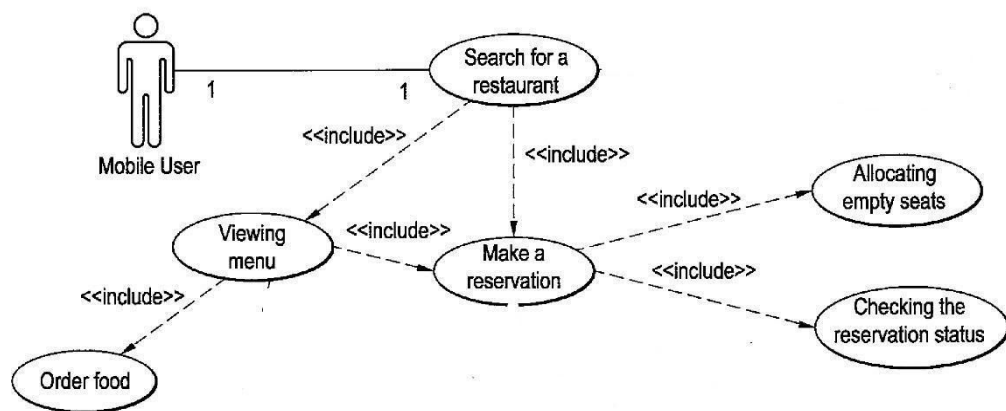


Fig-5.4 use case diagram

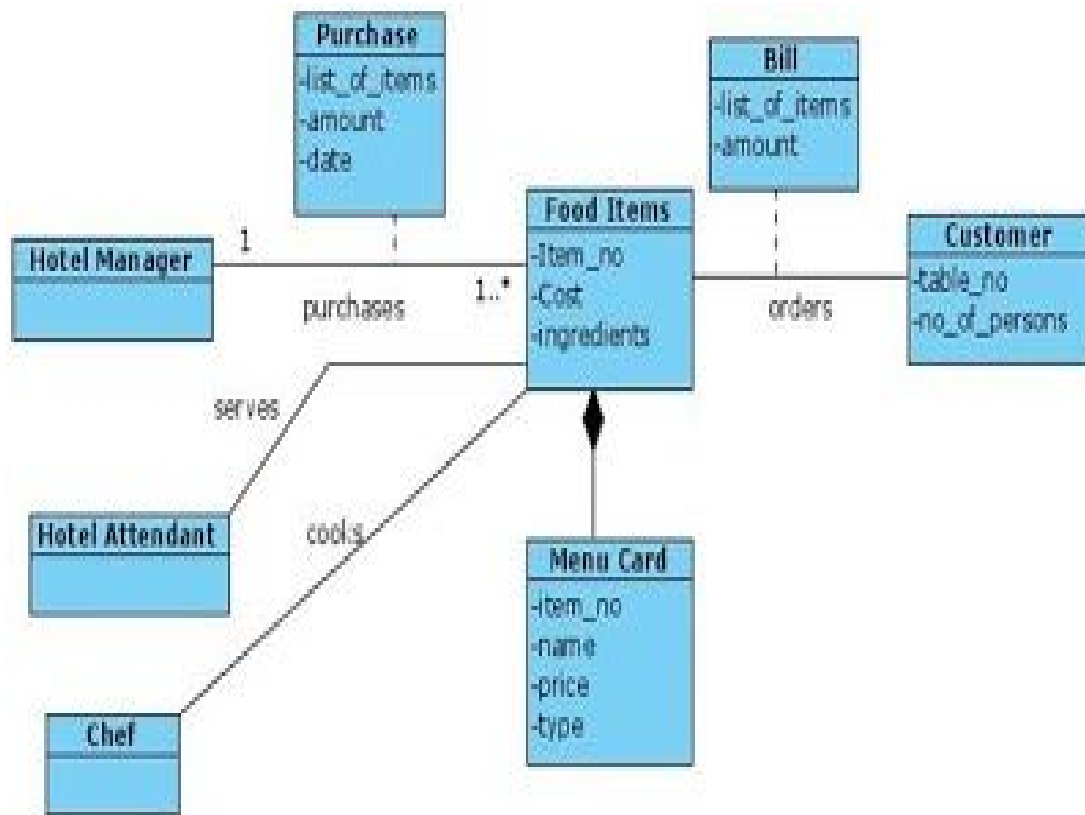


Fig -5.5 class diagram

CHAPTER-6

PROJECT

NAME : RESTAURANT WEBSITE

USED TECHNOLOGIES:

- JAVASCRIPT(Nodejs)
- WIREFRAME PRO
- HTML

TECHNICAL DETAILS :

- To design the frontend i.e UI design we used html, css, bootstrap ,jquery.
- To design the backend of our project we used a database named mysql database and also we used nodejs to write the code for designing the backend .

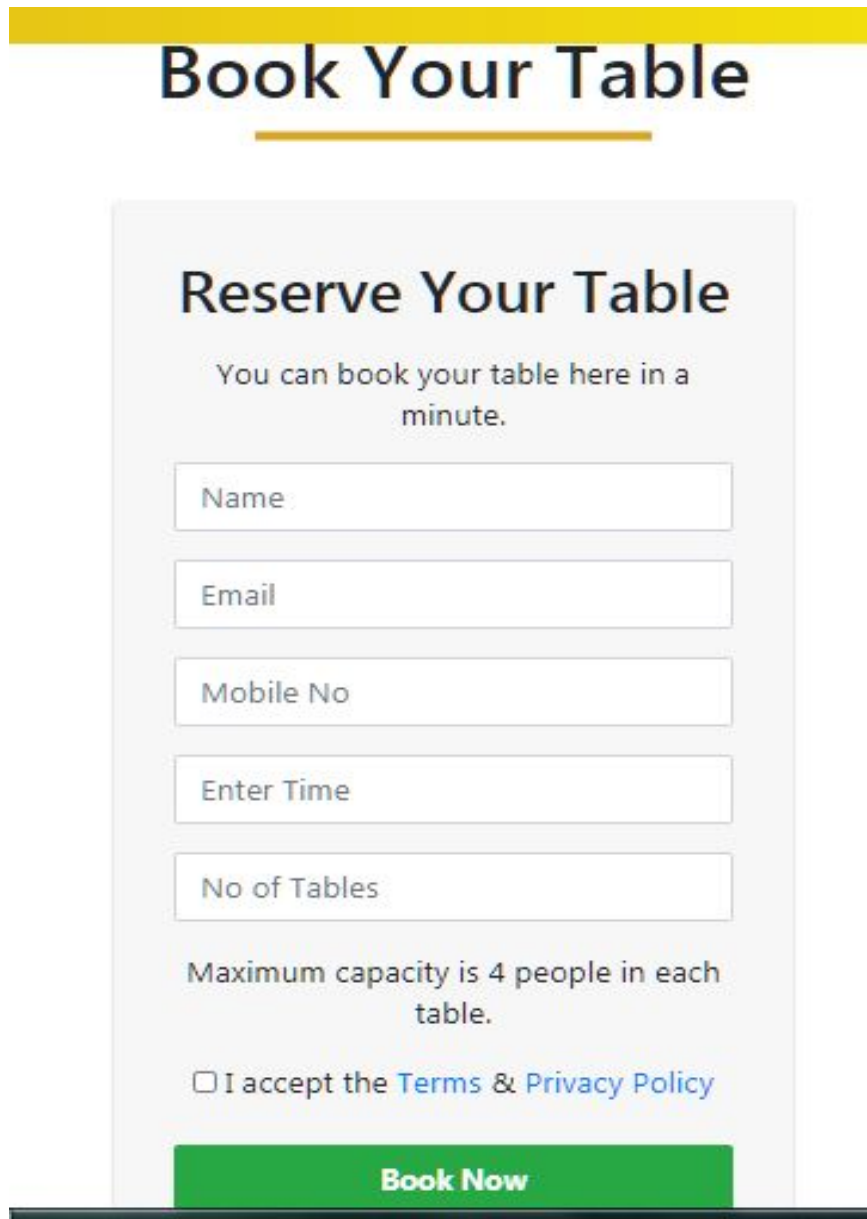
NAME	ROLLNO	WORK DONE
A.Kartheek	221710301004	Front End(Booking Form),BackEnd (Nodejs,MySql Tables).
U.kushalava Reddy	221710308058	Front End(Html,CSS)
J.srinivas Reddy	221710308022	Frontend(Js,Feedback .Html),Wire Frames.

Database: MySql

Operating System: Windows7/8/8.1/10

Wireframing tool: (<https://wireframepro.mockflow.com/>)

6.1 SCREENSHOTS



The screenshot displays a web interface for booking a table. At the top, a yellow horizontal bar contains the text 'Book Your Table' in a large, bold, black font. Below this, a light gray rectangular box contains the heading 'Reserve Your Table' in bold black text. Under the heading is a subtext: 'You can book your table here in a minute.' This is followed by five vertically stacked input fields, each with a light gray border and placeholder text: 'Name', 'Email', 'Mobile No', 'Enter Time', and 'No of Tables'. Below the input fields, a note states 'Maximum capacity is 4 people in each table.' followed by a checkbox and the text 'I accept the [Terms & Privacy Policy](#)'. At the bottom of the gray box is a green rectangular button with the white text 'Book Now'. The entire form is set against a white background with a thin black horizontal line at the very bottom.

Book Your Table

Reserve Your Table

You can book your table here in a minute.

Name

Email

Mobile No

Enter Time

No of Tables

Maximum capacity is 4 people in each table.

☐ I accept the [Terms & Privacy Policy](#)

Book Now

Fig 6.1.1-booking table

Feedback

Give your valuble feedback here.

Name

Email

Mobile No

Give your Feedback here

☐ I accept the [Terms](#) & [Privacy Policy](#)

Submit Feedback

Fig 6.1.2- feedback table

CHAPTER-7

MODULES ASSIGNED

7.1 BOOKING PAGE

This page generally contains booking a table in a restaurant through which users can book a table. This page concentrates on the user details, this page contains a form through which the user can fill the form such as name, email id and no. of tables. After filling the form, click on submit; then the user will get a confirmation message i.e. your tables are booked.

7.2 NODEJS

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

7.3 MYSQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database.

CHAPTER-8

CODES AND OUTPUTS

- Booking.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <title>project </title>

  <link rel="stylesheet" href="style.css">

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
></script>

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.
css">

  <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>

</head>

<body>

  <section id="nav-bar">

    <nav class="navbar navbar-expand-lg navbar-light">

      <a class="navbar-brand" href="#"></a>
```

```

    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">

        <i class="fa fa-bars" aria-hidden="true"></i>

    </button>

    <div class="collapse navbar-collapse" id="navbarNav">

        <ul class="navbar-nav ml-auto">

            <li class="nav-item active">

                <a class="nav-link" href="index.html">Home </a>

            </li>

            <li class="nav-item">

                <a class="nav-link " href="#booking" >Book your Table</a>

            </li>

            <li class="nav-item">

                <a class="nav-link" href="#footer">Contact</a>

            </li>

        </ul>

    </div>

</nav>

</section>

<!-------banner section----->

<section id="banner">

<div class="container">

```

```

<div class="row">

<div class="col-md-6">

  <p class="promo-title">FOODIE</p>

  <p>Taste the best food and drinks here.</p>

</div>

<div class="col-md-6 text-center">

</div>

</div>

</div>

<div>



</section>

<!--booking-->

<section id="booking" style="padding-top: 50px;">

  <div class="container text-center">

    <h1 class="title">Book Your Table</h1>

    <div class="feedback-form">

      <form action="" method="POST">

        <h2 class="text-center">Reserve Your Table</h2>

        <p>You can book your table here in a minute.</p>

```

```

<div class="form-group">

    <input type="text" id="txtname" class="form-control" name="name"
placeholder="Name" required="required">

</div>

<div class="form-group">

    <input type="email" id="txtemail" class="form-control"
placeholder="Email" required="required">

</div>

<div class="form-group">

    <input type="mobile" id="mobile" class="form-control" name="mobile"
placeholder="Mobile No" pattern="^[7-9][0-9]{9}$"

    required="required"></div>

<div class="form-group">

    <input type="text" id="txttime" class="form-control" name="time"
placeholder="Enter Time"
pattern="([0-9]|0[0-9]|1[0-9]|2[0-3]):([0-5][0-9])\s*([AaPp][Mm])"
required="required">

</div>

<div class="form-group">

    <input type="number" id="number" min="1" max="5"
class="form-control" placeholder="No of Tables" required="required">

</div>

<p>Maximum capacity is 4 people in each table.</p>

<div class="form-group">

    <label class="checkbox-inline">

```

```

        <input type="checkbox" required="required"> I accept the <a
href="">Terms</a> &amp;

        <a href="">Privacy Policy</a>

    </label>

</div>

<div class="form-group">

    <button type="submit" id="btnSubmit" class="btn btn-success btn-block"
onclick="alert('your tables booked')">Book Now

    </button>

</div>

</form>

</div>

</div>

</section>

<!--social media-->

<section id="social-media">

    <div class="container text-center">

        <p>FIND US ON SOCIAL MEDIA</p>

        <div class="social-icons">

            <a href="#"></a>

        </div>

    </div>

</section>

```



```

var scroll = new SmoothScroll('a[href*="#"]');

</script>

<style type="text/css">

    .feedback-form {

        width: 340px;

        margin: 50px auto;

    }

    .feedback-form form {

        margin-bottom: 15px;

        background: #f7f7f7;

        box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.3);

        padding: 30px;

    }

    .feedback-form h2 {

        margin: 0 0 15px;

    }

    .form-control, .btn {

        min-height: 38px;

        border-radius: 2px;

    }

    .btn {

        font-size: 15px;

        font-weight: bold;

    }

    .a1 {

```

```

        height: 100px;

        text-align: center;

    }

    .a2{

        height: 45px;

    }

</style>

<script src="js/jquery.min.js"></script>

<script type="text/javascript" accesskey="">

$(document).ready(function () {

    $("#btnSubmit").click(function () {

        debugger;

        var api_url = "http://localhost:3000/bookings";

        var data = {

            name: $('#txtname').val(),

            email: $('#txtemail').val(),

            mobile: $('#mobile').val(),

            time: $('#tx time').val(),

            number: $('#number').val(),

        }

        $.ajax({

            url: api_url,

            type: "POST",

            dataType: "json",

            data: data,

```

```

    success: function (d) {

        alert("values are submitted");

    },

    error: function () {

        alert("wrong with insert");

    }

});

});

});

</script>

</head>

</body>

</html>

```

- db.config.js

```

module.exports = {

    HOST : "localhost",

    USER : "root",

    PASSWORD : "karthu@123",

    DB:"restaurant"

};

```

- booking.controller.js

```
const Booking = require("../models/booking.model.js");
```

```
exports.create = (req, res) => {
```

```
  // Validate request
```

```
  if (!req.body) {
```

```
    res.status(400).send({
```

```
      message: "Content can not be empty!"
```

```
    });
```

```
  }
```

```
  const booking = new Booking({
```

```
    name: req.body.name,
```

```
    email: req.body.email,
```

```
    mobile: req.body.mobile,
```

```
    time: req.body.time,
```

```
    number: req.body.number,
```

```
  });
```

```
  Booking.create(booking, (err, data) => {
```

```
    if (err)
```

```
      res.status(500).send({
```

```

        message:

            err.message || "Some error occurred while creating the Booking."

        });

    else res.send(data);

    });

};

exports.findAll = (req, res) => {

    Booking.getAll((err, data) => {

        if (err)

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving bookings."

                });

            else res.send(data);

        });

    });

};

exports.findOne = (req, res) => {

    Booking.findById(req.params.bookingId, (err, data) => {

        if (err) {

            if (err.kind === "not_found") {

                res.status(404).send({

```

```

        message: `Not found Booking with id ${req.params.bookingId}`

    });

    } else {

        res.status(500).send({

            message: "Error retrieving Booking with id " + req.params.bookingId

        });

    }

    } else res.send(data);

});

};

exports.delete = (req, res) => {

    Booking.remove(req.params.bookingId, (err, data) => {

        if (err) {

            if (err.kind === "not_found") {

                res.status(404).send({

                    message: `Not found Booking with id ${req.params.bookingId}`

                });

            } else {

                res.status(500).send({

                    message: "Could not delete Booking with id " + req.params.bookingId

                });

            }

        } else res.send({ message: `Booking was deleted successfully!` });

    });

};

```

```

exports.deleteAll = (req, res) => {

  Booking.removeAll((err, data) => {

    if (err)

      res.status(500).send({

        message:

          err.message || "Some error occurred while removing all Bookings."

      });

    else res.send({ message: `All Bookings were deleted successfully!` });

  });

};

```

- db.js

```

const mysql = require('mysql');

const dbConfig = require('../config/db.config.js');

const connection = mysql.createConnection({

  host:dbConfig.HOST,

  user:dbConfig.USER,

  password:dbConfig.PASSWORD,

  database:dbConfig.DB

});

connection.connect(error =>{

  if(error){

```



```

        return console.error(error.message);
    }

    console.log('Successfully connected to MySQL DATABASE');
});

module.exports = connection;

```

- booking.model.js

```

const sql = require('../models/db.js');

const Booking = function (booking) {
    this.name = booking.name;
    this.email = booking.email;
    this.mobile=booking.mobile;
    this.time=booking.time;
    this.number = booking.number;
};

Booking.create = (newBooking, result) => {
    sql.query(`insert into bookings set ?`, newBooking, (err, res) => {
        if (err) {
            console.log(err);
            result(err, null);
            return;
        }
    })
}

```

```

        console.log("Created Booking : ", { id: res.insertedId, ...newBooking });

        return (null, { id: res.insertedId, ...newBooking });
    })
};

```

```

Booking.findById = (bookingId, result) => {

    sql.query(`select * from bookings where Id = ${bookingId}`, (err, res) => {

        if (err) {

            console.log(err);

            result(err, null);

            return;

        }

        if (res.length) {

            console.log('Found Booking:', res[0]);

            result(null, res[0]);

            return;

        }

        result({ kind: 'not_found' }, null);

    })

};

```

```

Booking.getAll = result => {

    sql.query('select * from bookings', (err, res) => {

        if (err) {

            console.log(err);

```

```

        result(err, null);

        return;
    }

    console.log('Bookings : ', res);

    result(null, res);
})
};

Booking.remove = (id, result) => {

    sql.query('delete from bookings where id = ?', id, (err, res) => {

        if (err) {

            console.log(err);

            result(null, res);

            return;

        }

        if (res.affectedRows == 0) {

            result({ kind: 'Not_Found' }, null);

            return;

        }

        console.log('Deleted booking with Id : ', id);

        result(null, res);

    });
};

Booking.removeAll = result => {

    sql.query('delete from bookings', (err, res) => {

        if (err) {

```

```

        console.log(err);

        result(null, res);

        return;
    }

    console.log('Deleted ${res.affectedRows} bookings')

    result(null, res);

    });
};

module.exports = Booking;

```

- booking.routes.js

```

module.exports = app =>{

    const bookings = require('../controllers/booking.controller.js');

    app.post ('/bookings',bookings.create);

    app.get('/bookings',bookings.findAll);

    app.get('/bookings/:bookingId',bookings.findOne);

    app.delete('/bookings/:bookingId', bookings.delete);

    app.delete("/bookings",bookings.deleteAll);

}

```

- customer.controller.js

```

const Customer = require("../models/customer.model.js");

```

```

// Create and Save a new Customer

exports.create = (req, res) => {

  // Validate request

  if (!req.body) {

    res.status(400).send({

      message: "Content can not be empty!"

    });

  }

  // Create a Customer

  const customer = new Customer({

    name: req.body.name,

    email: req.body.email,

    mobile: req.body.mobile,

    feedback: req.body.feedback

  })

  // Save Customer in the database

  Customer.create(customer, (err, data) => {

    if (err)

      res.status(500).send({

        message:

          err.message || "Some error occurred while creating the Customer."

      });

  });

```

```

    else res.send(data);

  });

};

// Retrieve all Customers from the database.
exports.findAll = (req, res) => {

  Customer.getAll((err, data) => {

    if (err)

      res.status(500).send({

        message:

          err.message || "Some error occurred while retrieving customers."

      });

    else res.send(data);

  });

};

// Find a single Customer with a customerId
exports.findOne = (req, res) => {

  Customer.findById(req.params.customerId, (err, data) => {

    if (err) {

      if (err.kind === "not_found") {

        res.status(404).send({

          message: `Not found Customer with id ${req.params.customerId}.`

        });

      } else {

```

```

        res.status(500).send({
            message: "Error retrieving Customer with id " + req.params.customerId
        });
    }
    } else res.send(data);
});
};

// Delete a Customer with the specified customerId in the request
exports.delete = (req, res) => {
    Customer.remove(req.params.customerId, (err, data) => {
        if (err) {
            if (err.kind === "not_found") {
                res.status(404).send({
                    message: `Not found Customer with id ${req.params.customerId}.`
                });
            } else {
                res.status(500).send({
                    message: "Could not delete Customer with id " + req.params.customerId
                });
            }
        } else res.send({ message: `Customer was deleted successfully!` });
    });
};

// Delete all Customers from the database.

```

```

exports.deleteAll = (req, res) => {

  Customer.removeAll((err, data) => {

    if (err)

      res.status(500).send({

        message:

          err.message || "Some error occurred while removing all customers."

      });

    else res.send({ message: `All Customers were deleted successfully!` });

  });

};

```

- customer.model.js

```

const sql = require('../models/db.js');

const Customer = function (customer) {

  this.name = customer.name;

  this.email = customer.email;

  this.mobile = customer.mobile;

  this.feedback = customer.feedback;

};

Customer.create = (newCustomer, result) => {

  sql.query('insert into customers set ?', newCustomer, (err, res) => {

    if (err) {

      console.log(err);

    }

  });

};

```



```

        result(err, null);

        return;
    }

    console.log("Created Customer : ", { id: res.insertedId, ...newCustomer });

    return (null, { id: res.insertedId, ...newCustomer });
})
};

```

```

Customer.findById = (customerId, result) => {

    sql.query(`select * from customers where Id = ${customerId}`, (err, res) => {

        if (err) {

            console.log(err);

            result(err, null);

            return;

        }

        if (res.length) {

            console.log('Found Customer:', res[0]);

            result(null, res[0]);

            return;

        }

        result({ kind: 'not_found' }, null);

    })

};

```

```

Customer.getAll = result => {

```

```

sql.query('select * from customers', (err, res) => {

  if (err) {

    console.log(err);

    result(err, null);

    return;

  }

  console.log('Customers : ', res);

  result(null, res);

})

};

Customer.remove = (id, result) => {

  sql.query('delete from customers where id = ?', id, (err, res) => {

    if (err) {

      console.log(err);

      result(null, res);

      return;

    }

    if (res.affectedRows == 0) {

      result({ kind: 'Not_Found' }, null);

      return;

    }

    console.log('Deleted Customer with Id : ', id);

    result(null, res);

  });

};

```

```

Customer.removeAll = result => {

  sql.query('delete from customers', (err, res) => {

    if (err) {

      console.log(err);

      result(null, res);

      return;

    }

    console.log('Deleted ${res.affectedRows} Customers')

    result(null, res);

  });

};

module.exports = Customer;

```

- customer.routes.js

```

module.exports = app => {

  const customers = require('../controllers/customer.controller.js');

  app.post('/customers', customers.create);

  app.get('/customers', customers.findAll);

  app.get('/customers/:customerId', customers.findOne);

  app.delete('/customers/:customerId', customers.delete);

  app.delete('/customers', customers.deleteAll);

};

```

- server.js

```

const express = require('express');

```

```
const bodyparser = require('body-parser');

const app = express();

app.use(bodyparser.json());

app.use(bodyparser.urlencoded({ extended: true }));

app.get("/", (req, res) => {

  res.json({ message: 'welcome' });

});

require('./app/routes/booking.routes.js')(app);
require('./app/routes/customer.routes.js')(app);

app.listen(3000, () => {

  console.log('Server is Running on port 3000');

});
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Vidya Rani\Desktop\restaurent project>node server.js
Server is running on port 3000
Successfully connected to MySQL DATABASE
█
```

Fig 8.1-connecting to database

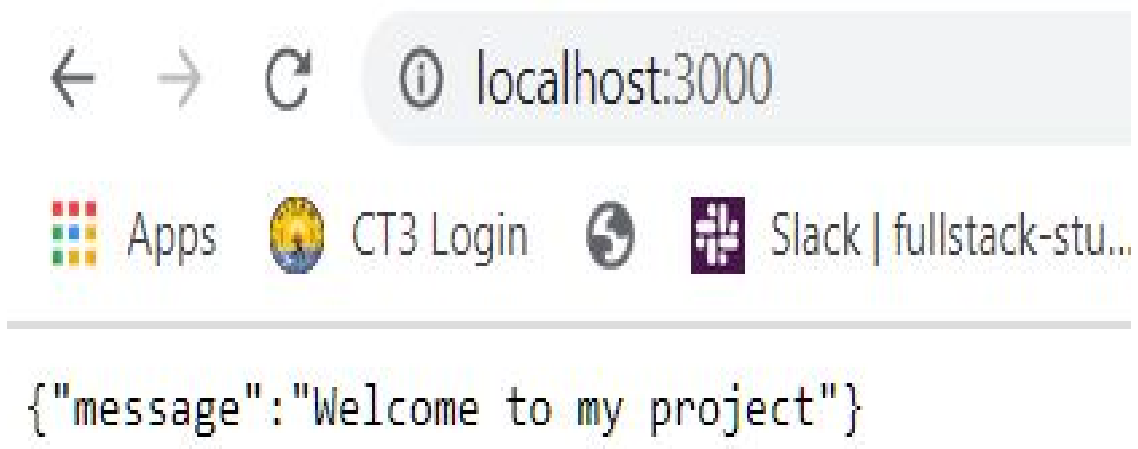


Fig 8.2-checking for localhost port number

This page says
your tables booked

OK

Reserve Your Table

You can book your table here in a minute.

ajit

ajit345k@gmail.com

7848032705

6:00pm

3

Maximum capacity is 4 people in each table.

☒ I accept the [Terms](#) & [Privacy Policy](#)

Book Now

Fig 8.3-output for booking table

This page says

your feedback submitted

OK

Feedback

Give your valuble feedback here.

Ravi

raviismart19k@gmail.com

9850576165

nice taste

☒ I accept the [Terms](#) & [Privacy Policy](#)

Submit Feedback

Fig 8.4-output for feedback table

- Mysql Database

```
create database restaurant;
use restaurant;
] create table bookings(
    Id int auto_increment,
    Name varchar(45) not null,
    Email varchar(50) not null,
    mobile varchar(10),
    time varchar(10),
    number varchar(20) not null,
    primary key(Id));
select * from bookings;
] create table customers(
    Id int auto_increment,
    Name varchar(50) not null,
    Email varchar(45) not null,
    mobile varchar(10),
    Feedback varchar(5000) not null,
    primary key(Id));
select * from customers;
```

Fig 8.5-creating database and tables

Id	Name	Email	mobile	time	number
1	kartheek	akarthekmsd@gmail.com	9505761654	4:30pm	1
2	kavya	kavya234@gmail.com	9848032705	1:30pm	3
3	arun	arun19@gmail.com	7848032705	3:00pm	2
4	demo2	demo21956@gmail.com	9850576165	1:00pm	2
5	pkpranu	pkpranui21@gmail.com	8885493274	2:30pm	5
6	kishan rao	kishanrao1996	7048032705	3:45pm	2
7	pawankalyan	kpawan2000@gmail.com	9850876165	12:30pm	4
8	maheshbabu	mahesh991@gmail.com	9834578164	7:00pm	5
9	suresh	sureshbabu21@gmail.com	9948092809	9:00pm	1
10	nikil	nikilwwe234@gmail.com	9618509693	7:30pm	3

Fig 8.6-output for table bookings data

Id	Name	Email	mobile	Feedback
1	kartheek	ak123@gmail.com	9505761654	good restaurant and super taste
2	kavya	kavya234@gmail.com	9848032705	nice taste
3	arun	arun19@gmail.com	7848032705	good restaurant and super taste
4	demo2	demo21956@gmail.com	9850576165	taste is wonderful
5	pkpranu	pkpranui21@gmail.com	8885493274	super restaurant
6	kishan rao	kishanrao@1996	7048032705	good restaurant
7	pawankalyan	kpawan2000@gmail.com	9850876165	super taste
8	maheshbabu	mahesh991@gmail.com	9834578164	good restaurant and super taste
9	suresh	sureshbabu21@gmail.com	9948092809	wow super restaurant
10	nikil	nikilwwe234@gmail.com	9618509693	nice restaurant

Fig 8.7-output for customers feedback data

st


This page says
wrong with insert

OK

Ravi

raviismart19k@gmail.com

6777xyxuc

 Please match the requested format.

1

Maximum capacity is 4 people in each table.

☒ I accept the [Terms](#) & [Privacy Policy](#)

Book Now

Fig 8.8-wrong with inserting into booking table

This page says

wrong with insert

OK

raviismart19k@gmail.com

678-54778-hcjxcm



Please match the requested format.

nice taste

☒ I accept the [Terms](#) & [Privacy Policy](#)

Submit Feedback

Fig 8.9-wrong with inserting into feedback table

CHAPTER-9

MAINTENANCE

During this maintenance phase we made some minor changes to our project . especially to user interface design we made the changes to not complex changes but some simple changes we have done. we changed our design as per the guidance of our mentor.

We tried our best to make it look better. We also used the updated versions of our tools with which the work for us became much more simple.

CHAPTER-10
FUTURE SCOPE AND ENHANCEMENT
PROJECT NAME : RESTAURANT WEBSITE

- We are going to add a new feature to our website with which the customer can order food from anywhere in the country.
- We are going to add a payment mode feature to our website with which the customers can pay bills online from anywhere in the country.
- We are going to add special features like some games in restaurants for which customers have to book their slot and can have a lot of fun.

CHAPTER-11

CONCLUSION

We made our website work successfully. We hope we can do more than what we think i.e adding an online order. End user is the thing that we have concentrated because satisfying the end user is the most satisfiable thing for us.

To complete this website we used tools which are of the latest versions, We don't want any of the people to sleep hungry so we are taking this to the next level by online order process.

BIBLIOGRAPHY

1. www.jquery-ui.com
2. www.fontawesome.com
3. www.wikipedia.com
4. www.w3schools.com
5. www.getbootstrap.com
6. www.youtube.com/html
7. <https://www.javatpoint.com>

GITHUB -LINK: <https://github.com/ak950/fullstack>

