**Batch: D - 1 Roll No.: 16010122096**

**Experiment / assignment / tutorial No. 07**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| Title: Implementation of MongoDB, Node.js and Express js. |

**AIM:** Implementation of MongoDB, Node.js and Express js.

**Problem Definition:**

The goal of this experiment is to implement a basic web application using MongoDB as the database, Node.js as the runtime environment, and Express.js as the web framework. This setup will allow us to understand the integration of these technologies in building a full-stack web application.

**Resources used:**

 **Books:**

1. Shelly Powers, Learning Node, O’Reilly, 2nd Edition, 2016.

 **Websites:**

1. MongoDB Official Documentation
2. Node.js Official Documentation

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO 4:** **Test the concepts and components of various front-end, back-end web app**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Shelly Powers Learning Node O’ Reilly 2 nd Edition, 2016.

**Pre Lab/ Prior Concepts:**

**Write details about the following content**

* Mongo DB

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It allows for the storage of unstructured or semi-structured data, making it easier to scale and work with large volumes of information. Key features include:

* Schema-less data model.
* Powerful querying capabilities.
* High availability and scalability.
* Connection using node js Express js And MongoDB

To connect MongoDB with a Node.js application, we typically use the Mongoose library, which provides a straightforward way to model application data. Express.js acts as the server framework, handling HTTP requests and responses, while Mongoose facilitates communication with the MongoDB database.

**Methodology:**

 Set up the development environment with Node.js and MongoDB.

 Create a new Node.js project and install necessary dependencies (Express and Mongoose).

 Establish a connection between Node.js and MongoDB using Mongoose.

 Create a basic Express.js server that handles CRUD operations (Create, Read, Update, Delete).

 Implement routes to interact with the database.

 Test the application to ensure data is correctly stored and retrieved.

**Implementation Details:**

Students have to write stepwise details of implementation.

Item.js:

const mongoose = require('mongoose');

const itemSchema = new mongoose.Schema({

  name: { type: String, required: true },

  quantity: { type: Number, required: true }

});

module.exports = mongoose.model('Item', itemSchema);

server.js:

const express = require('express');

const mongoose = require('mongoose');

const Item = require('./models/Item');

const app = express();

const PORT = process.env.PORT || 3000;

*// Middleware*

app.use(express.json());

*// Connect to MongoDB*

mongoose.connect('mongodb://localhost:27017/mydatabase')

  .then(() => console.log('MongoDB connected'))

  .catch(*err* => console.log(err));

*// Create Item*

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

  const item = new Item(*req*.body);

  try {

    await item.save();

*res*.status(201).send(item);

  } catch (error) {

*res*.status(400).send(error);

  }

});

*// Get all Items*

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

  try {

    const items = await Item.find();

*res*.send(items);

  } catch (error) {

*res*.status(500).send(error);

  }

});

*// Update Item*

app.patch('/items/:id', async (*req*, *res*) => {

  try {

    const item = await Item.findByIdAndUpdate(*req*.params.id, *req*.body, { new: true });

    if (!item) {

      return *res*.status(404).send();

    }

*res*.send(item);

  } catch (error) {

*res*.status(400).send(error);

  }

});

*// Delete Item*

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

  try {

    const item = await Item.findByIdAndDelete(*req*.params.id);

    if (!item) {

      return *res*.status(404).send();

    }

*res*.sendStatus(204);

  } catch (error) {

*res*.status(500).send(error);

  }

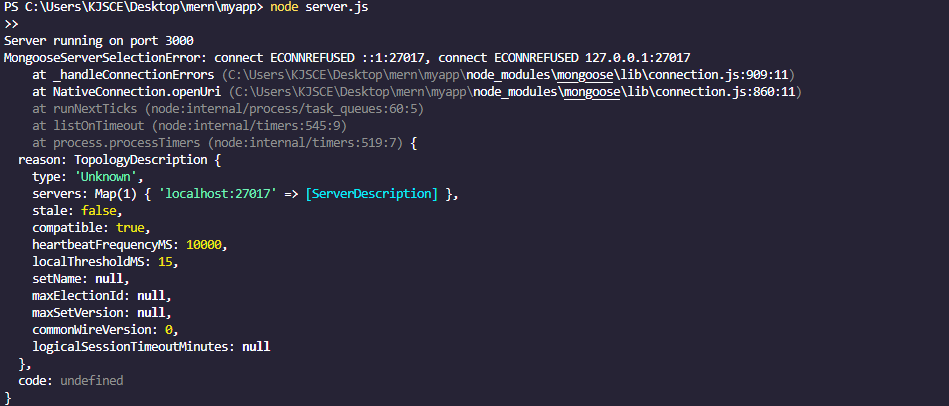
});

*// Start the server*

app.listen(PORT, () => {

  console.log(`Server running on port ${PORT}`);

});



**Conclusion:** This experiment successfully demonstrates the integration of MongoDB, Node.js, and Express.js, showcasing their effectiveness in building full-stack web applications.