



Department of Computer Science & Engineering

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Approved by AICTE, Permanently Affiliated to JNTU, KAKINADA

Accredited by NBA & Accredited by NAAC with 'A' Grade

Nambur (V), Peda Kakani (M), Guntur (Dt) – 522508

An Internship Report

on

WEB FULL STACK DEVELOPER

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

Bachelor of Technology

in

Computer Science & Engineering

by

GUDAVALLI KUMAR

22BQ1A0576

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY,
NAMBUR
(Autonomous)**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision of the Department

Providing quality education to enable the generation of socially conscious software engineers who can contribute to the advancement in the field of computer science and engineering.

Mission of the Department

- To equip the graduates with the knowledge and skills required to enable them to be industry ready.
- To train socially responsible, disciplined engineers who work with good leadership skills and can contribute for nation building.
- To make our graduates proficient in cutting edge technologies through student centric teaching-learning process and empower them to contribute significantly to the software industry
- To shape the department into a center of academic and research excellence

Program Educational Objectives	
PEO-1	To provide the graduates with solid foundation in Computer Science and Engineering along with the fundamentals of Mathematics and Sciences with a view to impart in them high quality technical skills like modeling, analyzing, designing, programming and implementation with global competence and helps the graduates for life-long learning .
PEO-2	To prepare and motivate graduates with recent technological developments related to core subjects like Programming, Databases, Design of Compilers and Network Security aspects and future technologies so as to contribute effectively for Research & Development by participating in professional activities like publishing and seeking copy rights.
PEO-3	To train graduates to choose a decent career option either in high degree of employability/Entrepreneur or, in higher education by empowering students with ethical administrative acumen, ability to handle critical situations and training to excel in competitive examinations.
PEO-4	To train the graduates to have basic interpersonal skills and sense of social responsibility that paves them a way to become good team members and leaders.

Program Specific Outcomes (PSOs)

PSO-1: Professional Skills: The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity.

PSO-2: Successful Career and Entrepreneurship: The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur and a zest for higher studies/employability in the field of Computer Science & Engineering.

Program Outcomes:

- 1. Engineering knowledge:** apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems. **(L3-Apply)**
- 2. Problem analysis:** identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences. **(L4-Analysis)**
- 3. Design/development of solutions:** design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations. **(L6-Create)**
- 4. Conduct investigations of complex problems:** use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. **(L5-Evaluation)**
- 5. Modern tool usage:** create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations. **(L3-Apply)**
- 6. The engineer and society:** apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. **(L3-Apply)**
- 7. Environment sustainability:** understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. **(L2-Understand)**
- 8. Ethics:** apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. **(L1-Remember)**
- 9. Individual and team work:** function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings. **(L1-Remember)**
- 10. Communication:** communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. **(L1-Remember)**
- 11. Project management and finance:** demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. **(L3-Apply)**
- 12. Lifelong learning:** recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change. **(L1-Remember)**

Department of CSE

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Approved by AICTE, Permanently Affiliated to JNTU, KAKINADA
Accredited by NBA & Accredited by NAAC with 'A' Grade
Nambur (V), Peda Kakani (M), Guntur (Dt) – 522508



BONAFIDE CERTIFICATE

This is to certify that this Internship report is the Bonafide work of “**GUDAVALLI KUMAR** (22BQ1A0576)”, who carried out the Internship under my SPOC during the academic year 2024-2025 towards partial fulfillment of the requirements of the Degree of **Bachelor of Technology in CSE** from **Jawaharlal Nehru Technological University, Kakinada**.

Signature of the SPOC

Mr. T. Seshu Chakravarthy M.Tech,(Ph.D)
Assistant Professor, CSE

Signature of the Head of the Department

Dr . V. RamaChandran M.Tech, Ph.D.
Professor, CSE

Submitted for Viva voce Examination held on _____

EXTERNAL EXAMINER

ABSTRACT

The Web Full Stack Developer Virtual Internship on the Eduskills platform is a comprehensive training program that immerses participants in core and advanced web development concepts. This virtual internship provides hands-on experience with both front-end and back-end technologies, empowering interns to build, deploy, and maintain full-featured web applications from scratch.

The curriculum covers essential technologies, including HTML, CSS, JavaScript, Tailwind CSS for streamlined styling, and version control with Git for effective project management. On the backend, participants gain expertise in database management with both SQL and MongoDB, preparing them to handle diverse data requirements. Throughout the program, interns learn to integrate and manage these technologies cohesively, enabling seamless, efficient web development.

Through real-world projects and collaborative tasks, interns develop skills in responsive design data management, and UI/UX best practices, as well as key problem-solving and debugging abilities. Working under the guidance of mentors, participants learn to create scalable applications, ensuring both functionality and user experience are optimized. The goal of the Web Full Stack Developer Virtual Internship is to equip participants with in-demand technical skills, practical experience, and a comprehensive understanding of the development lifecycle. By the end of the program, interns are ready to contribute to professional development teams, capable of building high-quality, dynamic web applications that meet modern industry standards.

LETTER OF UNDERTAKING

To
The Principal
Vasireddy Venkatadri Institute of
Technology ,
Nambur,

Subject: Submission of Internship Report on Web Full Stack Developer Virtual
Internship on Eduskills platform.

Dear Sir,

I am pleased to submit my internship report on “**Web Full Stack Developer Virtual Internship**” as per your instruction to fulfil the requirements of the Degree of **Bachelor of Technology** in CSE from **Jawaharlal Nehru Technological University, Kakinada**. While preparing this report, I have tried my level best to include all the relevant information, explanations, things I learned from the Internship Courses, my contribution to this programme to make the report informative and comprehensive. It would not have been possible to complete this report without your assistance, of which I am very thankful. Working for three months on Web Full Stack Developer Virtual Internship in online was amazing and a huge learning opportunity for me. Also, it was a great experience to prepare this report and I will be available for any clarification, if required. Therefore, I pray and hope that you would be kind enough to accept my Internship Report and oblige thereby.

Yours Obediently,

Gudavalli Kumar

Regno: 22BQ1A0576

E-mail: 22bq1a0576@vvit.net

CERTIFICATE OF INTERNSHIP



अखिल भारतीय तकनीकी शिक्षा परिषद्
All India Council for Technical Education



Certificate of Virtual Internship

This is to certify that

gudavalli kumar

Vasireddy Venkatadri Institute of Technology

has successfully completed 10 weeks

Web Full Stack Developer Virtual Internship

During July - September 2024

Supported By



Shri Buddha Chandrasekhar
Chief Coordinating Officer (CCO)
NEAT Cell, AICTE

Dr. Satya Ranjan Biswal
Chief Technology Officer (CTO)
EduSkills



Certificate ID :7b8c752a584f158872866f1462c5109e

Student ID :STU6677b5e05f8231719121376



GRADE- O (Outstanding):90-100 | E (Excellent):80-89 | A (Very Good):70-79 | B (Good): 60-69 | C (Fair): 50-59 | D (Average): 40-49 | P (Pass): 30-39 | F (Fail): Below 30

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and appreciation to all those people who made this Internship work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and help me towards the successful completion of this Internship work.

First and foremost, we express our deep gratitude to **Mr. Vasireddy VidyaSagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the Computer Science & Engineering program.

We express our sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the Computer Science & Engineering program.

We express our sincere gratitude to **Dr. V. Rama Chandran**, Professor & HOD, Computer Science & Engineering, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express our sincere gratitude to our VVIT INTERNSHIP I/C **Mr. Y.V. Subba Reddy**, SPOC **Mr. T. Seshu Charkravathy** and our Internship Coordinators **Mrs. D. Vamsi & Mrs. B. Ramya AsaLatha** for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this Internship.

We would like to take this opportunity to express our thanks to the **teaching and non- teaching** staff in the Department of Computer Science and Technology, VVIT for their invaluable help and support.

Gudavalli Kumar
22BQ1A0576

Table of contents

EDUSKILLS WEB FULL STACK DEVELOPER VIRTUAL INTERNSHIP

MODULES	CONTENTS	DATE	PAGE NO
1. HTML	<ul style="list-style-type: none">• Internet Basics• HTML Fundamentals• Tables• Lists	13-07-24	1 – 6
2. CSS	<ul style="list-style-type: none">• Selectors• Box Model• Display• Position	20-07-24	7 - 10
3. Javascript	<ul style="list-style-type: none">• Variables• Data types• Operators• Functions• Control Structures	27-07-24	11 - 15
4. Tailwind CSS	<ul style="list-style-type: none">• Introduction• Advantages• Background Classes & shades• Element sizing• Padding and Margins• Text styles• Borders• Flex Box• Responsive design	03-08-24	16 - 19
5. Version Control	<ul style="list-style-type: none">• GitHub Repository• Git Basics• Git tags and releases• Project Management on GitHub• Repository Settings	17-08-24	20 - 23
6. Web Hosting	<ul style="list-style-type: none">• Types of Hosting• Domain/Domain Name• Steps for Hosting	24-08-24	24 - 30

7. SQL	<ul style="list-style-type: none"> • Overview of Database and relational database • Introduction to MYSQL • CRUD operations • Indexing & Optimisation • Procedures & Functions • Security 	7-09-24	31 - 34
8. MongoDB	<ul style="list-style-type: none"> ❖ Introduction and Setup • What is MongoDB? • Installing MongoDB • Key Concepts ❖ CRUD Operations • Creating Database and Collections • Inserting documents • Querying documents • Updating Documents • Deleting Documents • Indexing ❖ Aggregation and Data Modelling • Aggregation framework • Examples of aggregation • Data Modelling 	14-09-24	35 - 40

AICTE INTERNSHIP WEEKLY REPORT

Department of CSE, VVIT

Student Roll Number : 22BQ1A0576
Student Name : GUDAVALLI KUMAR
Branch : CSE
Year of Study : 2024 – 2025
AICTE Student Profile ID : STU654deead1b3661699606189
AICTE Regd. E-Mail ID : 22BQ1A0576@vvit.net
Contact Number : 9032413505
Internship Course Taken : EDUSKILLS WEB FULL STACK DEVELOPER

Weeks & Dates	Module	SPOC Signature
Week-1	HTML <ul style="list-style-type: none">• Internet Basics• HTML Fundamentals• Lists & Tables	
Week-2	CSS <ul style="list-style-type: none">• Selectors• Box Model• Position & Display	
Week-3	JavaScript <ul style="list-style-type: none">• Variables• Data types• Functions	

Week-4	Tailwind CSS <ul style="list-style-type: none"> • Background Classes & shades • Element sizing • Padding and Margins 	
Week-5	Tailwind CSS <ul style="list-style-type: none"> • Borders • Flex Box • Responsive design 	
Week-6	Version Control <ul style="list-style-type: none"> • GitHub Repository • Project Management on GitHub • Repository Settings 	
Week-7	Web Hosting <ul style="list-style-type: none"> • Types of Hosting • Domain/Domain Name 	
Week-8	Web Hosting <ul style="list-style-type: none"> • Steps for Hosting 	
Week-9	SQL <ul style="list-style-type: none"> • Overview of Database and relational database • Introduction to MYSQL • CRUD operations 	
Week-10	MongoDB <ul style="list-style-type: none"> • Introduction and Setup • CRUD Operations • Updating Documents • Aggregation and Data Modelling 	

HTML

HTML (Hyper Text Markup Language):

The standard markup language for creating web pages. It structures the content on the web.

Internet Basics

Introduction to the Internet

- Internet provides a global network, where interconnected nodes are accessible to organizations and individuals through communication devices and media. A **node** refers to any device, such as a computer, tablet, or smartphone that is part of a network. A network is formed when two or more computers are connected wired or wirelessly to share resources and information.
- Currently, **Internet of Things (IoT)** encompasses the growth of devices connecting to a network, ranging from televisions to household appliances. Data transfer lines in networks facilitate the transfer of data between different computers. The Internet backbone consists of high-speed data transfer connections that interconnect major computer systems worldwide. (Server, cloud ... etc.)

Domain Name System (DNS)

- DNS or Domain Name System, serves as the Internet's equivalent of a phonebook. While humans access information online through easily memorable domain names like google.com or eduskillsfoundation.org, web browsers communicate through less user-friendly Internet Protocol (IP) addresses. DNS plays a crucial role in translating these human-readable domain names into machine-friendly IP addresses, enabling browsers to load the desired Internet resources.
- In the intricate process of DNS resolution, a hostname (e.g ., www.example.com) is converted into a corresponding IP address (e.g ., 192.168.1.1).
- Point to remember, An Internet Service Provider (ISP) is a company with a permanent connection to the Internet backbone, enabling it to provide access to the internet for users.
- Similar to how a home address guides us to a specific home, an IP address is essential for locating a particular device/server on the Internet. When users wish to access a webpage/website, there must be a translation between what they input into their web browser (e.g ., example.com) and the machine-friendly address required to locate the desired webpage.

HTML Fundamentals

What is HTML?

HTML, which stands for Hyper Text Markup Language, is the basic code used to create web pages. It defines the layout and structure of a webpage through a set of elements. These elements instruct the browser on how to present content, such as headings, paragraphs, links, and more.

HTML Evolution:

HTML 1.0 (1991):

This is where it all began. The first version of HTML was introduced by Tim Berners-Lee in 1991. It was a simple markup language used to create basic documents with headings, paragraphs, and links. Think of it as the foundational stage, where the web was just starting to take shape.

HTML 2.0 (1995):

The second version brought improvements and added new features. It introduced attributes like "align" for better control over page elements. Tables were introduced, allowing for better organization of content.

HTML 3.2 (1997):

HTML 3.2 expanded further on the capabilities of the language. This version introduced support for scripting languages like JavaScript. It also added new elements like the table caption and improved support for forms.

HTML 4.01 (1999):

HTML 4.01 was a significant step forward in standardizing the language. It introduced style sheets, allowing for better control of page layout and design. The concept of frames for dividing the browser window into multiple sections was also introduced.

XHTML (2000):

XHTML (extensible HyperText Markup Language) was introduced to make HTML more compatible with XML (extensible Markup Language). It followed stricter syntax rules and aimed for a cleaner, more organized code structure.

HTML5 (2014): (currently in use)

HTML5 marked a major milestone in the evolution of HTML.

HTML Elements

- HTML is the primary language for crafting web documents, and these documents typically bear the file extensions .html or .htm. In HTML, tags are strings enclosed by the symbols "<" and ">." For instance, an opening tag is represented as <html>, and its corresponding closing tag is </html>.
- Now let's try to understand the below basic HTML code shown in Fig-1.



Fig-1: Code and output on HTML elements

- The <! DOCTYPE html> declaration indicates that the document follows the HTML5 standard.
- The <html> element serves as the root element for an HTML page.

- The <head> element contains meta information related to the HTML page.
- The <title> element specifies the title displayed in the browser's title bar or tab.
- The <body> element encapsulates all visible content, including headings, paragraphs, images, links, tables, lists, etc.
- The <h1> element denotes a large heading.
- The <h2> element denotes a medium heading
- The <p> element is used to define a paragraph.

Understanding more HTML tags:

-
 tag defines a line break, and is an empty element without a closing tag.
- <meta> tag provides metadata about the HTML document such as descriptions and keywords for search engine.
- <table> tags are used to create tables.
- tags are used to insert images.
- <!-- > tag is used to write comments.

Tables

Understanding HTML Table Structure

HTML tables allow web developers to arrange data into rows and columns in a webpage.

<table>: Specifies the creation of a table.

<th>: Designates a header cell within a table.

<tr>: Identifies a row within a table.

<td>: Specifies a cell within a table.

<caption>: Provides a caption for a table.

<colgroup>: Specifies a set of one or more columns in a table, facilitating formatting.

<col>: Defines properties for each column within a <colgroup> element.

<thead>: Groups the content related to the header in a table.

<tbody>: Groups the content related to the body in a table.

<tfoot>: Groups the content related to the footer in a table.

Table Borders:

To add Basics border we will be using the CSS border property on <table>, <th>, and <td> elements.

Syntax:

```
table, th, td{
    border : 2px solid black;
}
```

Let use see an example code in Fig-2.1 and its output in Fig-2.2.


```

<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
}
</style>
</head>
<body>

<h2>Table With Border</h2>

<p>Use the CSS border property to add a
border to the table.</p>

<table style="width:100%">
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Alice</td>
    <td>Johnson</td>
    <td>30</td>
  </tr>
  <tr>
    <td>Bob</td>
    <td>Miller</td>
    <td>45</td>
  </tr>
  <tr>
    <td>Charlie</td>
    <td>Williams</td>
    <td>55</td>
  </tr>
</table>
</body>
</html>

```

Fig-2.1: Code on Table Border

Table With Border

Use the CSS border property to add a border to the table.

First Name	Last Name	Age
Alice	Johnson	30
Bob	Miller	45
Charlie	Williams	55

Fig-2.2: Output of above code

The border attribute in the <table> tag is used to control the width of the table border. Let's see table border example for the code given in Fig-2.3 and its output is shown in Fig-2.4.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title> Table Border
  Example</title>
  <style>
    table {
      border-collapse: collapse;
    }
    td, th {
      border: 2px solid black;
      padding: 8px;
    }
  </style>
</head>
<body>

  <h2>Table Border Example</h2>

  <table border="1">
    <tr>
      <td>Cell 1</td>
      <td>Cell 2</td>
    </tr>
    <tr>
      <td>Cell 3</td>
      <td>Cell 4</td>
    </tr>
  </table>

</body>
</html>

```

Fig-2.3: Border example for table



Fig-2.4: Output of above code

Lists

Numbered List: A list that can be either numbered or ordered is referred to as a numbered/ordered list.

Tags utilized to construct an ordered list include:

- ``
- `` tag Defines an individual list item.
Should be employed as a subordinate tag of either the `` or `` tag.
- `` tag defines an ordered list.

- The ordered list can feature numerical or alphabetical sequencing.
- Attributes within the `` tag include:

1)start

2)type

Lets see an example code and its output in Fig-3:

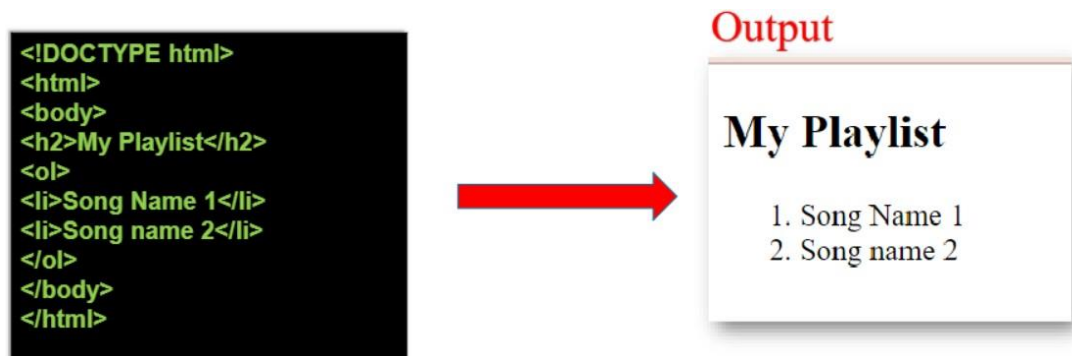


Fig-3: Code and Output of `` list

Type Attribute:

Indicates the type of marker to utilize in the list.

Acceptable values for the Type Attribute include:

1. A: Uppercase letters
2. a: Lowercase letters
3. I: Uppercase Roman letters
4. i: Lowercase Roman letters
5. 1: Standard numbers (default)

Let's see an example code using type attribute in Fig-4.

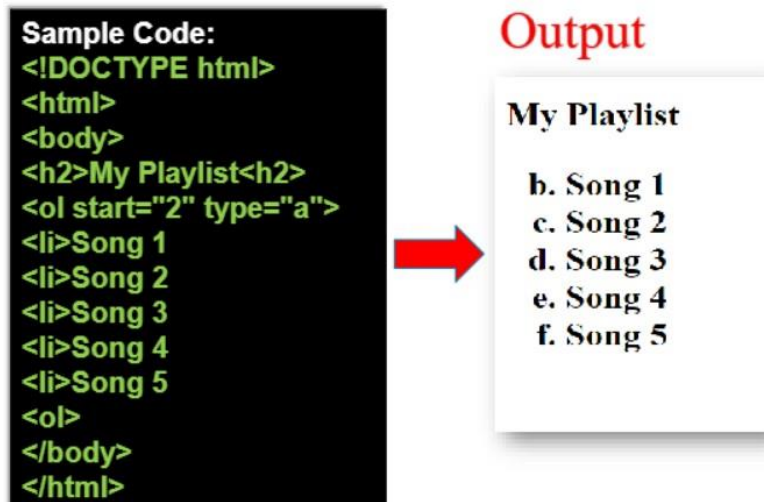


Fig-4: Code and output of type attribute

Bulleted List: List of items prefixed with bullets is called Bulleted /Unordered List.

** tag** defines an unordered list. In short, an unordered list items will be marked with bullets.

Let's see an example for bulleted list in Fig-5.

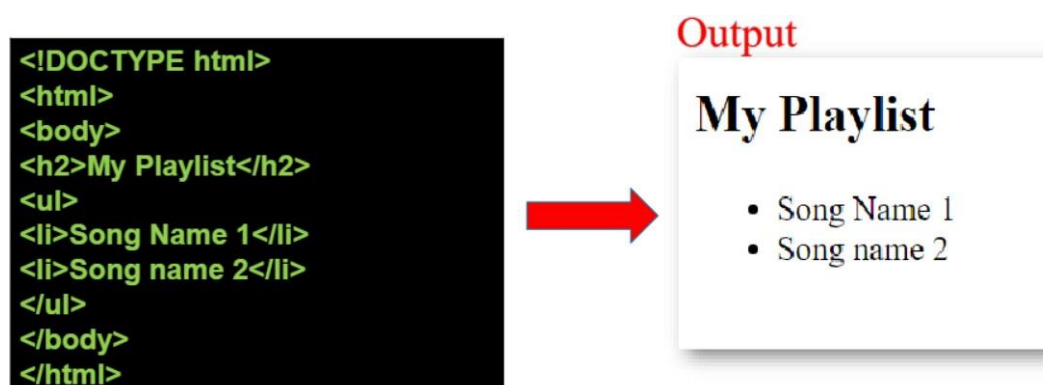


Fig-5: Code and output of Bulleted list

CSS

About CSS

CSS Syntax:

CSS (Cascading Style Sheets) is used to style HTML elements on a webpage. To apply styles, you use a selector to target the HTML element you want to style.

CSS Selector:

The selector is like a pointer that indicates which HTML element you want to style.

For example, if you want to style all paragraphs, you use the selector `p`.

Declaration Block:

The declaration block is where you define the styles for the selected HTML element.

It is enclosed in curly braces `{ }`.

Declarations:

Inside the declaration block, you have one or more declarations separated by semicolons. Each declaration consists of a CSS property name and a value, separated by a colon.

Selectors

Simple selectors:

1. Element Selector: Selects HTML elements by their name.

Example: `p {`
`//style`
`}`

2. Class Selector: Selects elements with a specific class attribute.

Example: `.class_name{`
`//style`
`}`

3. ID Selector: Selects a specific element by its ID attribute.

Example: `#id{`
`//style`
`}`

Attribute Selectors:

1. Attribute Exists Selector: Selects elements with a specified attribute.

Example:
`[target]{`
`//style`
`}`

2. Attribute Value Selector: Selects elements with a specific attribute value.

`[type='text']{`
`//style`
`}`

Pseudo-Class Selectors:

Pseudo-class selectors are used to select and style elements based on their state or position in the document structure. They are denoted by a colon (`:`) followed by the pseudo-class name. Here are some commonly used pseudo-class selectors:

1. **:hover**- Selects and styles an element when the mouse is over it.

Example :

```
button:hover{  
    color:red;  
}
```

2. **:active**: Selects and styles an element when it is being activated (e.g., clicked).

Example:

```
button:active{  
    color:green;  
}
```

CSS Box Model

Note: All HTML elements can be considered as boxes.

The concept of the "box model" in CSS pertains to the design and layout of elements. It involves a rectangular structure that envelops each HTML element. The fundamental components of the CSS box model include content, padding, borders, and margins. The diagram below visually represents the hierarchical arrangement of these elements within the box model.

The box model is shown in Fig-6.

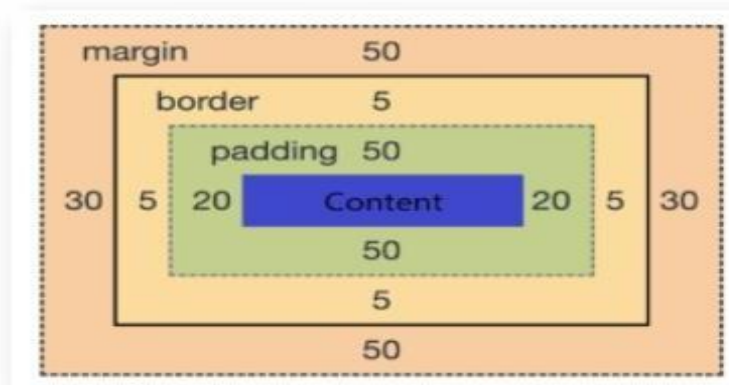


Fig-6: Box model

Explanation:

- **Content:** This refers to the internal area of the box where text and images are displayed.
- **Padding:** It defines the space between the content and the border. Essentially, it clears an area around the content. The padding is transparent, allowing the background to show through.
- **Border:** A border surrounds both the padding and the content, providing a visual distinction. It defines the outermost boundary of the box.
- **Margin:** The margin is the transparent space outside the border. It clears an area around the border, separating the element from its neighbouring elements or the outer container.
- The box model allows us to add a border around elements, and to define space between elements.

Display in CSS:

The display property defines how an element is displayed. It can alter the default behaviour of an element, making it block, inline, or other display types.

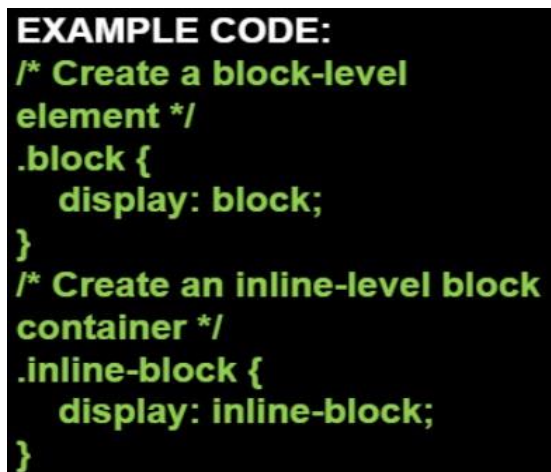
Syntax:

```
selector{  
    display: value;  
}
```

Common Values:

- block: The element will generate a block-level box.
- Inline: The element will generate an inline box.
- Inline-block: The element will generate an inline-level block container.
- none: The element is not displayed.
- flex: The element becomes a flex container.
- grid: The element becomes a grid container.

Let's see an example code for display property in Fig-6.



```
EXAMPLE CODE:  
/* Create a block-level  
element */  
.block {  
    display: block;  
}  
/* Create an inline-level block  
container */  
.inline-block {  
    display: inline-block;  
}
```

Fig-6: Code for using display property

Position in CSS:

The position property is used to control the positioning of an element within its containing element.

Syntax:

```
Selector {  
    Position : value;  
}
```

Common Values:

- static: Default value. Elements are positioned according to the normal flow of the document.
- relative: Positioned relative to its normal position.
- absolute: Positioned relative to its nearest positioned ancestor (if sticky {any}), otherwise relative to the initial containing block.
- fixed: Positioned relative to the browser window or the device screen.
- sticky: A hybrid of relative and fixed positioning. The element is treated as relative positioned until it crosses a specified point, then it is treated as fixed positioned.

Let's see an example for using the position property to style our html elements in the below given figure (Fig-7).

```
EXAMPLE CODE:  
/* Position an element  
absolutely */  
.absolute {  
    position: absolute;  
    top: 20px;  
    left: 30px;  
}  
/* Make an element sticky */  
.sticky {  
    position: sticky;  
    top: 0;
```

Fig-7: Code for using position property

JavaScript

JavaScript is a high-level, interpreted programming language that is primarily used to create interactive effects within web browsers. It allows developers to manipulate web page content, respond to user actions, and dynamically update the page without reloading. JavaScript is essential for modern web development, enabling functionalities like form validation, interactive maps, and animations.

JavaScript (JS) is a cross-platform, object-oriented programming language used by developers to make web pages interactive. It allows developers to create dynamically updating content, use animations, pop-up menus, clickable buttons, control multimedia, etc.

Embedding to HTML

To insert a script in an HTML document, use the `<script>` tag. Use the `type` attribute to define the scripting language.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>
</head>
<body>
  <script type="text/javascript">
    document.write("hello World");
  </script>
</body>
</html>
```

- The two forward slashes(//) are a JavaScript comment symbol.
- Prevent the JavaScript from trying to compile the line.

```
<html lang="en">
<head>
<meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <script type="text/javascript">
    // document.write("hello World");
  </script>
```



```
</body>
```

```
</html>
```

External JavaScript

Write a script in an external file, and save with .js file extension.

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>Document</title>
```

```
</head>
```

```
<body>
```

```
    <script src="script_file.js"></script>
```

```
</body>
```

```
</html>
```

Variables

A Variable is a named item in a program that stores a data value. You can introduce variables in your code by declaring them. Declaring a variable tells the JavaScript interpreter to reserve memory space for the variable.

To declare a JavaScript variable, use the statement :

```
var variable_name = value;
```

Variables declared within a function is called local

- Is accessible only within the function
- Is destroyed when the function exits
- Variables declared outside the function is called global
- Is accessible anywhere
- Is destroyed when the page is closed

Working with Datatypes

- You must declare a variable before using it.
- JavaScript variables
- Numeric
- String
- Boolean
- Null

Numeric value is any number, such as 13,22.5 , - 3.1456 etc. Text string is any group of text characters such as "Hello" or "Happy Holidays!", must be enclosed within either double or single quotations .Boolean Values accepts only TRUE or FALSE value. NULL value has no value at all.

Operators:

Arithmetic Operators: +, -, *, /, %, ++, --

Assignment Operators: =, +=, -=, *=, /=, %=

Comparison Operators: ==, !=, >, <, <=, >=

Logical Operators: &&, ||, !

Functions

A function is a collection of commands that performs an action or returns a value. A function name identifies a function.

Syntax:

```
function function_name(argument1,argument2,etc)
```

```
{
```

```
some statements;
```

```
}
```

Use return statement to return a value to the calling expression.

```
function result(a,b)
```

```
{
```

```
c=a+b;
```

```
return c;
```

```
}
```

Flow with control Structure

If and If.....else statement

Syntax:

```
if (condition)
```

```
{
```

```
code to be executed if condition is true
```

```
}
```

Syntax:

```
if (condition)
```

```
{
```

```
code to be executed if condition is true
```

```
} else {
```

code to be executed if condition is false

```
}
```

Example:

```
<script type="text/javascript">
//If the time on your browser is less than 10,
//you will get a "Good morning" greeting.
var d=new Date() var time=d.getHours()
    if (time<10)
    {
        document.write("<b>Good morning</b>")
    } else {
        document.write("Good day!")
    }
</script>
```

Switch Statement

Syntax:

```
switch (expression)
{
    case label1:
        code to be executed if expression = label1
        break
    case label2:
        code to be executed if expression = label2
        break
    default:
        code to be executed if expression is different from both label1 and
        label2
}
```

Example:

```
<script type="text/javascript">
//You will receive a different greeting based on what day it is.
// Note that Sunday=0, Monday=1, Tuesday=2, etc.
```

```
var d=new Date()
    theDay=d.getDay()
    switch (theDay)
    {
    case 5:
    document.write("Finally Friday"); break
    case 6:
    document.write("Super Saturday"); break
    case 0: document.write("Sleepy Sunday"); break
    default: document.write("I'm looking forward to this weekend!")
    }
</script>
```

Looping

While

Syntax:

```
While (Condition){
    Code to be executed
}
```

The block of code will executed until the condition is false.

Do..... while

Syntax:

```
do
{
    Code to be executed
} while (condition)
```

The block of code will be executed until the condition is false.

For

Syntax:

```
For (initialization , condition , increment)
{
    Code to be executed
}
```

Tailwind CSS

Introduction

- The first release of Tailwind CSS came in 2017. It was accepted that the development is much faster with this new framework.
- Tailwind is known as the Utility-First CSS Framework.
- In this framework, you are not required to write a lot of CSS code. Instead, you will be creating many classes for the HTML elements.
- It is highly flexible for transferring the look and feel of the element on the website.

Advantages of Tailwind CSS

- No need for huge CSS files with custom CSS
- Easy to make design changes right from the view files
- Better developer experience
- Mobile-first designs from the start
- Designed with defaults for a polished and designed look and feel
- Easily customizable with the config file.

Background classes & shades

- This set of classes change the background color of an element using a scale of 100-900 for shades and a palette of over 90 shades.
- Syntax: `.bg-*-{100-900} {}`

An example is shown in Fig-8.



Fig-8:Background class and shades example

Element sizing

- All of the numbers in Tailwind are based around the rem unit of measurement.
- 1 rem is equal to the size of the base font of the document. As an example, if the base font size is 16px then 1 rem is equal to 16px and we can deduce that 1.25 rem is equal to 20px. To help with these fractional numbers, Tailwind's numbered classes are multiplied by 4 to avoid having numbers with decimal places.

Sizing with classes

- Syntax: `.w-*{} , .h-* {}`
- Available sizes in REM {0, 1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 20, 24, 32, 40, 48, 56, 64}
- Sizing in percentages 1/2... 1/{3, 4, 5, 6, 12}
- Sizing utilities {screen, full}

Padding and margins

- These classes add padding and margin to an element using the Tailwind numbering system.
- Example: `.p-*{} , .m-*{} , .p{x,y}-*{} , .m{x,y}-*{}`

Styling Text

Font styling is a huge part of any design and Tailwind has plenty of classes that we can use to style the text on our apps. It even includes utility classes for transformations like uppercase.

`.font-sans, .font-serif { }, .font-mono { }`

Sizing

- `.text-xs { }`
- `.text-sm { }`
- `.text-base{ }`
- `.text-lg{ }`
- `.text-xl { }`
- `.text-2xl { }`
- `.text-3xl { }`
- `.text-4xl { }`
- `.text-5xl { }`
- `.text-6xl { }`

Text Align

- `.text-left { }`
- `.text-center{ }`
- `.text-right { }`
- `.text-justify{ }`

Text Color

- `.text-{color}-{shade(100-900)} { }`

Font Weight (Bold)

- `.font-hairline{ } // 100`
- `.font-thin{ } // 200`
- `.font-light{ } // 300`
- `.font-normal{ } //0`
- `.font-medium{ } // 500`
- `.font-semibold { } // 600`
- `.font-bold{ }// 700`
- `.font-extrabold { } // 800`

`.font-black{ } // 900`

Borders

These classes will color, stylize and add radius to any border or corners.

`.border { } // 1px`

`.border-0 { } //0`

`.border-2 { } //2px`

`.border-4 { } //4px`

`.border-8 { } // 8px`

Flexbox

Tailwind uses Flexbox for the layout of items on the document. Flexbox is a css display property that defines a flex container. Once a container has been assigned as a flex container, we can use all of the alignment utility classes to achieve the desired look.

Default direction - horizontal alignment

- `.justify-start{ }`
- `.justify-center{ }`
- `.justify-end { }`
- `.justify-between{ }`
- `.justify-around{ }`

Flex Direction

- `.flex-row { }`
- `.flex-row-reverse{ }`
- `.flex-col { }`
- `.flex-col-reverse{ }`

Wrapping

- `.flex-no-wrap{ }`
- `.flex-wrap { }`
- `.flex-wrap-reverse{ }`

Responsive Design

All modern applications should be able to responsively fit into the screen size.

Tailwind is a mobile-first framework, meaning that all of the classes that we have talked about thus far, are for mobile and trickle up to desktop. But we can change this with a couple of modifiers.

Default breakpoints [all] // 0px

- `.sm: // 640px`
- `.md: // 768px`
- `.lg: // 1024px`
- `.xl: // 1280px`

Default responsive classes

<code>.sm:bg-* { }</code>	// background color
<code>.sm:w-* { }</code>	// width
<code>.sm:h-* { }</code>	// height
<code>.sm:p-* { }</code>	// padding
<code>.sm:m-* { }</code>	// margin
<code>.sm:font-sans { }</code>	//fontfamily - sans, serif, mono
<code>.sm:text-lg { }</code>	// font size - xs, sm, base, lg, xl, {2-6}xl
<code>.sm:text-left { }</code>	// left, center, right, justify
<code>.sm:text-{color}-{shade(100-900)} { }</code>	// text color
<code>sm:font-bold { }</code>	// font weight
<code>.sm:tracking-tighter { }</code>	//letter spacing
<code>.sm:leading-tight { }</code>	// line spacing/height
<code>.sm:uppercase { }</code>	// text transform
<code>.sm:border-{color}-{shade (100-900)}</code>	// border color
<code>.sm:border-{style} { }</code>	// border style
<code>.sm:border-width { }</code>	// border width

<code>.sm:rounded-{size} {}</code>	<code>// border radius</code>
<code>.sm:{display} {}</code>	<code>// block, inline, flex, table, hidden...</code>
<code>.sm:flex {}</code>	<code>// display flex</code>
<code>.sm:flex-{collrow} {}</code>	<code>// flex direction</code>

VERSION CONTROL

What is Version Control?

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

Version control systems record changes in a special database called a repository. Each change to the code is called a commit or revision. Each revision is assigned a unique ID which allows referencing and rolling back to specific versions later on.

Benefits Of using Version Control

Collaboration - Version control enables multiple developers to work on the same codebase simultaneously without interfering with each other's work. Changes made by different developers can be merged intelligently.

History and Versioning - Version control maintains logs of all changes made to the code, who made the changes, and when they were made. Developers can refer back to or restore previous versions of the code.

Restore and Rollback - If bugs are introduced, developers can revert the codebase back to a previous known good state. Files can also be restored after accidental deletion or corruption.

Branching - Branches allow developers to work on new features or test out bug fixes in isolation. Multiple branches can be created for different streams of work to avoid disrupting the main code.

Traceability - Detailed logs show who made what changes when, improving transparency and accountability. Finding the root cause of issues becomes easier.

Integrations - Version control integrates well with other DevOps tools like bug tracking, CI/CD, automated testing etc. This leads to more efficient development workflows.

Faster Onboarding - New team members can get up to speed by looking at the code history and changes made by other developers.

Code Reviews - Code reviews and approvals become easier by looking at changed lines and associated explanations.

Experimental Edits - Developers can experiment boldly with changes. If the changes do not work out, they can simply rollback without side effects.

Disaster Recovery - Version controlled code repositories act as backup stores if code is lost due to system failures, human errors etc.

Summary:

To summarize, version control is essential for collaborating and managing changes when building software. It provides key capabilities like history, backups, collaboration and branching to enable developers build software more efficiently.

GitHub Repository

Git:

GitHub is built on top of Git, a distributed version control system. Git allows multiple developers to work on a project simultaneously without interfering with each other's work.

Repository:

In GitHub, a repository (repo) is a collection of files and the entire history of changes to those files. Repositories can be public or private, depending on whether you want to share your code

with the public or keep it private.

Collaboration:

GitHub facilitates collaboration by enabling multiple developers to work on the same project. Developers can clone a repository, make changes, and then submit those changes for review and incorporation into the main codebase.

Branching:

Developers can create branches to work on features or fixes independently.

Branches allow for parallel development without affecting the main codebase until changes are ready to be merged.

Pull Requests:

Pull Requests (PRs) are proposed changes submitted by a developer who has made changes in their branch.

PRs are reviewed by others, and once approved, the changes can be merged into the main branch.

Issues:

GitHub Issues are used to track and discuss tasks, enhancements, bugs, or other types of questions or problems within a repository. Issues can be assigned to specific individuals and labelled for better organization.

Importance of GitHub**Code Versioning:**

GitHub provides a robust version control system, allowing developers to track changes, revert to previous states, and collaborate effectively.

Collaboration and Teamwork:

GitHub enables seamless collaboration among team members, fostering a sense of community and teamwork.

Code Review:

Pull Requests and code reviews on GitHub ensure that changes are thoroughly examined before merging into the main codebase, maintaining code quality.

Documentation:

GitHub repositories often include documentation, README files, and wikis that provide crucial information for users and developers.

Continuous Integration/Continuous Deployment (CI/CD):

GitHub integrates with CI/CD tools, automating testing and deployment processes for increased efficiency and reliability.

Open Source Contributions:

GitHub is a hub for open-source projects, allowing developers worldwide to contribute to various projects and learn from each other.

Project Management:

GitHub offers project management tools such as boards, milestones, and labels, helping teams organize and prioritize tasks.

Community Support:

Developers can seek help, share knowledge, and collaborate with the broader community through discussions, issues, and contributions.

Note: GitHub is a powerful platform that supports collaborative software development, version control, and project management, making it an integral part of modern software development workflows.

Manage Codes

1. Git Basics:

Repository (Repo):

A repository is a container for your project and includes all of its files and version history.

To create a new repository, use the `git init` command locally or create one on GitHub and clone it to your local machine.

Cloning a Repository:

Use **git clone** to copy a repository from GitHub to your local machine.

Example: `git clone https://github.com/username/repository.git`

Committing Changes:

After making changes to your files, use `git add` to stage changes and `git commit` to save them to the version history.

Branching:

Create a new branch using `git branch` and switch to it with `git checkout` or use `git checkout -b` to create and switch in one command.

Example: `git checkout -b new-feature`

Merging:

Merge changes from one branch into another using `git merge`.

Pushing Changes:

Upload local changes to GitHub using `git push`.

Example: `git push origin main`

2. GitHub Features for Code Management:

Pull Requests (PRs):

Open a pull request to propose changes from your branch to the main branch.

PRs facilitate code review and collaboration.

Code Review:

GitHub's interface allows reviewers to comment on specific lines of code, suggest changes, and approve or request changes to a pull request.

Branch Protection:

Protect branches to prevent direct commits or force pushes, ensuring that changes go through the pull request process.

Forks and Upstream:

Fork a repository to create a personal copy. The original repository is referred to as the "upstream" repository.

Code Searching:

Utilize GitHub's search functionality to find code snippets, files, or specific terms within repositories

Blame and History:

Use `git blame` or GitHub's web interface to see who made changes to a specific line of code and view the commit history.

3. Git Tags and Releases:

Tags:

Tags are references to specific points in Git history, often used to mark release points.

Create a tag with `git tag`.

Releases:

GitHub allows you to create releases, associating them with a tag. Releases often include release notes and downloadable assets.

4. Project Management on GitHub:

Issues:

Use issues to track tasks, bugs, or feature requests. Issues can be assigned, labeled, and linked to pull requests.

Projects:

GitHub Projects provide boards to organize and prioritize work. They can be used for sprint planning, bug tracking, and more.

Wiki:

Repositories can have a Wiki for collaborative documentation.

5. Collaboration:

Collaborators:

Add collaborators to your repository with different levels of access (read, write, or admin).

Teams:

Organize collaborators into teams for easier access management.

6. Repository Settings:

Access Controls:

Manage permissions, branch protection, and repository visibility in the repository settings.

Webhooks:

Set up webhooks for events like pushes, issues, or pull requests to trigger external actions.

Git Ignore and Git Attributes:

.gitignore:

Create a `.gitignore` file to specify files or directories to be ignored by Git.

.gitattributes:

Use `gitattributes` to control attributes such as line endings and merge strategies.

WEB HOSTING

Web hosting refers to the service of providing storage space and access for websites on servers that are connected to the internet. Web hosting companies offer various plans and services to store website files, databases, and other resources, making websites accessible to users worldwide.

What is Web Hosting?

Web hosting is a service that allows organizations and individuals to post a website or web page onto the Internet.

Web hosting is a necessity for any website - it is the physical location of your website on the Internet, an online storage center that houses the information, images, video, and other content that comprises your website.

A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web. Web hosts are companies that provide space on a server owned or leased for use by clients, as well as providing Internet connectivity, typically in a data center.

Web hosting is the place where all the files of your website live. It is like the home of your website where it actually lives.

The images briefly describing web host is shown in Fig-9.1



Fig-9.1: Web Hosting

In a nutshell, web hosting is the process of renting or buying space to house a website on the World Wide Web. Website content such as HTML, CSS, and images has to be housed on a server to be viewable online.

When a hosting provider allocates space on a web server for a website to store its files, they are hosting a website. Web hosting makes the files that comprise a website (code, images, etc.) available for viewing online. Every website you've ever visited is hosted on a server.

Types of Hosting:

Shared hosting

Shared hosting is perfect for entry-level website hosting. This is where your website will be stored on the same server as multiple other websites. With a shared hosting plan, all domains share the same server resources, such as RAM (Random Access Memory) and CPU (Central Processing Unit). However, because all resources are shared, the cost of shared hosting plans are relatively low, making them an excellent option for website owners in their beginning stages.

Virtual private server (VPS) hosting

A VPS hosting plan is the ultimate middle ground between a shared server and a dedicated server. It's ideal for website owners that need more control, but don't necessarily need a dedicated server.

VPS hosting is unique because each website is hosted within its own space on the server, though it still shares a physical server with other users. While VPS hosting provides website owners with more customization and storage space, they're still not able to handle incredibly high traffic levels or spikes in usage meaning that the site performance can still be affected by other sites on the server.

Dedicated server hosting

Dedicated hosting gives website owners the most control over the server that their website is stored on. That's because the server is exclusively rented by you and your website is the only one stored on it. This means that you have full root and admin access, so you can control everything from security to operating system that you run.

What is Domain/ Domain Name?

- Domain name is the address of your website that people type in the browser URL bar to visit your website.
- In simple terms, if your website was a house, then your domain name will be its address.
- A domain name can be any combination of letters and numbers, and it can be used in combination of the various domain name extensions, such as .com, .net and more.
- The domain name must be registered before you can use it. Every domain name is unique. No two websites can have the same domain name. If someone types in www.yourdomain.com, it will go to your website and no one else's.
- The Internet is a giant network of computers connected to each other through a global network of cables. Each computer on this network can communicate with other computers.
- To identify them, each computer is assigned an IP address. It is a series of numbers that identify a particular computer on the internet. A typical IP address looks like this: 66.249.66.1
- Now an IP address like this is quite difficult to remember. Imagine if you had to use such numbers to visit your favourite websites.
- Domain names were invented to solve this problem.
- Now if you want to visit a website, then you don't need to enter a long string of numbers. Instead, you can visit it by typing an easy to remember domain name in your browser's address bar.

For example, www.google.com.

A domain name is unique to your website (just like a fingerprint), and cannot be shared between different websites.

Steps For Hosting

Step 1: Open website of any hosting company.

With that out of the way, the most popular global registrars include:

- Namecheap
- GoDaddy
- Name.com
- Domain.com
- Bluehost.com

Step 2: Create Your account.

Once you've had a look at the top hosting companies, you need to pick one! There are a number of features you should consider when choosing a website hosting company.

Below are some of the key factors to consider:

- Uptime
- Support
- Free domain name
- Value for Money
- Bandwidth
- WordPress integration
- Money-back guarantees

The below figure (Fig-9.2) shows GoDaddy website.

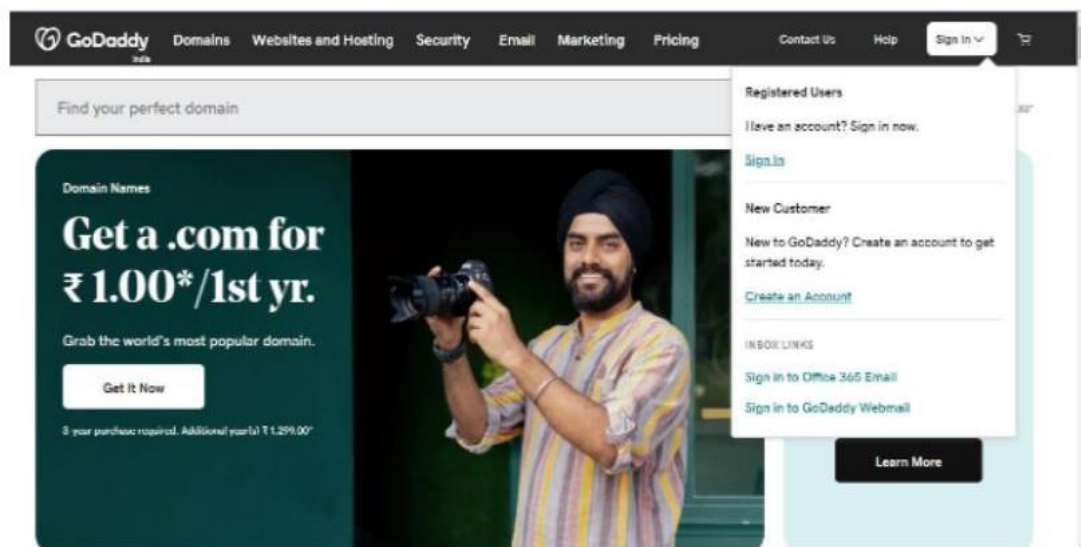


Fig-9.2: GoDaddy Website

Step 3: Choose a website hosting plan.

Once you have a host platform you like the look of, you still need to narrow your decision down to a plan. There are numerous types of hosting (shared, dedicated, VPS, and more), and there is usually a selection of plan tiers within each type.

Step 4: Register a domain name.

It's all well and good having a plot of internet land, but without an address no one will be able to find it! That's what a domain name is. It's your digital address.

Google's is www.google.com. You get the idea. Your site will need a domain as well.

Nearly every web hosting provider includes domain name registration in its signup process. Sometimes it's included as a freebie in the plan you've chosen (GoDaddy and Bluehost, for example it is shown in fig-9.3).

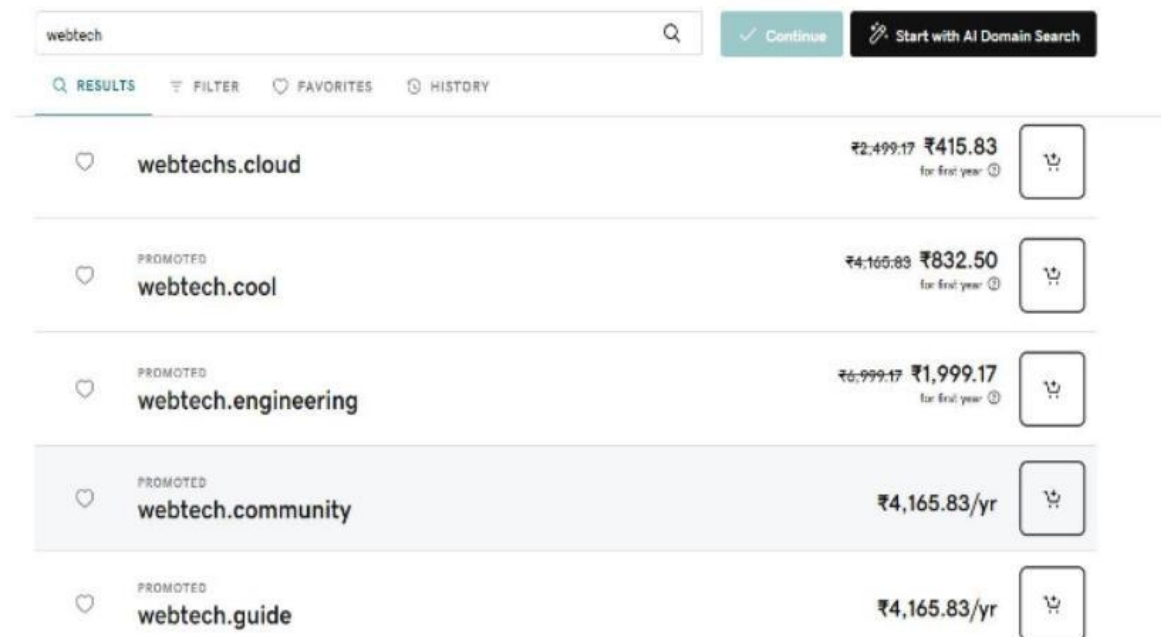


Fig-9.3: Page for registering domain name

A good domain is simple, easy to remember, and usually number- free. Don't feel bound to the .com

convention either. More and more sites are playing around with atypical top level domains like .xyz or .co. So long as it fits with your brand, you'll be fine.

If you already have a domain name it's simply a case of attaching it to your new server. Many hosting

platforms include this in the signup process and handle it for you.

Step 5: Choose domain from the list.

After login go to visit my account:

The visit my account page is shown in Fig-9.4.

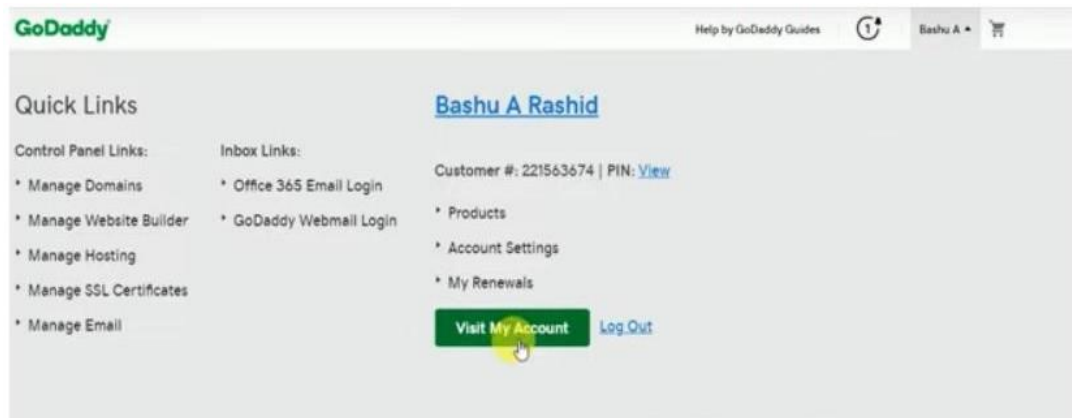


Fig-9.4: visit my account page button

Choose Web Hosting and Manage

The page to choose web hosting is shown in Fig-9.5.



Fig-9.5: Choosing web hosting and manage

Go to File Manager option

After clicking manage the screen appears as shown in Fig-9.6.

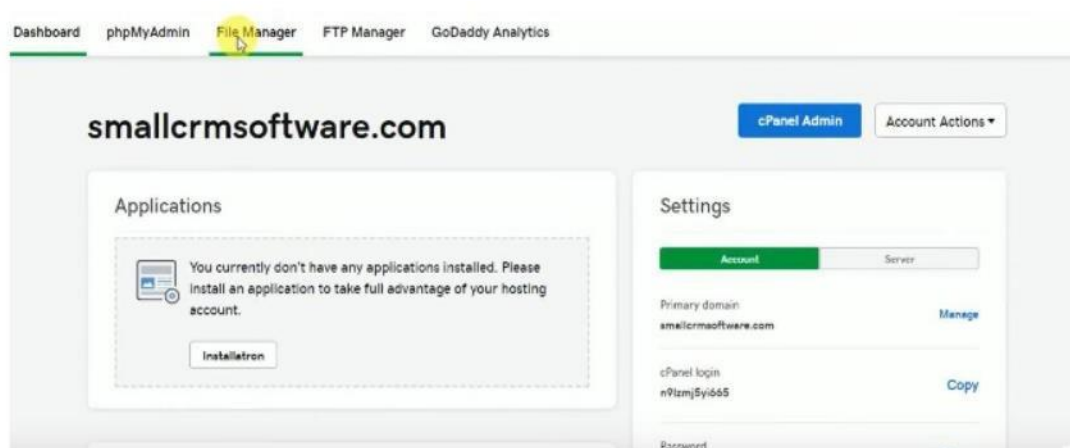


Fig-9.6 : Screen after clicking Manage button

In file manager find public html folder

The file manger screen is shown in Fig-9.7.

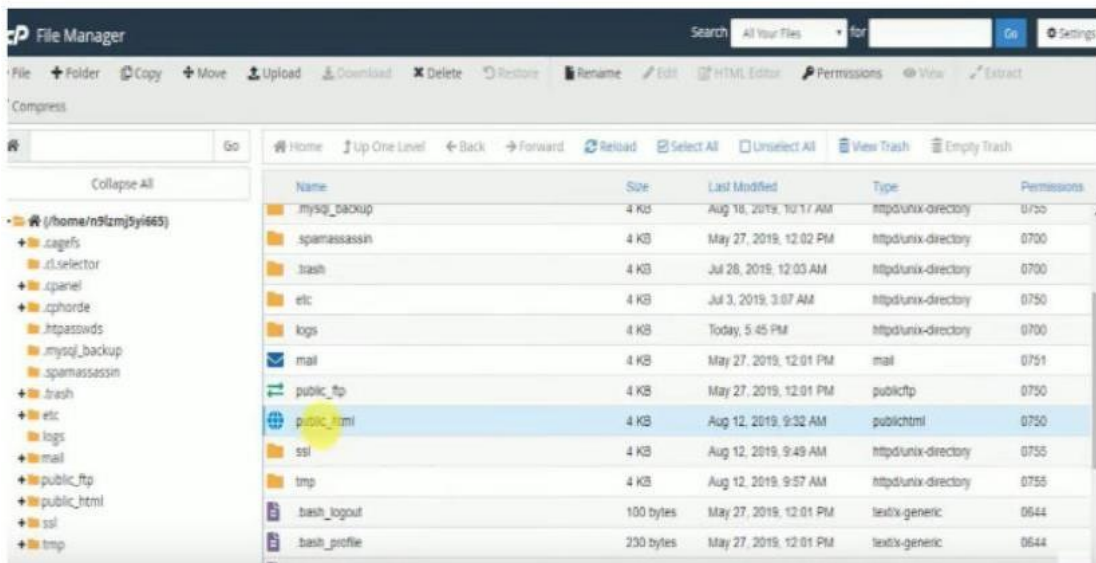


Fig-9.7: File manager

Then find your domain name folder and open it

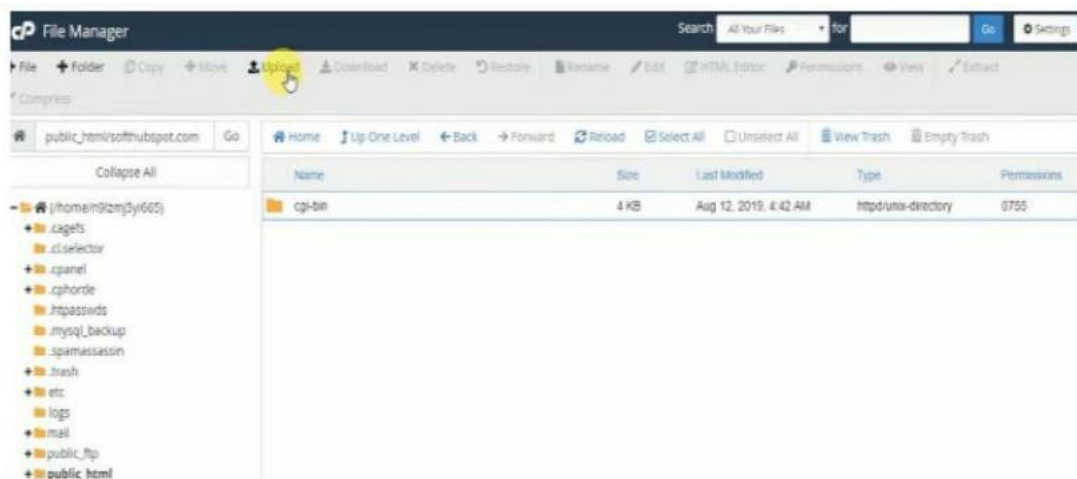


Fig-9.8. Choosing files

Choose the upload option and upload all files one-by-one as shown in Fig-9.8.

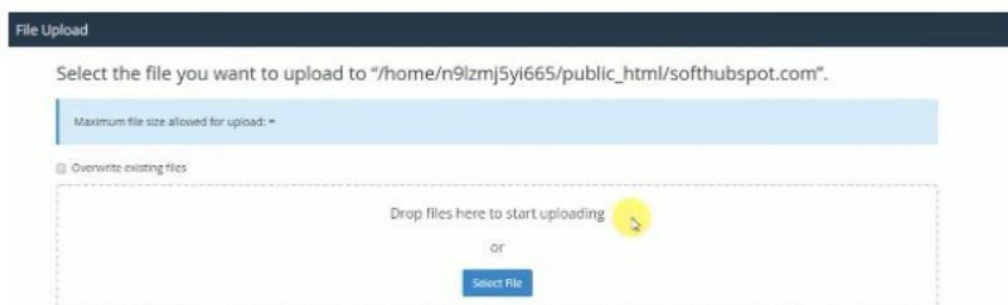


Fig-9.9: Uploading Files

After Uploading your files are uploaded like this as shown in Fig-9.9.

Open any browser and paste your web address and see the screen shown in Fig-10.

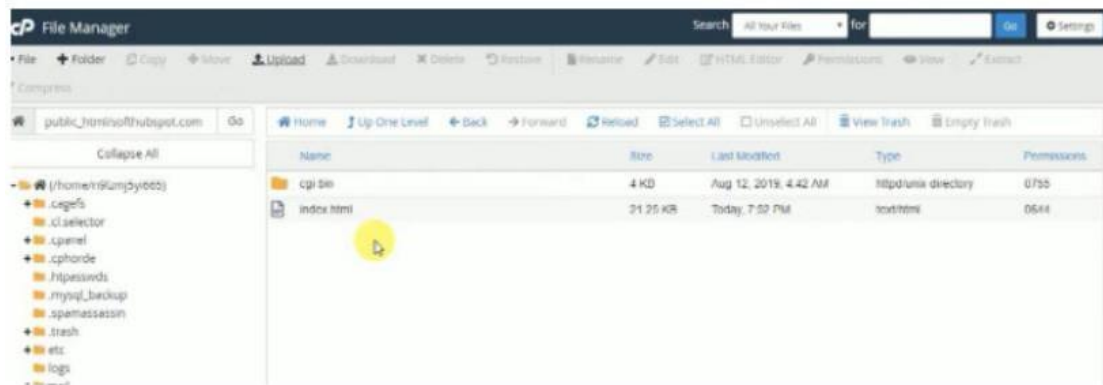


Fig-10: File Manager

SQL

SQL, or Structured Query Language, is a standardized programming language used for managing and manipulating relational databases.

OVERVIEW OF DATABASE AND RELATIONAL DATABASES

What is a Database?

A database is an organized collection of structured information or data, typically stored electronically in a computer system. It is usually controlled by a database management system (DBMS), which serves as an interface between the database and its end users or programs. This allows users to retrieve, update, and manage the data efficiently

- **Understanding Relational Databases**

It is a type of DBMS that organizes data into tables consisting of rows and columns. These tables are related to each other based on common attributes, enabling efficient data management and retrieval

- **Key Concepts in Relational Databases:**

Tables: In this, data is stored in the form of tables. Each table consists of rows and columns where rows represent individual records and columns represents attributes or fields.

Relationships: It establishes the connection between tables based on shared data. The most common type of relationship is foreign key-primary key relationship, where data in one table refers to data in another table.

Normalization: It is the process of organizing data in database to minimize redundancy and dependency.

Importance of RDBMS

- Data Integrity
- Flexibility
- Scalability
- Security

Here, we understand the database and relational database is the fundamental in today's data driven world. RDBMS like MYSQL provide powerful tools for storing, organizing and retrieving data efficiently.

Introduction to MYSQL

What is MYSQL?

- [MySQL](#) is an open-source relational database management system ([RDBMS](#)) developed by Oracle Corporation.
- It uses Structured Query Language ([SQL](#)) for database management and is known for its reliability, speed and ease of use.
- MySQL is widely used for various applications, from small websites to large-scale enterprise systems.

Key Features of MYSQL

Open Source: It is freely available for use and distribution

Cross-Platform Compatibility: It compatible with various OS

Scalable: It refers the ability of systems to work easily with small amounts of data, large amounts of data, clusters of machines.

INSTALLATION OF MYSQL

Download MySQL by visiting official website of MYSQL

<https://dev.mysql.com/downloads/>.

- Choose the appropriate MYSQL Community server addition for your OS.
- Click on the download button to initiate the download process.

CRUD OPERATIONS:

Creating Databases:

In MYSQL you can create a new database using CREATE DATABASE statement followed by desired database name.

- CREATE Database: To create a specific database

Ex: CREATE DATABASE database;

- Selecting Database: To work with specific database, you need to select it using the USE statement

USE database_name;

- Dropping the database: To delete a database and its contents

DROP DATABASE database_name

Exercise caution when you using this command, as it permanently deletes all in specific data

Example for creating table is shown in Fig-11.

```
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(50) NOT NULL,  
    email VARCHAR(100) UNIQUE  
);
```

Fig-11: Creating table 'users'

In this, we create a table with table name user and id as a primary key and integer column with auto-increment attribute, which automatically assigns unique value to each row, email and name is variable character with the data type var not null means it should not be empty whereas unique ensures that all the emails are entered into the database differently

In this, we are essential tasks in MYSQL database administration. By mastering these operations, you can effectively organize and store data in MYSQL database.

Retrieve: Used to retrieve data from table and database.

- RETRIEVE : SELECT col1,col2,... FROM TABLE_NAME;

Update: Used to update the existing data in table.

- UPDATE: UPDATE table_name SET column1=value1, column2=value2 , ... WHERE condition;

These are CRUD operations to be performed on a table.

INDEXING AND OPTIMIZATION

Indexing: It is a database optimization technique used to improve the speed and efficiency of the data retrieval operations. An index in the data structure that organizes the values of one or more columns in a table, allowing for faster lookup and retrieval of data

CREATING AND MANAGING INDEXES:

Creating indexes: you can create an index on one or more columns of a table using CREATE INDEX statement.

Syntax:

CREATE INDEX index_name ON table_name(column1,...);

Managing indexes: You can manage indexes by creating, dropping, or altering them based on performance requirements of your database.

QUERY OPTIMIZATION TECHNIQUES:

Use indexes: Ensure that relevant columns are indexed to speed up query execution

Optimize Where Clause: Use appropriate conditions and indexing to filter rows efficiently

EXPLAIN STATEMENT AND QUERY ANALYSIS

EXPLAIN statement: It is used to analyse the execution plan of SQL query. It provides information about how MYSQL executes the query and helps identify potential performance.

SYNTAX:

EXPLAIN SELECT column1,column2,... FROM table _ name where condition;

STORED PROCEDURES AND FUNCTIONS

Stored Procedures: It is a precompiled collection of SQL statements stored in the database catalog. It can accept input parameters, perform operations, and return results to calling program.

Syntax:

CREATE [OR REPLACE] **PROCEDURE** procedure_name[(parameter [,parameter])]

IS

[declaration_section]

BEGIN

executable_section

[EXCEPTION

exception_section]

END [procedure_name];

Function: A function is a subroutine stored in the database catalog accepts that accepts parameters perform computations, and returns single value.

Syntax:

CREATE [OR REPLACE] **FUNCTION** function_name [parameters]

[(parameter_name [IN | OUT | IN OUT] type [, ...])]

RETURN return_datatype

{**IS** | **AS**}

BEGIN

< function_body >

END [function_name];

Security and User Management

- **Authentication:** Verifies that clients are allowed to connect as a certain user.
- **Authorization:** Determines what actions an account is permitted to perform after authentication
- **Granting Privilege:** you can grant the privilege using GRANT statement
Syntax: GRANT privilege on database _ name.table _ name to 'username'@'host'
- **Revoking privilege:** to revoke it from the user to the REVOKE statement
Syntax: REVOKE privilege on database _ name.table _ name to 'username'@'host'

- Viewing privileges: you can view the privilege assigned to the user using the SHOW GRANTS statements.

Syntax: SHOW GRANT privilege FOR 'username'@'host'

Securing the MYSQL Server:

Securing your MySQL database is vital to safeguard sensitive data and maintain application integrity. Here's how to do it effectively:

- Use Strong Passwords: Make sure all MySQL users have tough, unique passwords with a mix of characters. Regularly update and rotate passwords for added security.
- Keep MySQL Updated: Always install the latest patches and versions to fix any security holes. This ensures you're protected against known vulnerabilities.
- Limit User Permissions: Give users only the permissions they need. Avoid using the powerful root account for everyday tasks. Instead, create specific accounts with restricted access.
- Enable Firewall Protection: Set up a firewall to control access to your MySQL server. Only allow connections from trusted sources to prevent unauthorized access.
- Encrypt Your Data: Protect sensitive data both at rest and in transit using encryption methods like [SSL/TLS](#). This ensures that even if someone gains access, they can't read the data.
- Regular Backups: Backup your databases regularly and store them securely. In case of data loss or corruption, you'll have a copy to restore from.
- Monitor and Audit: Keep an eye on your database's activity using monitoring tools. Regularly review logs for any suspicious behavior that could indicate a security breach.
- Disable Remote Root Access: Prevent remote access to the powerful root account to minimize the risk of unauthorized access. Instead, use secure methods like SSH tunneling for remote access.
- Remove Unused Accounts and Databases: Regularly clean up unused accounts and databases to reduce the potential entry points for attackers. This minimizes the attack surface and enhances security.

MongoDB

Module 1: Introduction and Setup

1. What is MongoDB?
2. Installing MongoDB
3. Key Concepts

1.1 What is MongoDB?

MongoDB is a NoSQL database program that uses a document-oriented data model. Unlike traditional relational databases, MongoDB stores data in flexible, JSON-like documents. This allows for more dynamic and scalable data storage, especially useful for handling large volumes of unstructured data.

Key Features of MongoDB:

- Document-oriented: Stores data in JSON-like documents.
- Schema-less: No fixed schema, allowing for flexible data structures.
- Scalable: Can handle large volumes of data with ease.
- High Performance: Optimized for read and write operations.
- Cross-platform: Available for Windows, MacOS, and Linux

1.2 Installing MongoDB

Step-by-Step Installation:

1. Download MongoDB:

Go to the official MongoDB website: [MongoDB Download Center](https://www.mongodb.com/download-center) Select the appropriate version for your operating system.

2.Install MongoDB:

Windows:

- Run the downloaded installer (.msi file).
- Follow the on-screen instructions to complete the installation.

MacOS:

- Open the downloaded .tgz file.
- Follow the instructions in the readme file to install MongoDB.
- Alternatively, use Homebrew: `brew tap mongodb/brew` and then `brew install mongodb-community`.

Linux:

- Follow the instructions on the MongoDB website for your specific Linux distribution (Ubuntu, Debian, CentOS, etc.).

3.Set up MongoDB:

- Create the data directory where MongoDB will store its data. For example, on Windows, you might use `C:\data\db`, and on Linux/MacOS, you might use `/data/db`.
- Make sure the directory has the appropriate permissions

4.Run MongoDB:

- Open a terminal or command prompt.
- Navigate to the directory where MongoDB is installed.
- Start the MongoDB server with the command: `mongod` (this will start server).
the database
- In another terminal or command prompt, start the MongoDB shell with the command: `mongo`

(this will open the MongoDB interactive shell).

1.3 Key Concepts

1. Databases and Collections:

- Database: A container for collections. Similar to a database in a relational database system.
- Collection: A group of MongoDB documents. Similar to a table in a database system.

2. Documents:

- A document is a set of key-value pairs (similar to JSON objects).
- Example of a document

```
{
  "name": "John Doe",
  "age": 30,
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "state": "CA"
  },
  "hobbies": ["reading", "travelling", "swimming"]
}
```

3. Schema Design:

- MongoDB is schema-less, meaning it does not require a fixed schema. This allows for flexibility in storing different types of data.
- Documents within a collection can have different fields.

4. CRUD Operations:

- **Create:** Insert new documents into a collection. `db.collection.insertOne({ name: "John Doe", age: 30 });`

- **Read:** Query documents in a collection. `db.collection.find({ age: 30 });`

- **Update:** Modify existing documents.

```
db.collection.updateOne({ name: "John Doe" }, { $set: { age: 31 } });
```

- **Delete:** Remove documents from a collection.

```
db.collection.deleteOne({ name: "John Doe" });
```

5. Indexes:

Indexes improve the speed of data retrieval operations.

- Created on fields that are frequently queried.
- `db.collection.createIndex({ name: 1 });`

6. Aggregation:

- Used for processing data and returning computed results.
- Aggregation pipeline: a framework for data aggregation modeled on the concept of data processing pipelines.

```
db.collection.aggregate([
  { $match: { age: { $gte: 30 } } },
  { $group: { _id: "$age", total: { $sum: 1 } } }
]);
```

Module 2: CRUD Operations and Indexes

- Creating Databases and Collections
- Inserting Documents
- Querying Documents
- Updating Documents
- Deleting Documents
- Indexes

2.1 Creating Databases and Collections

Creating Databases:

- A database is a collection of data organized in a structured way.
- To create a database, you simply need to use the 'use' command database name followed by the Database name.
- use myDatabase;
- This command will create a new database if it does not exist or switch to it if it does.

Creating Collections:

- A collection is analogous to a table in a relational database.
- Collections are created within a database to store documents.
- db.createCollection("myCollection");
- Alternatively, a collection is implicitly created when you insert the first document into it.

2.2 Inserting Documents

- Documents are records in a collection, usually in JSON format.
- To insert a document into a collection: db.myCollection.insertOne({ name: "John", age: 25, city: "New York" });
- This command will add a single document to the myCollection collection.
- You can also insert multiple documents at once.

2.3 Querying Documents

- Querying is used to retrieve data from a collection.
- To find all documents in a collection:
- db.myCollection.find();
- This command returns all documents in myCollection.
- You can also use comparison operators.
- db.myCollection.find({ age: { \$gt: 25 } });
- This command returns documents where the age is greater than 25.

2.4 Updating Documents

- Updating allows you to modify existing documents in a collection.
- To update a single document:
- db.myCollection.updateOne({ name: "John" }, { \$set: { age: 26 } });
- This command finds the document where the name is "John" and update the age to 26.
- To update multiple documents:
- db.myCollection.updateMany({ city: "New York" }, { \$set: { city: "San Francisco" } });
- This command updates the city field to "San Francisco" for all where the city is "New York".

2.5 Deleting Documents

- Deleting removes documents from a collection.
- To delete a single document:
`db.myCollection.deleteOne({ name: "John" });`
- This command deletes the first document where the name is "John".
- To delete multiple documents:
`db.myCollection.deleteMany({ age: { $lt: 25 } });`
- This command deletes all documents where the age is less than 25.

2.6 Indexes

Indexes improve the speed of query operations on a collection.

- To create an index on a field:
`db.myCollection.createIndex({ name: 1 });`
- This command creates an ascending index on the name field.
- You can also create a descending index:
`db.myCollection.createIndex({ age: -1 });`
- This command creates a descending index on the age field.
- Compound indexes can be created on multiple fields:
`db.myCollection.createIndex({ name: 1, age: -1 });`
- This command creates an index on both the name and age fields.

Module 3: Aggregation and Data Modeling

- Aggregation Framework
- Examples of Aggregation
- Data Modeling

3.1 Aggregation Framework

• Introduction to Aggregation Framework

Aggregation in databases involves processing data records and returning computed results. It's akin to the SQL GROUP BY clause but often more powerful and flexible. Aggregation operations group values from multiple documents and perform a variety of operations on the grouped data to return a single result.

Key Concepts

- **Stages:** Each stage transforms the documents as they pass through the pipeline.
- **Pipelines:** A sequence of stages through which data is processed.
- **Operators:** Used within stages to process data (e.g., `$match`, `$group`, `$sort`).

Common Aggregation Stages

- **`$match`:** Filters the documents to pass only the ones that match the specified condition(s).
- **`$group`:** Groups input documents by a specified identifier expression and applies the accumulator expressions.
- **`$sort`:** Orders the documents.
- **`$limit`:** Restricts the number of documents in the output.

Example

```
[
  { "$match": { "status": "active" } },
  { "$group": { "id": "$customerId", "total": { "$sum": "$amount" } } },
  { "$sort": { "total": -1 } },
```

```
{ "$limit": 10 }
]
```

In this example, we:

- Filter documents with status equal to "active".
- Group by customerId and sum up the amount.
- Sort the results in descending order of total.
- Limit the output to the top 10 customers.

3.2 Examples of Aggregation

Example 1: Calculating Average Age Consider a collection of student records:

```
[
  { "name": "Alice", "age": 23 },
  { "name": "Bob", "age": 24 },
  { "name": "Charlie", "age": 22 }
]
```

To calculate the average age:

```
[
  { "$group": { "id": null, "averageAge": { "$avg": "$age" } } }
]
```

Output:

```
[ { "_id": null, "averageAge": 23 } ]
```

Example 2: Summing Order Amounts

For a collection of orders:

```
[
  { "orderId": 1, "amount": 100 }, { "orderId": 2, "amount": 150 },
  { "orderId": 3, "amount": 200 }
]
```

To sum the order amounts:

```
[
  { "$group": { "id": null, "totalAmount": { "$sum": "$amount" } } }
]
```

Output:

```
[ { "_id": null, "totalAmount": 450 } ]
```

3.3 Data Modeling

Introduction to Data Modeling

Data modeling is the process of creating a data model for the data to be stored in a database. This model organizes the data and defines the relationships between them. It ensures that the data is stored in a way that is efficient and can be accessed easily.

Key Concepts

- **Entities and Attributes:** An entity represents a real-world object, and details about that entity.

attributes are the

- **Relationships:** Defines how entities are related to each other.
- **Normalization:** Process of organizing data to minimize redundancy.

Steps in Data Modeling

- **Identify Entities:** Determine the main objects (e.g., students, courses).
- **Define Relationships:** Establish how these entities are related (e.g., in courses).
students enroll.
- **Map Attributes:** Identify the attributes for each entity (e.g., student title).
name, course.
- **Normalization:** Organize attributes to reduce redundancy (e.g., separating student details from course details).

Example

For a university database:

- Entities: Students, Courses, Enrollments
- Attributes:
 - Students: studentId, name, age
 - Courses: courseId, title, credits
 - Enrollments: enrollmentId, studentId, courseId, dateEnrolled

Relationships:

- A student can enroll in multiple courses.
- A course can have multiple students.



VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY
(An Autonomous Institution Affiliated to JNTUK, Kakinada
Approved by AICTE New Delhi - Accredited by NBA,
NAAC with 'A' Grade and ISO 9001:2008 Certified)
NAMBUR – 522508, Guntur, AP

SUPERVISOR EVALUATION OF INTERNSHIP RUBRIC

Student Name:

Host Organization/Company:

Internship Supervisor:

Date of Evaluation:

Note: The external assessment evaluated by the committee consisting of HoD, senior faculty, supervisor concerned and external Examiner. There shall be no internal marks for Summer Internship.

External Assessment:

Report Preparation: 20 Marks (40%)

Presentation & Viva-Voce: 30 Marks (60%)

: 50 Marks

The purpose of this assessment is to provide the student intern with constructive feedback on his/her internship experience. This evaluation form should be completed by the internship site supervisor or the individual who is most responsible for supervising the intern's work assignments.

The student's grade is partially based on your evaluation of his/her/their performance on each of the internship dimensions identified below. Use the evaluation rubric to assess the student's performance on each dimension by specifying a score based on the performance ratings and descriptors delineated in the rubric form. Candid and objective comments about the student's performance are also appreciated. Please add your relevant comments in the space provided in the form.

Quality of Work: The degree to which the student's work is thorough, accurate, and completed in a timely manner.

Ability to Learn: The extent to which the student asks relevant questions, seeks out additional information from appropriate sources, understands new concepts/ideas/work assignments, and is willing to make needed changes and improvements.

Initiative and Creativity: The degree to which the student is self-motivated, seeks out challenges, approaches and solves problems on his/her own, and develops innovative and creative ideas/solutions/options.

Character Traits: The extent to which the student demonstrates a confident and positive attitude, exhibits honesty and integrity on the job, is aware of and sensitive to ethical and diversity issues, and behaves in an ethical and professional manner.

Dependability: The degree to which the student is reliable, follows instructions and appropriate procedures, is attentive to detail, and requires supervision.

Attendance and Punctuality: The degree to which the student reports to work as scheduled and on-time.

Organizational Fit: The extent to which the student understands and supports the organization's mission, vision, and goals; adapts to organizational norms, expectations, and culture; and functions within appropriate authority and decision-making channels.

Response to Supervision: The degree to which the student seeks supervision, when necessary, is receptive to constructive criticism and advice from his/her supervisor, implements suggestions from his/her supervisor, and is willing to explore personal strengths and areas for improvement.

Supervisor Evaluation of Internship - Grading Rubric							
Evaluation Dimensions	Performance Rating						Score
	Needs Improvement		Meets Expectations		Excellent		
	1	2	3	4	5	6	
Internship Evaluation Dimensions - Grading Criteria							
Quality of Work	Work was done in a careless manner and was of erratic quality; work assignments were usually late and required review; made numerous errors		With a few minor exceptions, adequately performed most work requirements; most work assignments submitted in a timely manner; made occasional errors		Thoroughly and accurately performed all work requirements; submitted all work assignments on time; made few if any errors		
	Comments:						
Ability to Learn	Asked few if any questions and rarely sought out additional information from appropriate sources; was unable or slow to understand new concepts, ideas, and work assignments; was unable or unwilling to recognize mistakes		In most cases, asked relevant questions and sought out additional information from appropriate sources; exhibited acceptable understanding of new concepts, ideas, and work assignments; was usually willing to		Consistently asked relevant questions and sought out additional information from appropriate sources; very quickly understood new concepts, ideas, and work assignments; was always willing to take		

	and was not receptive to making needed changes and improvements	take responsibility for mistakes and to make needed changes and improvements	responsibility for mistakes and to make needed changes and improvements	
	Comments:			
Initiative and Creativity	Had little observable drive and required close supervision; showed little if any interest in meeting standards; did not seek out additional work and frequently procrastinated in completing assignments; suggested no new ideas or options	Worked without extensive supervision; in some cases, found problems to solve and sometimes asked for additional work assignments; normally set his/her own goals and, in a few cases, tried to exceed requirements; offered some creative ideas	Was a self-starter; consistently sought new challenges and asked for additional work assignments; regularly approached and solved problems independently; frequently proposed innovative and creative ideas, solutions, and/or options	
	Comments:			
Character Traits	Regularly exhibited a negative attitude; was dishonest and/or showed a lack of integrity on several occasions; was unable to recognize and/or was insensitive to ethical and diversity issues; displayed significant lapses in ethical and professional behavior	Except in a few minor instances, demonstrated a positive attitude; regularly exhibited honesty and integrity in the workplace; was usually aware of and sensitive to ethical and diversity issues on the job; normally behaved in an ethical and professional manner	Demonstrated an exceptionally positive attitude; consistently exhibited honesty and integrity in the workplace; was keenly aware of and deeply sensitive to ethical and diversity issues on the job; always behaved in an ethical and professional manner	
	Comments:			
Dependability	Was generally unreliable in completing work assignments; did not follow instructions and procedures promptly or accurately; was careless, and work needed constant follow-up; required close supervision	Was generally reliable in completing tasks; normally followed instructions and procedures; was usually attentive to detail, but work had to be reviewed occasionally; functioned with only moderate supervision	Was consistently reliable in completing work assignments; always followed instructions and procedures well; was careful and extremely attentive to detail; required little or minimum supervision	

	Comments:			
Attendance and Punctuality	Was absent excessively and/or was almost always late for work	Was never absent and almost always on time; or usually reported to work as scheduled, but was always on time; or usually reported to work as scheduled and was almost always on-time	Always reported to work as scheduled with no absences and was always on-time	
	Comments:			
Organizational Fit	Was unwilling or unable to understand and support the organization's mission, vision, and goals; exhibited difficulty in adapting to organizational norms, expectations, and culture; frequently seemed to disregard appropriate authority and decision-making channels	Adequately understood and supported the organization's mission, vision, and goals; satisfactorily adapted to organizational norms, expectations, and culture; generally functioned within appropriate authority and decision-making channels	Completely understood and fully supported the organization's mission, vision, and goals; readily and successfully adapted to organizational norms, expectations, and culture; consistently functioned within appropriate authority and decision-making channels	
	Comments:			
Response to Supervision	Rarely sought supervision when necessary; was unwilling to accept constructive criticism and advice; seldom if ever implemented supervisor suggestions; was usually unwilling to explore personal strengths and areas for improvement	On occasion, sought supervision when necessary; was generally receptive to constructive criticism and advice; implemented supervisor suggestions in most cases; was usually willing to explore personal strengths and areas for improvement	Actively sought supervision when necessary; was always receptive to constructive criticism and advice; successfully implemented supervisor suggestions when offered; was always willing to explore personal strengths and areas for improvement	
	Comments:			

Evaluator contentment: Based on the student's overall performance, a rating will be given

Performance Rating	
Good	Excellent
1	2

Summary Performance Ratings on Internship	
Evaluation Criteria	Score (From above)
Quality of Work	
Ability to Learn	
Initiative and Creativity	
Character Traits	
Dependability	
Attendance and Punctuality	
Organizational Fit	
Response to Supervision	
Evaluator contentment	
Total Score	

Overall Performance Evaluation of Student Intern				
Outstanding	Very Good	Satisfactory	Marginal	Unsatisfactory
Comments:				

we have reviewed this evaluation with the student intern.	Yes	No
If yes, the date of review:	Date of Review	
Comments:		

Signature of Committee members:

- 1.
- 2.
- 3.
- 4.