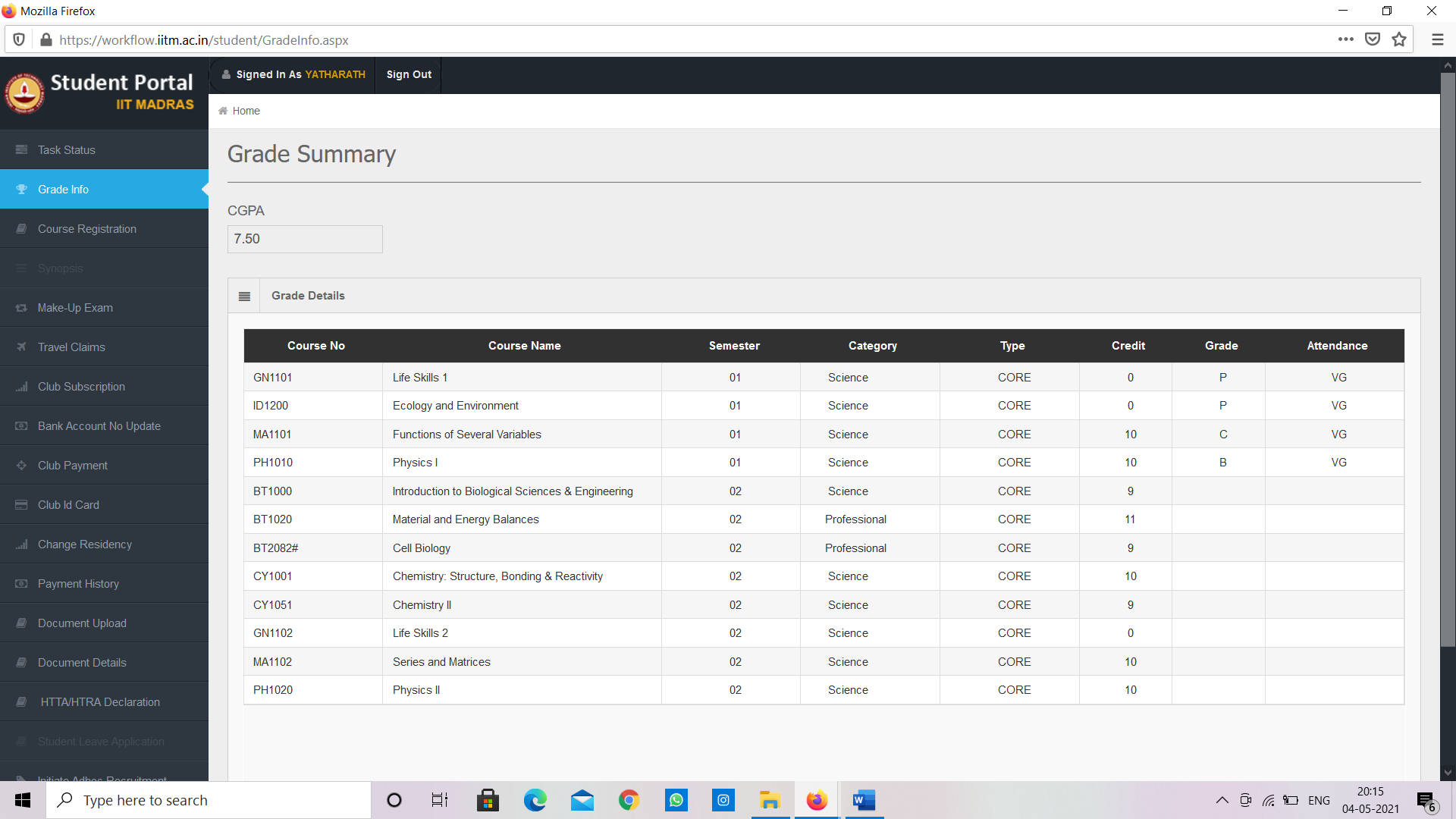
**Name:** Yatharath

**Roll No:** BS20B039

**PHONE:** 9958158847

**CGPA:** 7.5

**Email:** [bs20b039@smail.iitm.ac.in](mailto:bs20b039@smail.iitm.ac.in)



* **Question 1**

What do you think are the pros and cons of the CFI website? Suggest some improvement.

* CFI website is built in a complete interactive and user-friendly way**.**

**PROS**

1. The website contains info about all the club’s competitive teams and their achievements.

2.Website can help clubs in getting sponsorships.

3. Ensures [Communication](https://honestproscons.com/pros-and-cons-of-websites/#3_Communication)

**CONS of a general website**

[1. Risks Privacy](https://honestproscons.com/pros-and-cons-of-websites/#1_Risks_Privacy)

[2. Encourages Illegal Activity](https://honestproscons.com/pros-and-cons-of-websites/#2_Encourages_Illegal_Activity)

[3. Dependence & Less Productivity](https://honestproscons.com/pros-and-cons-of-websites/#3_Dependence_Less_Productivity)

[4. Increase Expenses](https://honestproscons.com/pros-and-cons-of-websites/#4_Increase_Expenses)

[5. Children Are Vulnerable and Exposed](https://honestproscons.com/pros-and-cons-of-websites/#5_Children_Are_Vulnerable_and_Exposed)

[6. Online Addiction](https://honestproscons.com/pros-and-cons-of-websites/#6_Online_Addiction)

I don’t think CFI website has any of these cons.

* Question 2
* Rate yourself based on the knowledge of HTML, CSS, JavaScript and briefly talk about some projects you’ve worked on and the approach you took from start to finish?
* HTML (hypertext mark-up language) as the name suggest is a mark-up language and is basically used to structure the content on a webpage.
* CSS (cascading style sheets) is on the other hand a set of instructions written to make our webpage look more appealing.
* Not exactly a project but I have built some basic web pages.

Some basic web pages that I have built so far,

1. User registration form.
2. Basic web page with a background and attached buttons like home button, about button and etc.

So far, I have learned everything from YouTube as everything which I learned about HTML in school had vanished by this time. I referred to the channel The Net Ninja, this channel has the kind of content which even a layman can refer to. It helped me a lot in learning HTML and CSS. Other than the two above webpages and have learned many HTML properties like fixed sicky and absolute. The deadline of filling out this application had drivenme from the very beginning to the end.

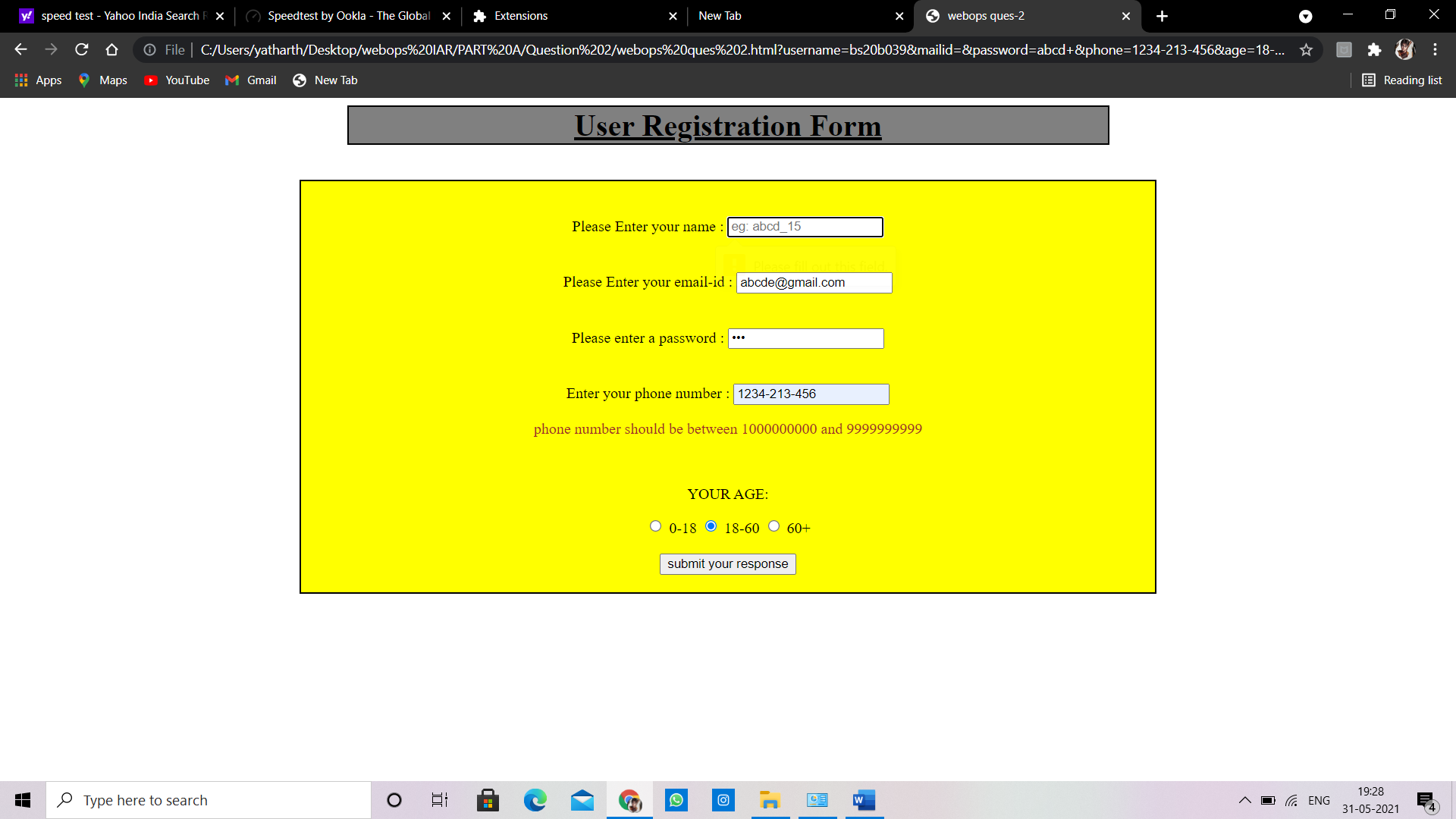
Description of user registration form.

* This is a basic HTML webpage designed for user to enter his/her details.
* I have basically used many a label tags for their specific ids so that I can create areas where user can enter his/her details.
* I have divided the body of the code using div tags and have made appropriate classes so that I can modify them in the style sheet.
* Place holder, pattern and the required property have been used. Basically, when user does not any field and submit the form a message pops that this field is required that’s the required property. Place holder property just makes the page more user friendly. Pattern property sets up the patten for entering phone number.
* I have also added age button and submit button by taking radio and submit as input types.

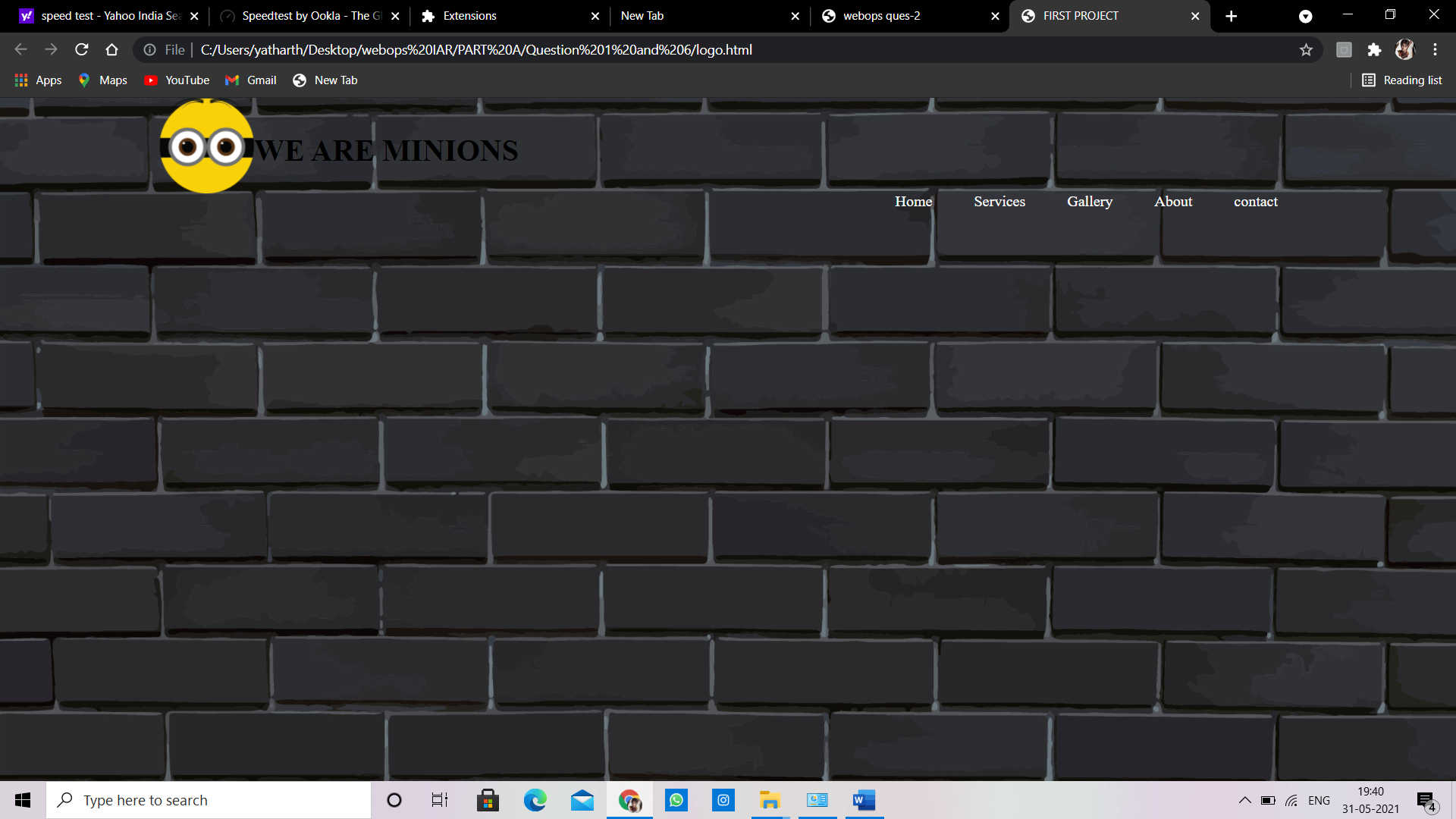
Description for the web-page logo and some buttons attached

* This is a basic webpage containing some important buttons, a logo, a brand name and an attractive background.
* I have first made an unordered list, so a as to make it horizontal inline property have been used.
* I have used anchor tags in the list elements giving them a hyperlink # so that they don’t direct us anywhere.
* I have used some classes so that I can decorate the webpage accordingly.
* I have used title property in anchor tags that make them user friendly by adding some description to buttons.

So, here is the form.



And here is the logo Webpage

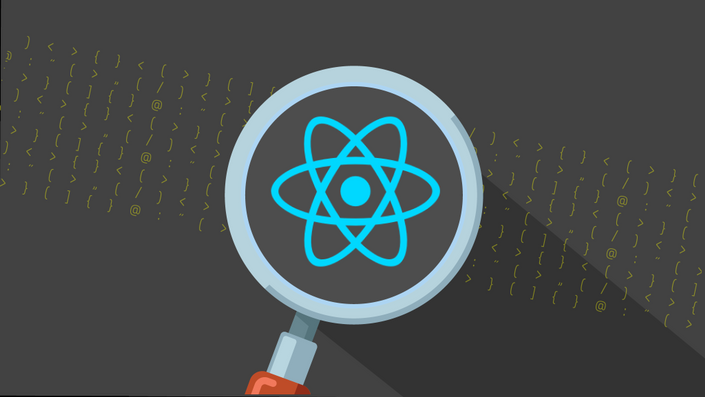


* Question 3

List out frameworks you are familiar with for website frontend development and backend development? Describe the frameworks in detail. (If you are not familiar with any frameworks, list out the available frameworks and describe them).

Q3) A)- **Frontend Frameworks: -**

1. **React.js [Frontend + JavaScript]:-**

* ReactJS is JavaScript library used for building reusable UI components.
* React is a library for building composable user interfaces.
* It encourages the creation of reusable UI components, which present data that changes over time.
* Lots of people use React as the V in MVC.
* React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.
* **React Features: -**
* **JSX** − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
* **Components** − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
* **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
* **Advantages:**-
* Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
* Can be used on client and server side as well as with other frameworks.

1. **Angular [Frontend + JavaScript]:-**

* Angular JS is an open source framework built over JavaScript.
* It was built by the developers at Google.
* This framework was used to overcome obstacles encountered while working with Single Page applications.
* **Angular Features: -**
* **Components −** Components help to build the applications into many modules. This helps in better maintaining the application over a period of time.
* **TypeScript −** The newer version of Angular is based on TypeScript. This is a superset of JavaScript and is maintained by Microsoft.
* **Services −** Services are a set of code that can be shared by different components of an application.
* **Advantages: -**
* It provides the capability to create Single Page Application in a very clean and maintainable way.
* It provides data binding capability to HTML.
* it gives user a rich and responsive experience.
* Easy to customize.

**(B) Backend frameworks: -**

**1.Django [Frontend + Backend + Python]:-**

* Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.
* Django makes it easier to build better web apps quickly and with less code.
* Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.
* **Django Features: -**
* **Rapid Development: -** Django was designed with the intention to make a framework which takes less time to build web application.
* **Secure: -** Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, etc.
* **Scalable: -** Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.
* **Versatile: -** Django is versatile in nature which allows it to build applications for different-different domains.
* **Advantages: -**
  + **Multilingual Support** − Django supports multilingual websites through its built-in internationalization system. So, we can develop our website, which would support multiple languages.
  + **Framework Support** − Django has built-in support for Ajax, RSS, Caching and various other frameworks.
  + **Administration GUI** − Django provides a nice ready-to-use user interface for administrative activities.
  + **Development Environment** − Django comes with a lightweight web server to facilitate end-to-end application development and testing.

 **2)Node.js [Backend + JavaScript]: -**

* 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.
* **NodeJS Features: -**
* **Single Threaded**: - It is based on the “Single Threaded Event Loop Model” architecture which can handle multiple client requests.
* **Asynchronous: -** when a client requests a server, a single thread handles the request; it checks if the request involves any database interaction if it does not; the request is processed and the server sends back the response to the client.
* **Event Driven:-** Node provides a module called “Event” which consists of an “Event Emitter” class that gives us the privilege to implement event-driven programming.
* **Very Fast −** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
* **No Buffering −** Node.js applications never buffer any data. These applications simply output the data in chunks.
* Question 4

What is the role of databases in websites? What is meant by SQL database and NoSQL database? Give some examples of these.

**ROLE OF DATABASE IN WEBSITES: -**

* The principal role of a database is to save and display updated information in a web application.
* For instance, registration websites, discussion forums, and retail commerce websites of web applications that depend upon a strong database element.
* If talking about the functions, web database applications are used to find, sort, filter and present data based upon web requests from users.
* Features of databases include that it allows and limit access to data based upon criteria such as user name, passcode, and region or account number.
* The database automatically updates web pages, removing the requirement to manually update the HTML code on individual pages

**Two Types of Databases: -**

**Relational Databases (SQL) :-**

* Organize data into one or more tables.
* Each table has columns and rows.
* A unique key identifies each row.

**Non-Relational (NoSQL/ not just SQL): -**

* Organize data is anything but a traditional table, any database that’s not relational comes under this category.
* Key-values stores
* Graphs
* Flexible tables

**SQL: -**

* SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.
* SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.
* The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP.

**Uses of SQL: -**

* Allows users to access data in the relational database management systems.
* Allows users to describe the data.
* Allows users to define the data in a database and manipulate that data.
* Allows users to create and drop databases and tables.
* Allows users to set permissions on tables, procedures and views.

**Example: -**

CREATE DATBASE (databasename); - this statement is used to create new SQL database.

DROP DATABASE (databasename); - this statement is used to drop an existing SQL database.

**NoSQL:** -

* NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale.
* The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data.
* In the year 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database.
* NoSQL databases never follow the relational model it is either schema-free or has relaxed schemas.

**Types of NoSQL Databases: -**

* Key-value Pair Based
* Column-oriented Graph
* Graphs based
* Document-oriented

**Advantage of NoSQL:** -

* Can be used as Primary or Analytic Data Source
* Big Data Capability
* No Single Point of Failure
* Easy Replication
* It provides fast performance and horizontal scalability.
* Can handle structured, semi-structured, and unstructured data with equal effect

Example: -

* MongoDB, CouchDB, Couchbase, Cassandra, HBASE, Redis, Riak, Neo4J are the popular NoSQL databases examples.
* Question 5

List different tags in HTML? Mention the uses of these tags.

* So, HTML tags are basically main key words that we add to our html document to structure the content of our webpage.

|  |  |
| --- | --- |
| [<!DOCTYPE>](https://www.w3schools.com/tags/tag_doctype.asp) | Defines the document type |
| [<html>](https://www.w3schools.com/tags/tag_html.asp) | Defines an HTML document |
| [<head>](https://www.w3schools.com/tags/tag_head.asp) | Contains metadata/information for the document |
| [<title>](https://www.w3schools.com/tags/tag_title.asp) | Defines a title for the document |
| [<body>](https://www.w3schools.com/tags/tag_body.asp) | Defines the document's body |
| [<h1> to <h6>](https://www.w3schools.com/tags/tag_hn.asp) | Defines HTML headings |
| [<p>](https://www.w3schools.com/tags/tag_p.asp) | Defines a paragraph |
| [<br>](https://www.w3schools.com/tags/tag_br.asp) | Inserts a single line break |
| [<hr>](https://www.w3schools.com/tags/tag_hr.asp) | Defines a thematic change in the content |

**Some HTML tags and their uses.**

|  |  |
| --- | --- |
| [<!--...-->](https://www.w3schools.com/tags/tag_comment.asp) | Defines a comment |
| [<blockquote>](https://www.w3schools.com/tags/tag_blockquote.asp) | Defines a section that is quoted from another source |
| [<form>](https://www.w3schools.com/tags/tag_form.asp) | Defines an HTML form for user input |
| [<input>](https://www.w3schools.com/tags/tag_input.asp) | Defines an input control |
| [<textarea>](https://www.w3schools.com/tags/tag_textarea.asp) | Defines a multiline input control (text area) |
| [<button>](https://www.w3schools.com/tags/tag_button.asp) | Defines a clickable button |
| [<select>](https://www.w3schools.com/tags/tag_select.asp) | Defines a drop-down list |
| [<optgroup>](https://www.w3schools.com/tags/tag_optgroup.asp) | Defines a group of related options in a drop-down list |
| [<option>](https://www.w3schools.com/tags/tag_option.asp) | Defines an option in a drop-down list |
| [<label>](https://www.w3schools.com/tags/tag_label.asp) | Defines a label for an <input> element |
| [<img>](https://www.w3schools.com/tags/tag_img.asp) | Defines an image |

* **Question 6**

List out ways of adding CSS to websites? Explain briefly

* CSS is added to our webpage to make them more appealing.
* CSS can be added to web in 3 ways.

## 1. External CSS

With an external style sheet, we can change the look of an entire website by linking just 1 file.

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

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

This is the way we add CSS with just a single line. In href we write name of CSS file with extension .css.

## 2. Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

<style>  
body {  
  background-color: linen;  
}  
  
h1 {  
  color: maroon;  
  margin-left: 40px;  
}

Here the style attributes are used within style tag.

**3. Inline CSS**

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

**Eg.**

<p style="color:red;">This is a paragraph.</p>

* **Question 7**

What is a functional component in React? How is it different from class-based components? What are the advantages and disadvantages of functional components? Explain the use of hooks in functional components.

**Functional component in react:-**

* Functional componentsin react are pure javascript functions.
* It takes an object as an argument called props( stands for properties) and returns JSX
* Before Hooks introduced, it’s also known as a Stateless Component.
* Now, we can’t call it a stateless component anymore since it can also have states and lifecycles.

|  |  |
| --- | --- |
| **Functional Components**   * A functional component is just a plain JavaScript function that accepts props as an argument and returns a React element. * There is no render method used in functional components * React lifecycle methods (for example, componentDidMount) cannot be used in functional components. | **Class Components**   * A class component requires you to extend from React. Component and create a render function which returns a React element * It must have the render() method returning HTML. * React lifecycle methods can be used inside class components (for example, componentDidMount) |

* **Advantages of functional components:-**
* **Easier to test:-** We don’t have to worry about hidden state and there aren’t as many side effects when it comes to functional components, so for every input, the functions will have exactly one output.
* **Easier to read/write:-** The syntax is less complex than that of class components, and it’s easier to read due to knowing how much you can’t do with functional components already.
* **Easier to debug.**
* **Disadvantages of fuctional component:-**
* **Relearning new syntax:-** The syntax could be unusual at first glance and difficult to pick up because of how long class components have been around.
* **Performance optimization:-** since functional components don’t have access to methods like shouldComponentUpdate and PureComponent, it could be a bit of an inconvenience to optimize them for performance.
* **Use of hooks in functional component:-**
* Using hooks, we can get the state of the component so that it can be easily tested and reused.
* Now we can facilitate the exchange of links between components or our entire application using hooks.
* Hooks allow you to encapsulate logic without affecting the hierarchy of components.
* **Hooks are just javascript functions, but they require only two rules: -**
* Hooks should be performed at the very top of the function hierarchy (this means that you should not call hooks in conditions and loops, otherwise the reaction cannot guarantee the execution order of hooks)
* Call hooks only in React functions or functional components or call hooks from custom hooks.
* **Question 8**

JWT AND COOKIES.

* JWT: JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.
* Cookies: An HTTP cookie is a small piece of data stored on the user's computer by the web browser while browsing a website. Cookies were designed to be a reliable mechanism for websites to remember stateful information or to record the user's browsing activity.
* **Question 9**

What is CORS?

# Cross-Origin Resource Sharing (CORS)

**Cross-Origin Resource Sharing** ([CORS](https://developer.mozilla.org/en-US/docs/Glossary/CORS)) is an [HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP)-header based mechanism that allows a server to indicate any other [origin](https://developer.mozilla.org/en-US/docs/Glossary/Origin)s (domain, scheme, or port) than its own from which a browser should permit loading of resources. CORS also relies on a mechanism by which browsers make a “preflight” request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

An example of a cross-origin request: the front-end JavaScript code served from https://domain-a.com uses [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) to make a request for https://domain-b.com/data.json.

The CORS mechanism supports secure cross-origin requests and data transfers between browsers and servers. Modern browsers use CORS in APIs such as XMLHttpRequest or [Fetch](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) to mitigate the risks of cross-origin HTTP requests.

**Following resources use CORL.**

* Invocations of the [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) or [Fetch APIs](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API), as discussed above.
* Web Fonts (for cross-domain font usage in font-face within CSS), [so that servers can deploy TrueType fonts that can only be cross-site loaded and used by web sites that are permitted to do so.](https://www.w3.org/TR/css-fonts-3/#font-fetching-requirements)
* [WebGL textures](https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Using_textures_in_WebGL).
* Images/video frames drawn to a canvas using [drawImage()](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/drawImage).
* [CSS Shapes from images.](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Shapes/Shapes_From_Images)