

# ASSIGNMENT - 3

## **OBJECTIVE :**

- CREATING A RESTFUL API USING EXPRESS JS
- RETRIEVING THE DATA USING MANGO DATA BASE

**The objective of this assignment is to familiarize yourself with building RESTful API using Express.js and using MongoDB as the database to store and retrieve data.**

## Creating a folder in express js:

Create a new folder for my project:mkdir your-api-project  
cd your-api-project Inside my project folder,

create the following folders and files:  
mkdir src  
cd src

touch app.jsNow, i place my Express.js code in the app.js file.Copy the Express.js code from the previous response into app.js.Install Express (if not installed already) and initialize my project:

npm init -y  
npm install express Update my package.json file to include a start script:  
Open package.json and add or update the "scripts" section:

```
"scripts": {  
  "start": "node src/app.js"  
}
```

To run my API, use the following command:

npm startNow you have a basic folder structure for your Express.js API project. I can expand this structure as my project grows, adding routes, controllers, models, etc., based on your my specific requirements.

We need to update our app.js to connect to MongoDB, so I was Modifying my app.js file to include the MongoDB connection using mongoose  
const express = require('express');  
const mongoose = require('mongoose');  
require('dotenv').config();

```
const app = express();  
const port = 3000;
```

## Creating restful api using express js :

```
const express = require('express');  
const app = express();  
const port = 3000;
```

```
app.use(express.json());
```

```
// Sample data
```

```
let data = [  
  { id: 1, name: 'Item 1' },  
  { id: 2, name: 'Item 2' },  
];
```

# Creating restful api using express js :

Get all items

```
app.get('/api/items', (req, res) => {  
  res.json(data);  
});
```

// Get a specific item by ID

```
app.get('/api/items/:id', (req, res) => {  
  const itemId = parseInt(req.params.id);  
  const item = data.find(item => item.id === itemId);  
  
  if (!item) {  
    return res.status(404).json({ message: 'Item not found' });  
  }  
  
  res.json(item);  
});
```

// Create a new item

```
app.post('/api/items', (req, res) => {  
  const newItem = req.body;  
  data.push(newItem);  
  res.status(201).json(newItem);  
});
```

// Update an existing item

```
app.put('/api/items/:id', (req, res) => {  
  const itemId = parseInt(req.params.id);  
  const updatedItem = req.body;  
  
  data = data.map(item => (item.id === itemId ? updatedItem : item));  
  
  res.json(updatedItem);  
});
```

// Delete an item

```
app.delete('/api/items/:id', (req, res) => {  
  const itemId = parseInt(req.params.id);  
  data = data.filter(item => item.id !== itemId);  
  
  res.json({ message: 'Item deleted successfully' });  
});
```

// Start the server

```
app.listen(port, () => {  
  console.log(`Server is running on port ${port}`);  
})
```

## **Connecting to mongo database :**

We need to update our app.js to connect to MongoDB, so I was Modifying my app.js file to include the MongoDB connection using mongoose

### **CODE FOR CONECTION :**

```
const express = require('express');
const mongoose = require('mongoose');
require('dotenv').config();

const app = express();
const port = 3000;

app.use(express.json());

// Connect to MongoDB
mongoose.connect(process.env.MONGODB_URI, { useNewUrlParser: true,
useUnifiedTopology: true });

const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

// ... Your existing routes and code

// Start the server
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

## **CONCLUSION :**

**In conclusion, WE set up a basic Express.js API with a folder structure for organization. By integrating MongoDB using Mongoose, MY API is now capable of performing CRUD operations with a database.**

**All the creation of rest api and connection to the mago data base are done then we have our fully functional database**