**ADVANCED WEB APPLICATION DEVELOPMENT**

**ASSIGNMENT WEEK 2**

1.Explain in detail about CRUD operation in MongoDB.

Answer 1: CRUD operations are the basic methods of interacting with data stored in MongoDB, a popular NoSQL database. [CRUD stands for Create, Read, Update, and Delete, and these operations allow you to insert, query, modify, and remove documents in a MongoDB collection1](https://www.mongodb.com/basics/crud)[2](https://www.mongodb.com/docs/manual/crud/).

To perform CRUD operations in MongoDB, you need to use the MongoDB Query Language (MQL), which is a JSON-like syntax for specifying filters, projections, updates, and aggregations. [You can use MQL with various MongoDB tools and drivers, such as the mongo shell, Compass, or Node.js driver1](https://www.mongodb.com/basics/crud)[3](https://www.mongodb.com/docs/drivers/node/current/fundamentals/crud/).

Here is a brief overview of how to perform CRUD operations in MongoDB using MQL:

* **Create**: To insert new documents into a collection, you can use the db.collection.insertOne() or db.collection.insertMany() methods. These methods take a document or an array of documents as arguments and return an acknowledgment object with the inserted document’s \_id field. For example:

// Insert one document db.users.insertOne({name: “Alice”, age: 25})

// Insert many documents db.users.insertMany([ {name: “Bob”, age: 30}, {name: “Charlie”, age: 35} ])

* **Read**: To query documents from a collection, you can use the db.collection.find() method. This method takes a filter document and an optional projection document as arguments and returns a cursor to the matching documents. You can use various operators and modifiers to specify complex queries and sort, limit, or skip the results. For example:

// Find all documents db.users.find()

// Find documents with age greater than 30 db.users.find({age: {$gt: 30}})

// Find documents with name starting with “A” and only return the name field db.users.find({name: /^A/}, {name: 1})

* **Update**: To modify existing documents in a collection, you can use the db.collection.updateOne(), db.collection.updateMany(), or db.collection.replaceOne() methods. These methods take a filter document, an update document or a replacement document, and an optional options document as arguments and return an acknowledgment object with the number of matched and modified documents. You can use various update operators to specify how to change the fields of the matching documents. For example:

// Update one document by incrementing its age by 1 db.users.updateOne({name: “Alice”}, {$inc: {age: 1}})

// Update many documents by setting their status to “active” db.users.updateMany({}, {$set: {status: “active”}})

// Replace one document with a new document db.users.replaceOne({name: “Bob”}, {name: “Robert”, age: 31})

* **Delete**: To remove documents from a collection, you can use the db.collection.deleteOne() or db.collection.deleteMany() methods. These methods take a filter document and an optional options document as arguments and return an acknowledgment object with the number of deleted documents. For example:

// Delete one document with name “Charlie” db.users.deleteOne({name: “Charlie”})

// Delete many documents with status “inactive” db.users.deleteMany({status: “inactive”})

2.What is a namespace in MongoDB?

Answer 2: A namespace in MongoDB is a canonical name for a collection or an index. It is a combination of the database name and the name of the collection or index, separated by a dot. For example, the namespace for a collection named users in a database named test is test.users. All documents belong to a namespace[1](https://stackoverflow.com/questions/61099290/what-is-the-use-of-namespace-in-mongodb)[2](https://www.mongodb.com/docs/manual/reference/glossary/).

A namespace is used to identify and access collections and indexes in MongoDB. [It is also used to store metadata about collections and indexes in the system.namespaces collection, which is a special collection that exists in every database](https://stackoverflow.com/questions/61099290/what-is-the-use-of-namespace-in-mongodb)

3.How do I handle 404 responses?

Answer3: A 404 response is a HTTP status code that indicates that the requested resource was not found on the server. To handle 404 responses, you need to implement a way to catch and handle the requests that do not match any existing routes or resources on your server. The way to handle 404 responses may vary depending on the framework or tool you are using. Here are some examples of how to handle 404 responses in different frameworks:

* In Express, a Node.js web framework, you can add a middleware function at the bottom of the stack that sends a 404 response to any unmatched requests. For example:

// Add this middleware function after all other routes and middleware functions app.use((req, res, next) => { res.status(404).send(“Sorry can’t find that!”) })

See https://expressjs.com/en/starter/faq.html for more details.

* In ASP.NET MVC, a .NET web framework, you can use the Application\_Error event handler in the Global.asax file to catch and handle any unhandled exceptions, including HttpException with a 404 status code. For example:

// Add this event handler in the Global.asax file protected void Application\_Error(object sender, EventArgs e) { Exception exception = Server.GetLastError(); Response.Clear();

HttpException httpException = exception as HttpException;

if (httpException != null) { switch (httpException.GetHttpCode()) { case 404: // Page not found Response.Redirect(“~/Error/HttpError404”); break; // Handle other status codes here } }

Server.ClearError(); }