5.1 INTRODUCTION

5.1.1 Introduction to Flask

Flask is a popular Python framework for developing web applications. It is a framework with a built-in development server and a debugger.

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies, and several common framework-related tools.

Flask is Python's micro-framework for web app development. It was developed by **Armin Ronacher**, who led an international team of Python enthusiasts called Pocco. Flask consists of Werkzeug WSGI toolkit and Jinja2 template engine. Both were also developed by Pocco. It was initially released in April 2010. Currently, the latest version as of writing this article is 2.0.2.

What is WSGI?

- Web Server Gateway Interface (WSGI) is the standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.
- Werkzeug is one of the most advanced WSGI modules that contain various tools and utilities that facilitate web application development. Flask implements Werkzeug.

What is Jinja 2?

- Jinja 2 is a template rendering engine. It renders the web pages for the server with any specified custom content given to it by the webserver. Flask renders its HTML based templates using Jinja 2.

5.1.2 Features of Flask

1. It is a lightweight framework.
2. Flask gives independent or full control to the developer for creating applications.
3. Flask provides a development server and a debugger.
4. It uses Jinja2 templates.
5. It is compliant with WSGI 1.0.
6. It provides integrated support for unit testing.
7. Many extensions are available for Flask, which can be used to enhance its functionalities.
8. Secure cookies
9. Support URL routing
10. Unicode support
11. Request dispatching

5.1.3 Advantages of Flask

Flask is a lightweight Web Server Gateway Interface (WSGI) web application framework. It has minimal or no external libraries. It is used to create a simple website and then scale it up to complex applications. Below are some of the advantages of Flask.

1. Adaptable to the latest technologies
2. Independent framework enables experimentation with architecture, libraries.
3. Suitable for small case projects.
4. Requires small codebase size for simple functions
5. Ensures scalability for simplistic applications
6. Easy to build a quick prototype
7. Routing URL functions through Werkzeug makes the process easier.
8. Hassle-free application development and maintenance.
9. Database integration is easy
10. Extensible and easy core system.
11. The power of the framework lies in its minimalistic features.
12. Flexible and allow full control access.

5.1.4 What is Micro-framework?

- In contrast to a traditional web framework, a micro-framework is a minimalistic framework where developers are provided with a lot of freedom to build the web application layer.
- As compared to an enterprise framework, a developer would not need to set up many things in a micro-framework to get the web app hosted and running.

6.1 INTRODUCTION TO AJAX

- AJAX stands for Asynchronous JavaScript and XML.
- AJAX or Asynchronous JavaScript and XML is a set of web development techniques which allows web applications to work asynchronously.
- It is used to communicate with the server without refreshing the web page and thus increasing the user experience and better performance.
- Ajax is made up of the following technologies:
  (1) XHTML and CSS for presenting information.
  (2) Document Object Model (DOM) for dynamically interacting with and displaying the presented information.
  (3) XMLHttpRequest object to manipulate data asynchronously with the web server.
  (4) XML, HTML, and XSLT for data interchange and manipulation.
  (5) JavaScript for binding data requests and information display.
- AJAX allows the web pages to be modified asynchronously by exchanging small bunch of information with the server.
- This means that it is easily possible to make changes in part of a web page, without reloading the entire page. The conventional web pages, (which are not using AJAX) reload the whole page if there is need to make changes in the content.

Examples of Applications using AJAX

- Google Maps, Gmail, Youtube, and Facebook tabs.
- You may have seen the live score of cricket match on different websites like www.yahoo.com.
- There are updations of each and every ball. In such case only small part of webpage is needed to be changed.
- Only that much information is sent by the server and updated in the webpage rather than updating the entire page. This is implemented by the AJAX.
- Just consider if it is done with conventional web pages without AJAX, then it will take lot much of duration every time to reload the whole page.

Features of Ajax

- User Friendly
- It make web page faster
- Independent of server technology
- Increase the Performance of web page
- Support for Live data binding
- Support for the Data View control
- Support for Client-side template rendering
- Rich and, responsive user interfaces
- Reduced consumption of server resources
- No need to pushing on a submit button and reloading the complete website.
- No need to reload the whole page only some part of page is reloaded which is required to reload.
- Apart from obtaining the XMLHTTP object, all processing is same for all browser types, because JavaScript is used.
- Using AJAX, develop faster and more interactive web applications.
- Not require to completely reload page due to this server uses less bandwidth.

For understanding the working of AJAX, let's first understand the classical model for web development i.e. synchronous model.

Synchronous (Classic Web-Application Model)

The synchronous model can be classified into three stages :

(1) On the client side, if the user does any activity on the browser, the browser sends a HTTP request to the server and waits for a response.

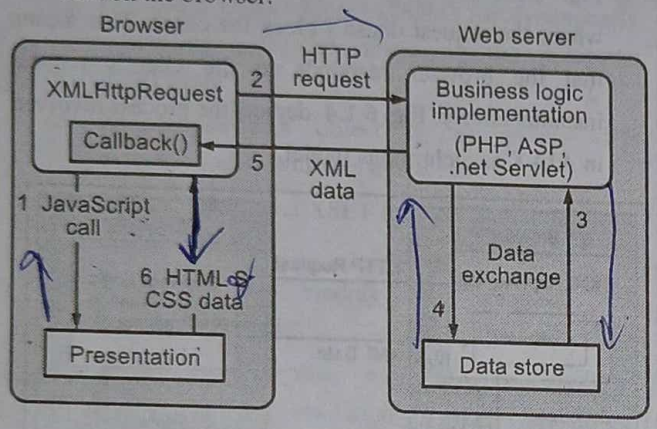
(2) The server does some processing on the request of the user and sends a response to the user.

(3) When the response is received at the user end, the browser reloads the webpage to show the information.

Fig. 6.1.1 illustrate the classical model or a synchronous model.

6.2 WORKING OF AJAX

- Ajax communicates with the server by using XMLHttpRequest Object. User send request from User Interface and JavaScript call goes to the XMLHttpRequest Object after that XMLHttp request is sent to the XMLHttpRequest Object.
- At that time server interacts with the database using php, servelet, ASP.net etc.
- The data is retrieved then the server sends data in the form of XML or JSON data to the XMLHttpRequest Callback function. Then HTML and CSS displayed the data on the browser.



(1F1)Fig. 6.2.1 : Working of AJAX

The functioning is as follows :

(1) User sends a request from the UI and a javascript call goes to XMLHttpRequest object.

(2) HTTP Request is sent to the server by XMLHttpRequest object.

(3) Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.

(4) Data is retrieved.

(5) Server sends XML data or JSON data to the XMLHttpRequest callback function.

(6) HTML and CSS data is displayed on the browser.

- In Ajax model, there is an Ajax engine involved in between the user and the server, which eliminates the to and fro from the user to the server and back.
- This Ajax engine is written in JavaScript and is in a hidden frame. It handles the user front by communicating to the user as well as handles the server front by itself.
- In the working of AJAX, the two aspects are very important: AJAX Request and AJAX Response.

AJAX Request

- AJAX sends a request to a server by using open() and send() methods of the XMLHttpRequest object.
- Following code snippet illustrates how an AJAX request can be sent to the server.

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

- In the above code snippet, the **GET** is the request type, **ajax_info** is the location of the server type and **true** shows that the request is asynchronous.
- Following are the methods used to send a request in AJAX.

(1) **Open(method, url, async):** Specifies the type of request
  - method : the type of request: GET or POST
  - url : the server (file) location
  - async : true (asynchronous) or false (synchronous)

(2) **send() :** Sends the request to the server (used for GET)

(3) **send(string):** Sends the request to the server (used for POST)

AJAX Response

- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes as shown in the following code snippet.

```
var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
```

- In the above code snippet, the response is decided by two properties of XMLHttpRequest: readyState and status. When the status is 200 and readyState is 4, the response is ready.

Difference between CMS and Framework

CMS	Framework
A CMS or content management system is a computer application that is used for creating and modifying digital content.	A framework is a software which contains a generic functionality that can be modified by additional user-written code depending on the application.
CMS helps to manage digital content.	A framework helps to organize the code to make the application development process simpler and flexible.
CMS is an open-source, so it is not as secure as frameworks.	The framework includes own custom code, so they are more secure than CMS.
It is not possible to mould CMS according to user requirement.	The framework can easily be moulded according to user requirement.
CMS is updated regularly, to secure them properly	The framework is rarely updated but it needs to be maintained.
Drupal, WordPress, and Joomla are some examples of CMS.	Django, Flask, CakePHP and CodeIgniter are some examples of frameworks.

6.12 DJANGO FRAMEWORK

What is Django ?

- Django is a free, open source web framework written in the python programming language, which offers standard methods for fast and effective website development.
- A web framework is a software that supports the development of dynamic web sites, applications, and services.
- It provides a set of tools and functionalities that solves many common problems associated with web development, such as security features, database access, sessions, template processing, URL routing, internationalization, localization, and much more.
- It enables you to make the development process smooth and time-saving for rapid development.
- The primary goal of this high-level web framework is to create complex database-driven websites. It helps you to build and maintain quality web applications.

Features of Django Framework

(1) Python Web-framework
(2) SEO optimised
(3) High scalability
(4) Versatile in nature
(5) Offers high security
(6) Thoroughly tested
(7) Provides rapid development

Advantages of Django

(1) **Written in Python :** Django is one of the web frameworks which are written in Python programming language. Hence, it becomes easier for programmers to build web applications with clean, readable, and maintainable code by taking advantage of syntax rules of Python.

(2) **Accelerates custom web application development :** Django is one of the most mature web frameworks for Python. It's design rules focus extensively on reducing web application development time. The features provided by Django enable developers to build custom web applications rapidly according to varying business requirements.

(3) **Designed as a batteries-included web framework :** Django is one of the web frameworks that adopt the batteries-included approach. While developing a custom web application, Django provides the resources required by developers out of the box. It provides code for common operations like database manipulation, HTML templating, URL routing, session management, and security.

(4) **Supports MVC programming paradigm :** Django, like other modern web frameworks, supports model-view-controller (MVC) design rule. The MVC programming paradigm allows programmers to keep a web application's user interface (UI) and business logic layers separated. The approach further helps

programmers to simplify and speed up development of large web applications by separating their user interface and business logic layers.

(5) **Compatible with major operating systems and databases :** Nowadays, users access web applications on various devices and platforms. Django enhances the accessibility of web applications by supporting major operating systems like Windows, Linux and MacOS. At the same time, the ORM system provided by Django makes it easier for programmers to work with several widely used databases. They can even use the ORM system to perform common database operations and migrate from one database to another without writing additional code.

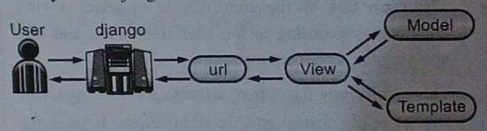
(6) **Provides robust security features :** The built-in security features provided by Django help developers to protect the web applications from a variety of targeted security attacks – cross-site scripting, SQL injection and cross-site request forgery.

(7) **Easy to extend and scale :** Django has been evolving consistently to enable programmers to build better and modern web applications. At the same time, the Django developers can easily customize, scale, and extend the web framework by making changes to its decoupled components.

(8) **Supported by a large and active community :** As an open source web framework for Python, Django helps developers to reduce web application development cost significantly. But it is supported by a large and active community of developers. The members of the Django community update new plug-ins and code snippets regularly to simplify web application development.

Django Architecture: MVC Pattern

- There are three primary components or code partitions for any website on the internet: Input Logic, Business Logic, and UI Logic.
- These partitions of code have specific tasks to achieve, the input logic is the dataset and how the data gets to organize in the database.
- It just accepts that input and transmits it to the database in the format specified.
- The main controller is the business logic, which handles the server's output in HTML or the required format. The HTML, CSS, and JavaScript pages comprise the UI Logic, as the name implies.
- When the traditional approach was used for programming all this code was written in a single file, i.e., every piece of code increases the webpage size, which is downloaded and rendered by the browser.
- This was not a big problem back in the time, the webpages were largely static and websites didn't contain much multimedia and large coding.
- Also, this architecture poses difficulty for developers while testing and maintaining the project as everything is inside one file.
- MVC pattern is a Product Development Architecture. It solves the traditional approach's drawback of code in one file, i.e., that MVC architecture has different files for different aspects of our web application/ website. The MVC pattern has three components, namely Model, View, and Controller.
- The Django architecture Fig. 6.12.1 shows the working cycle of Django MVC architecture.



(1F3)**Fig. 6.12.1 : MVC Architecture**

1. Model

- The Model is the component which contains business logic in Django architecture.
- The Model is the part of the web-app which acts as a mediator between the website interface and the database. In technical terms, it is the object which implements the logic for the application's data domain. There are times when the application may only take data in a particular dataset, and directly send it to the view (UI component) without needing any database then the dataset is considered as a model.

For example :

- When you sign up on any website you are actually sending information to the controller which then transfers it to the models which in turn applies business logic on it and stores in the database.

2. View

- This component contains the UI logic in the Django architecture.
- View is actually the user interface of the web-application and contains the parts like HTML, CSS and other frontend technologies.
- Generally, this UI creates from the Models component, i.e., the content comes from the Models component.

For example :

- When you click on any link or interact with the website components, the new webpages that website generates is actually the specific views that stores and generates when we are interacting with the specific components.

3. Controller

- The controller as the name suggests is the main control component. What that means is, the controller handles the user interaction and selects a view according to the model.
- The main task of the controller is to select a view component according to the user interaction and also applying the model component.
- This architecture has lots of advantages and that's why Django is also based on this architecture. It takes the same model to an advanced level.

For example :

- When we combine the two previous examples, then we can very clearly see that the component which is actually selecting different views and transferring the data to the model's component is the controller.

6.13 DRUPAL

Django	Flask
Django could be a Python-based free, open source system which takes after the MVT(model view Template) approach of structural design	Flask could be a Python-based smaller scale system without any set of specific instruments or outside libraries. It too doesn't have a database layer or arrangements for shape approval and makes utilize of expansions.
Urls.py is utilized to set the association properties and demands are handled by the primary coordinating see of regex list	URI is most regularly than not set by the see decorator and centralized setup is additionally conceivable. Sometimes the recent designs are coordinated with the URIs, the last mentioned is sorted in a default arrange
Doesn't exclude setting flexibility	It is accepted that all the conceivable stages to organize a Flask code rises to the applications number show in flask as of now
Extend Layout is Conventional extend structure	Extend Layout is Arbitrary structure
Django gives an all-inclusive encounter: you get an admin board, database interfacing, an ORM, and registry structure for your apps and ventures out of the box.	Flask gives straightforwardness, adaptability and fine-grained control. It is unopinionated
It is suitable for multi page applications.	It is suitable for single page applications only.
Its framework structure is more conventional.	Random web framework structure.
It doesn't support any virtual debugging.	It has a built-in debugger that provides virtual debugging.
Its working style is Monolithic.	Its working style is diversified style.

6.13 DRUPAL

What is Drupal?

- Drupal is a free and open source Content Management System (CMS) that allows organizing, managing and publishing your content.
- It is built on PHP based environments. Drupal CMS is more flexible if you develop a website with a content management system than any other CMS.
- Drupal is mighty and can be used for building large,

Advantages of Drupal

(1) Drupal is a flexible CMS that allows handling content types, including video, text, blog, menu handling, real-time statistics, etc.

(2) Drupal provides several templates for developing web applications. There is no need to start from scratch if you are building simple or complicated web applications.

(3) Drupal is easy to manage or create a blog or website. It helps to organize, structure, find and reuse content.

(4) Drupal provides some interesting themes and templates which give your website an attractive look.

Disadvantages of Drupal

A certain stiffness of the system is often an advantage but, admittedly, can also be a disadvantage. Listed are a few of the downsides of Drupal that can make the system inconvenient.

- The difficulty with familiarization. For those users who have not previously built a site, the creation and promotion of a site using Drupal can be difficult. While it will not take long to master the basic functions of the system, to fully understand it may take you several months.
- Advanced interface. Professional construction systems make Drupal difficult to master. Due to its complex interface, it is not very intuitive.
- High system requirements. Before turning to the use of this CMS, you must be prepared to forget about cheap fares and pay an expensive hosting fee.
- The lack of high-quality designs in free access. Most users of the system are engaged in self-development designs for their projects. This leads to a shortage of quality topics for the site design.
- Difficulty in the installation of updates and program modules. This system is complicated for those web developers who previously had to deal with a simple CMS, as the installation of updates and additional modules in the Drupal software is made through FTP.

AS List applications of Mongoose.

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, which provides a straightforward and easy-to-use API for interacting with MongoDB databases. Here are some of the popular applications of Mongoose:

1. Building web applications: Mongoose is commonly used to build web applications using Node.js and MongoDB. It provides a powerful and flexible API for querying and manipulating data in MongoDB databases, making it an excellent choice for building web applications that require a database backend.
2. Building RESTful APIs: Mongoose can also be used to build RESTful APIs that interface with MongoDB databases. Its powerful schema system allows developers to define the structure of their data and validate incoming requests, making it easy to build robust APIs that can handle a wide range of requests.
3. Real-time applications: Mongoose is a popular choice for building real-time applications that require fast and efficient data processing. It can be used to store and retrieve data in real-time, making it an excellent choice for applications that require instant data updates.
4. Data analysis and reporting: Mongoose can also be used to analyze and report on data stored in MongoDB databases. Its flexible query system allows developers to perform complex data analysis and generate reports based on the data stored in MongoDB.
5. Data migration: Mongoose can be used to migrate data from one MongoDB database to another. Its powerful API makes it easy to perform complex data migrations, even across different MongoDB versions.
6. Automated testing: Mongoose can be used for automated testing of Node.js applications that use MongoDB databases. Its API allows developers to easily create and destroy test databases, populate them with data, and run automated tests against the data stored in MongoDB.

Regenerate response

AS Define CRUD operations

CRUD stands for Create, Read, Update, and Delete, which are the four basic operations that can be performed on a persistent data store such as a database.

1. Create: The Create operation involves adding a new record or data item to the data store. In a database, this is done by inserting a new row into a table or creating a new document in a NoSQL database.
2. Read: The Read operation involves retrieving an existing record or data item from the data store. In a database, this is done by querying a table or a collection and retrieving the desired data.
3. Update: The Update operation involves modifying an existing record or data item in the data store. In a database, this is done by updating the values of one or more columns in a row or modifying the fields of a document.
4. Delete: The Delete operation involves removing an existing record or data item from the data store. In a database, this is done by deleting a row from a table or deleting a document from a collection.

These four basic operations are fundamental to working with any persistent data store, and they form the basis of most data-driven applications.

Sr. No.	Key	RDBMS	MongoDB
1	Concept	RDBMS is a relational database management system and works on relational database.	MongoDB is a non-relational, document oriented database management system and works on document based database.
2	Hiearchical	Difficult to store hiearchical data.	Have inbuilt support to store hiearchical data.
3	Scalablity	RDBMS is vertically scalable. Performance increases with increase of RAM.	MongoDB is horizontally scalable as well. Its performance increases with addition of processor.
4	Schema	Schema need to be defined in RDBMS before using a database.	Schema can be dynamically created and accessed in MongoDB.
5	SQL Injection	Vulnerable to SQL Injection attack.	SQL injection is not possible.
6	Principle	Follows ACID principle, Atomicity, Consistency, Isolation, and Durability.	Follows CAP theorem, Consistency, Availability, and Partition tolerance.
7	Basis	Database uses Row.	Database uses Document.
8	Basis	Database uses Column.	Database uses Field.
9	Performance	RDBMS is slower in processing large hierachical data.	MongoDB is blazingly fast in processing large hierachical data.
10	Joins	RDBMS supports complex joins.	MongoDB has no support for complex joins.
11	JavaScript Client	RDBMS do not provide JavaScript based client to query database.	MongoDB provides Javascript based client to query database.
12	Query Language	RDBMS uses SQL to query database.	MongoDB uses BSON to query database.

4.8 REST API

- An API, or application programming interface, is a set of rules that define how applications or devices can connect to and communicate with each other. A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.
- REST stands for representational state transfer and was developed by Roy Fielding in 2000.
- REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways.
- When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular file format to use because, despite its name, it's language-agnostic, as well as readable by both humans and machines.

4.8.1 REST Design Principles

The API needs to meet the following architectural requirements to be considered a REST API.

1. **Uniform Interface :** All API requests for the same resource should look the same, no matter where the request comes from. The REST API should ensure that the same piece of data, such as the name or email address of a user, belongs to only one uniform resource identifier (URI). Resources shouldn't be too large but should contain every piece of information that the client might need.
2. **Client-Server :** In REST API design, client and server applications must be completely independent of each other. The only information the client application should know is the URI of the requested resource; it can't interact with the server application in any other ways. Similarly, a server application shouldn't modify the client application other than passing it to the requested data via HTTP.
3. **Stateless :** REST APIs are stateless, meaning that each request needs to include all the information necessary for processing it. In other words, REST APIs do not require any server-side sessions. Server applications aren't allowed to store any data related to a client request.
4. **Cacheable :** When possible, resources should be cacheable on the client or server side. Server responses also need to contain information about whether caching is allowed for the delivered resource. The goal is to improve performance on the client side, while increasing scalability on the server side.
5. **Layered System Architecture :** In REST APIs, the calls and responses go through different layers. As a rule of thumb, don't assume that the client and server applications connect directly to each other. There may be a number of different intermediaries in the communication loop. REST APIs need to be designed so that neither the client nor the server can tell whether it communicates with the end application or an intermediary.
6. **Code on Demand :** REST APIs usually send static resources, but in certain cases, responses can also contain executable code (such as Java applets). In these cases, the code should only run on-demand.

4.8.2 Rules of REST API

There are certain rules which should be kept in mind while creating REST API endpoints.

1. REST is based on the resource or noun instead of action or verb based. It means that a URI of a REST API should always end with a noun.

   Example: /api/users
2. HTTP verbs are used to identify the action. Some of the HTTP verbs are – GET, PUT, POST, DELETE, UPDATE, PATCH.

   HTTP verbs: Some of the common HTTP methods/verbs are described below:

   (i) **GET :** Retrieves one or more resources identified by the request URI and it can cache the information receive.

   (ii) **POST :** Create a resource from the submission of a request and response is not cacheable in this case. This method is unsafe if no security is applied to the endpoint as it would allow anyone to create a random resource by submission.

   (iii) **PUT :** Update an existing resource on the server specified by the request URI.

   (iv) **DELETE :** Delete an existing resource on the server specified by the request URI. It always return an appropriate HTTP status for every request.
3. A web application should be organized into resources like users and then uses HTTP verbs like - GET, PUT, POST, and DELETE to modify those resources. And as a developer it should be clear that what needs to be

Module 4

AS Write any 4 Built-in Roles in MongoDB

MongoDB provides several built-in roles that allow users to perform different actions on databases and collections. Here are four of the most commonly used built-in roles in MongoDB:

1. Read: The Read role grants users the ability to read data from a database or collection. Users with this role can execute the find() and findOne() methods to retrieve data, but they cannot make any modifications to the data.
2. ReadWrite: The ReadWrite role grants users the ability to both read and write data to a database or collection. Users with this role can execute all CRUD operations, including insert(), update(), and remove() methods.
3. dbAdmin: The dbAdmin role grants users the ability to perform administrative tasks at the database level, such as creating and deleting databases, creating and managing collections, and creating and modifying indexes.
4. userAdmin: The userAdmin role grants users the ability to create, modify, and delete user accounts in a MongoDB database. Users with this role can create new users, modify user permissions, and delete user accounts.

There are also other built-in roles in MongoDB, such as the clusterAdmin role for managing a MongoDB cluster, the backup role for performing backups, and the restore role for performing restores. Additionally, MongoDB allows users to define custom roles that provide specific permissions tailored to their application's needs.

What Are the Drawbacks of MongoDB?

MongoDB has its share of disadvantages as well, including:

- It doesn't support transactions. Although fewer applications require transactions today, some still do need it to update multiple collections or documents
- It lacks triggers, something that makes life easier in relational database management systems (RDBMS)
- MongoDB requires more storage than other well-known databases
- It doesn't automatically clean up its disk space, so it must be done manually or with a restart
- It isn't easy to join two documents in MongoDB. If you need to pull data from multiple collections using a single query, you're out of luck
- Finally, you better ensure that your indexes are correctly implemented or in the correct order. Otherwise, MongoDB's speed drops significantly

If you plan on pursuing a position where you need knowledge of MongoDB, then you need an understanding of its pros and cons. Additionally, brush up on your familiarity with these MongoDB interview questions.

1.3.1 Characteristics of Web 3.0

Below are 5 main features that can help us define Web 3.0 :

1. **Semantic Web :** The next evolution of the Web involves the Semantic Web. The semantic web improves web technologies in order to generate, share and connect content through search and analysis based on the ability to understand the meaning of words, rather than on keywords or numbers.

2. **Artificial Intelligence :** Combining this capability with natural language processing, in Web 3.0, computers can understand information like humans in order to provide faster and more relevant results. They become more intelligent to satisfy the needs of users.

3. **3D Graphics :** The three dimensional design is being used extensively in websites and services in Web 3.0. Museum guides, computer games, ecommerce, geospatial contexts, etc. are all examples that use 3D graphics.

4. **Connectivity :** With Web 3.0, information is more connected thanks to semantic metadata. As a result, the user experience evolves to another level of connectivity that leverages all the available information.

5. **Ubiquity :** Content is accessible by multiple applications, every device is connected to the web, the services can be used everywhere.

AS What are the application of AJAX in Blogs

AJAX (Asynchronous JavaScript and XML) is a web development technique that allows web pages to update dynamically without requiring a full page refresh. Here are some of the ways that AJAX can be used in blogs:

1. Live search: AJAX can be used to implement a live search feature in a blog, allowing users to search for content without having to reload the page. As the user types in their search query, the results are displayed in real-time, making it easier and faster to find the desired content.
2. Commenting system: AJAX can be used to create a commenting system that updates in real-time without requiring the user to reload the page. As new comments are added, they can be displayed immediately, providing a more interactive and engaging user experience.
3. Pagination: AJAX can be used to implement pagination in a blog, allowing users to navigate through multiple pages of content without having to reload the entire page. As the user clicks on the page links, the content is loaded dynamically, improving the speed and efficiency of the pagination system.
4. Social media integration: AJAX can be used to integrate social media features into a blog, such as allowing users to share content on social media platforms without leaving the page. As the user clicks on the share button, the content is shared in real-time, providing a seamless user experience.

Overall, AJAX can be used to create a more dynamic, responsive, and engaging user experience in a blog, improving the overall user satisfaction and interaction with the content.

AS Write any 4 User defined roles in MongoDB

In MongoDB, users can define custom roles that provide specific permissions tailored to their application's needs. Here are four examples of user-defined roles in MongoDB:

1. Editor: An editor role might be defined to allow users to edit documents in a specific collection, but not to create or delete them. This role could be useful in a content management system where certain users are responsible for updating existing content.
2. Analyst: An analyst role might be defined to allow users to run aggregation queries on a specific collection, but not to modify or delete any data. This role could be useful in a business intelligence system where analysts need to analyze data without the risk of changing it.
3. Manager: A manager role might be defined to allow users to perform administrative tasks at the database level, such as creating and deleting collections, managing indexes, and creating backups. This role could be useful for a database administrator or a team leader.
4. Developer: A developer role might be defined to allow users to create, modify, and delete documents in a specific collection, as well as create indexes and run queries. This role could be useful for developers who need to interact with the database as part of their development process.

User-defined roles in MongoDB provide a flexible way to grant specific permissions to users based on their role in the organization or their specific tasks. By defining custom roles, users can ensure that their database is secure and that users have access to the data they need to perform their tasks.

Regenerate response