**Module 65-Mongodb, database integration, CRUD**

আজকের একদম প্রথেমই আছে mongodb জিনিসটা কি। সেটা দিয়ে কি করা লাগবে। এবং sql আর nosql এর মধ্যে ডিফারেন্স কি এই জিনিসটা সম্পর্কে হালকা ধারণা দেয়া হয়েছে। তবে একটু সময় বের করে এই ডিফারেন্স সম্পর্কে আরো ধারণা নিয়ে নিও কাজে লাগবে। তবে আজকের মডিউল এর আরেকটা উদ্দেশ্য হচ্ছে CRUD অপারেশন। অর্থাৎ REST API দিয়ে যে যে কাজ করা যায় তার সবগুলা চেখে দেখা। এমনকি কিছু কিছু সময় ইন্টারভিউ এর জন্য যে সব কাজ দেয় তার ৩০-৪০% সময় কাজ দেয় কোন একটা CRUD সিস্টেম ডেভেলপ করে দিতে। তাই আজকের ভিডিওগুলো একটু ভালো করে যে দেখার চেষ্টা করবে।

সেকেন্ড ভিডিওতে আছে mongodb এর ক্লাউড ডাটাবেজ। অর্থাৎ ডাটাবেজ তোমার কম্পিউটারে থাকবে না। ডেটাবেজ থাকবে অন্য একটা সার্ভারে। সেটা হতে পারে aws, গুগল ক্লাউড বা অন্য কোন ক্লাউড এ। সেটার মধ্যে একাউন্ট কিভাবে খুলবে। কিভাবে একসেস নিশ্চিত করবে।

যখন আমরা আমাদের ক্লাউড ডাটাবেজ এর সাথে কানেকশন রেডি করে ফেলবো তারপর আমাদের কাজ হচ্ছে ডাটাবেজ এ ডাটা পাঠানো। বা নতুন ডাটা তৈরি করা। এই তৈরি করার অপারেশন কে বলে create করা। এবং সেটা করা হয় rest এর post দিয়ে। তারপর আমরা সেই ডাটাকে ডাটাবেজ এ সেইভ করবো। সেইভ করার পর সেই ডাটা আমরা লোড করে আমাদের ওয়েবসাইট এ দেখাবো। তাইলে আমাদের ডাটা read করার কাজও হয়ে যাবে। অর্থাৎ CRUD এর C এবং R এর কাজ হয়ে যাবে। C দিয়ে বুজাচ্ছে Create করার কাজ। আর R দিয়ে বুজাচ্ছে ডাটা Read করার কাজ অর্থাৎ ডাটা get করার কাজ।

এরপর আসবে আমরা যদি কোন একটা ডাটা ডিলিট করতে চাই সেই অপারেশন। এবং লাস্টের দিকে থাকবে ডাটা এর কোন একটা অংশ যদি আমরা আমরা আপডেট করতে চাই সেই সিস্টেম। অর্থাৎ আজকের মডিউলে যেকোন ডাটাকে create, read, update, এবং delete করার সিস্টেম দেখবো।

এই কাজ করতে গেলে, আমরা যখন সার্ভার থেকে ঠিকমতো ডাটা সেন্ড করবো না। তখন আমাদের nodemon এর রেস্পন্স বারবার ক্লোজ করে আবার ব্যাকএন্ড এর সার্ভার চালু করতে হবে। এবং লাস্টের দিকে এসে এই প্রসেসটাকে আমরা স্মুথ করে ফেলবো।

এরর খাচ্ছি তো আর খাচ্ছি:

কারো কারো জন্যে গতকালের আর আজকের মডিউলটা প্যারাদায়ক হয়ে যাবে। যেমন মঙ্গো সার্ভারে কানেকশন হচ্ছে না। কানেক্ট হইছে কিন্তু ডাটা সেইভ হচ্ছে না। বা ডাটা ডিলিট হচ্ছে না। বা হাবিজাবি এরর দিচ্ছে। এই জিনিসগুলো ধৈর্য্য ধরে ধরে কেইস বাই কেইস হ্যান্ডেল করতে হবে। তবে কয়েকটা কমন সমাধান আমি দিয়ে দিচ্ছি।

১. দেখো যে কানেকশন স্ট্রিং ঠিক আছে কিনা। যদি কানেকশন স্ট্রিং এ <password> বা <dbname> এই টাইপের কিছু থাকে তাহলে সেটা ঠিক করতে হবে।

২. যদি ইউজার বা পাসওয়ার্ড নিয়ে নিশ্চিত না হও তাহলে atlas এ গিয়ে ক্লাস্টার এর নিচে database একসেস এ গিয়ে নতুন ইউজার খুলে তার ইউজার নেম আর পাসওয়ার্ড দিয়ে ট্রাই করো

৩. network access এ গিয়ে connect from anywhere দিয়ে দাও। IP এড্রেস ০.০.০.০.

৪. কোড এর মধ্যে সব ঠিক করে করছো কিনা ট্রাই করো। দরকার হলে কোড রিমুভ করে আবার নতুন করে শুরু করো।

৫. চেক করো "client.close()" লেখা কমেন্ট করা আছে কিনা।

৬. express, mongodb, cors ঠিকমতো ইন্স্টল্ করছো কিনা

৭. middleware ঠিক মতো বসাইছো কিনা

৮. nodemon এর সার্ভার Ctrl+C মেরে স্টপ করে আবার নতুন করে স্টার্ট করে দিয়ে দেখো। কাজ হয় কিনা।

65-1 Module Overview, node recap, mongodb vs MySql

**What is MONGODB?**

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License.

**What are MongoDB and MySQL?**

| MongoDB | MySQL |
| --- | --- |
| [MongoDB](https://www.geeksforgeeks.org/mongodb-an-introduction/) is an open-source database developed by MongoDB, Inc. MongoDB stores data in JSON-like documents that can vary in structure. It is a popular NoSQL database. | [MySQL](https://www.geeksforgeeks.org/php-mysql-database-introduction/) is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation. |

**How Data is Stored?**

| MongoDB | MySQL |
| --- | --- |
| In MongoDB, each individual records are stored as [‘documents’](https://www.geeksforgeeks.org/mongodb-getting-started/). | In MySQL, each individual records are stored as ‘rows’ in a table. |

**HIERARCHICAL UPPER OF A RECORD**

| MongoDB | MySQL |
| --- | --- |
| Documents belonging to a particular class or group as stored in a [‘collection’](https://www.geeksforgeeks.org/mongodb-getting-started/). Example: collection of users. | A ‘table’ is used to store rows (records) of similar type. |

**SQL or NoSQL**

| MongoDB | MySQL |
| --- | --- |
| MongoDB is what is called a [NoSQL database](https://en.wikipedia.org/wiki/NoSQL" \t "https://en.wikipedia.org/wiki/nosql). This means that pre-defined structure for the incoming data can be defined and adhered to but also, if required different documents in a collection can have different structures. It has a dynamic schema. | MySQL as the name suggests uses [Structured Query Language (SQL)](https://www.geeksforgeeks.org/structured-query-language/) for database access. The schema can not be changed. The inputs following the given schema are only entered. |

**Example:** In a table, if there are columns for **name, address** and there is need to enter **‘age’** in new column in one of the entries, it will not take it as the column is not defined in schema.  
This can be achieved in MongoDB, any new field can be inserted irrespective of the schema and is thus known to have dynamic schema.

**SALIENT FEATURES**

| MongoDB | MySQL |
| --- | --- |
| MongoDB was designed with high availability and scalability in mind, and includes out-of-the-box [replication](https://docs.mongodb.com/manual/replication) and [sharding](https://docs.mongodb.com/manual/sharding" \t "_blank). | MySQL concept does not allow efficient replication and sharding but in MySQL one can access associated data using joins which minimizes duplication. |

#### 65-2 Create mongodb atlas account and Connect to database

**What us NoSQL?**

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

| **SQL** | **NoSQL** |
| --- | --- |
| RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS) | Non-relational or distributed database system. |
| These databases have fixed or static or predefined schema | They have dynamic schema |
| These databases are not suited for hierarchical data storage. | These databases are best suited for hierarchical data storage. |
| These databases are best suited for complex queries | These databases are not so good for complex queries |
| Vertically Scalable | Horizontally scalable |
| Follows ACID property | Follows CAP(consistency, availability, partition tolerance) |

#### 65-3 Send data to database using mongodb insertOne

const uri =

  "mongodb+srv://hamid42:SktUkEqKFMX5qden@cluster0.1geox.mongodb.net/myFirstDatabase?retryWrites=true&w=majority";

const client = new MongoClient(uri, {

  useNewUrlParser: true,

  useUnifiedTopology: true,

  serverApi: ServerApiVersion.v1,

});

async function run(){

    try{

        await client.connect();

        const userCollection = client.db("foodExpress").collection("users");

        const user={name:"Hamid Hosen",email:"hamidhosen@gmail.com"};

        const result=await userCollection.insertOne(user);

        console.log(`User inserted with id: ${result.insertedId}`);

    }

    finally{

        // await client.close();

    }

}

run().catch(console.dir);

#### 65-4 (Recap) Async await, try catch and mongodb connection

#### 65-5 Send user data to the server and get response to client

        // POST USER: add a new user

        app.post('/user',async (req,res)=>{

            const newUser=req.body;

            console.log("adding new user",newUser);

            const result=await userCollection.insertOne(newUser);

            res.send(result);

            // res.send({result:"success"});

        });

#### 65-6 Save data from React Client side to mongo database

        app.get('/user',async (req,res)=>{

            const query={};

            const cursor=userCollection.find(query);

            const users=await cursor.toArray();

            res.send(users);

        })

#### 65-7 Load and display data from database, create delete button

#### 65-8 Delete a user from the database

        //delete a user

        app.delete('/user/:id',async(req,res)=>{

            const id=req.params.id;

            const query={\_id:ObjectId(id)};

            const result=await userCollection.deleteOne(query);

            res.send(result);

        })

  const handleUserDelete=(id)=>{

      const proceed=window.confirm("Are you sure you want to delete??");

      if(proceed)

      {

          console.log("Deleting user with id",id);

          const url = `http://localhost:5000/user/${id}`;

          fetch(url,{

              method:"DELETE"

          })

          .then(res=>res.json())

          .then(data=>{

             if(data.deletedCount>0)

             {

                 console.log("Deleted");

                 const remaining=users.filter(user=>user.\_id!==id);

                 setUser(remaining);

             }

          })

      }

  }

#### 65-9 (bonus) Load single item by id and Update user info

    //update user

    app.put("/user/:id", async (req, res) => {

      const id = req.params.id;

      const updateUser=reg.body;

      const filter = { \_id: ObjectId(id) };

      const options={upsert:true};

      const updateDoc={

          $set:{

              name:updateUser.name,

              email:updateUser.email

          }

      };

      const result = await userCollection.updateOne(filter, updateDoc, options);

      res.send(result);

    });