**Practical-19**

**Aim:-**

Create a page which ask user to upload/Download the files.   
Note: Explore Multer Package

**Code:-**

const express = require("express");

const nodemailer = require("nodemailer");

const bodyParser = require("body-parser");

const app = express();

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

// Middleware for parsing JSON and form data

app.use(bodyParser.urlencoded({ extended: false }));

app.use(bodyParser.json());

// Setup Nodemailer with your email service provider's SMTP details

const transporter = nodemailer.createTransport({

  //   service: "YourEmailService", // Example: 'Gmail'

  host: "smtp.gmail.com",

  port: 587,

  secure: false,

  requireTLS: true,

  auth: {

    user: "darshaswani123@gmail.com",

    pass: "rnlrwktmndfvvzfg",

  },

});

// Serve a simple HTML form for email subscription

app.get("/", (req, res) => {

  res.send(`

    <h1>Email Subscription</h1>

    <form method="POST" action="/subscribe">

      <label>Email:</label>

      <input type="email" name="email" required>

      <button type="submit">Subscribe</button>

    </form>

  `);

});

// Handle POST request to subscribe

app.post("/subscribe", (req, res) => {

  const email = req.body.email;

  // Create email message

  const mailOptions = {

    from: "darshaswani123@gmail.com",

    to: email,

    subject: "Welcome to Our Newsletter",

    text: "Thank you for subscribing to our newsletter!",

  };

  // Send the email

  transporter.sendMail(mailOptions, (error, info) => {

    if (error) {

      console.error(error);

      res.status(500).send("Subscription failed. Please try again later.");

    } else {

      console.log("Email sent: " + info.response);

      res

        .status(200)

        .send("Subscription successful! Check your email for confirmation.");

    }

  });

});

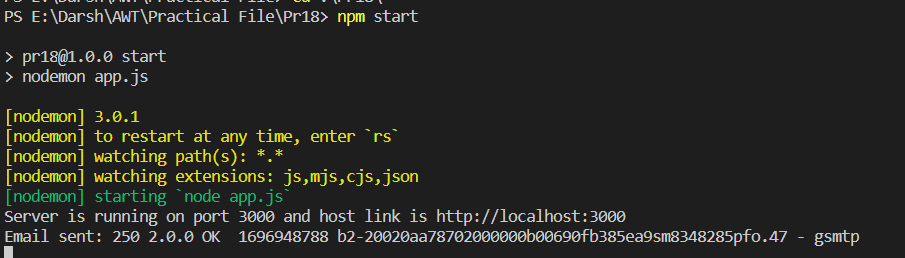
// Start the server

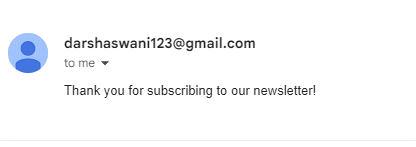
app.listen(PORT, () => {

  console.log(`Server is running on port ${PORT} and host link is http://localhost:${PORT}`);

});

**Output:-**



**Practical-20**

**Aim:-**

Create CRUD Operation page using ExpressJs, MongoDB and Sequelize ORM   
Framework

**Code:-**

**app.js:-**

require("./Database/dbconnection");

require("dotenv").config({ path: "../.env" });

const express = require("express");

const app = express();

const PORT = process.env.PORT;

app.use(express.urlencoded({ limit: "10mb", extended: true }));

app.use(express.json());

require("./Routes/ProjectRoute")(app);

app.listen(PORT, () => {

  console.log(`Server listening on : http://localhost:${PORT}`);

});

**dbconnection.js:-**

require("dotenv").config({ path: "./.env" });

const URL = process.env.DBURL;

const mongoose = require("mongoose");

mongoose

  .connect(URL, {

    useNewUrlParser: true,

    useUnifiedTopology: true,

  })

  .then(() => {

    console.log("Connected to Database..");

  })

  .catch((error) => {

    console.error("Error Connecting to the MongoDB : " + error);

  });

**model.js:-**

const mongoose = require("mongoose");

const projectSchema = mongoose.Schema({

  project\_name: {

    type: String,

  },

  project\_details: {

    type: String,

  },

  project\_links: {

    type: String,

  },

  project\_tags: { type: String },

  createdAt: { type: Date, default: Date.now },

  updatedAt: { type: Date, default: Date.now },

});

const Project = mongoose.model("Project", projectSchema);

module.exports = Project;

**route.js:-**

const {

  AddProject,

  GetProjects,

  GetProjectDetails,

  UpdateProject,

  DeleteProject,

} = require("../Controllers/ProjectController");

const ProjectRoutes = (app) => {

  app.post("/project/add-project", AddProject);

  app.get("/project/allprojects", GetProjects);

  app.get("/project/project/:id", GetProjectDetails);

  app.put("/project/update-project/:id", UpdateProject);

  app.delete("/project/delete-project/:id", DeleteProject);

};

module.exports = ProjectRoutes;

**controller.js:-**

const Project = require("../Models/Project");

exports.AddProject = async (req, res) => {

  try {

    const projectTags = req.body.project\_tags; // Array of project tags

    const projectName = req.body.project\_name; // Array of project tags

    const projectDetails = req.body.project\_details; // Array of project tags

    const projectLinks = req.body.project\_links; // Array of project links

    console.log(projectName, projectDetails, projectTags, projectLinks);

    const newProject = new Project({

      project\_name: projectName,

      project\_details: projectDetails,

      project\_links: projectLinks,

      project\_tags: projectTags,

    });

    const createdProject = await newProject.save();

    if (createdProject) {

      return res.status(200).json({

        success: true,

        Project: createdProject,

      });

    }

  } catch (error) {

    console.error(error);

    return res

      .status(500)

      .json({ success: false, message: "Internal server error.", error });

  }

};

exports.GetProjects = async (req, res) => {

  try {

    const projects = await Project.find(

      {},

      {

        \_id: 1,

        project\_name: 1,

        project\_details: 1,

        project\_links: 1,

        project\_tags: 1,

      }

    );

    return res.status(200).send({ success: true, projects });

  } catch (error) {

    res

      .status(500)

      .json({ success: false, message: "Internal server error.", error });

  }

};

exports.GetProjectDetails = async (req, res) => {

  try {

    const projectId = req.params.id;

    const project = await Project.findById(projectId);

    if (!project) {

      return res

        .status(404)

        .json({ success: false, message: "Project not found." });

    }

    const responseData = {

      project\_name: project.project\_name,

      project\_details: project.project\_details,

      project\_links: project.project\_links,

      project\_tags: project.project\_tags,

    };

    res.status(200).json({

      success: true,

      data: responseData,

    });

  } catch (error) {

    res

      .status(500)

      .json({ success: false, message: "Internal server error.", error });

  }

};

exports.UpdateProject = async (req, res) => {

  try {

    const projectId = req.params.id;

    const { project\_name, project\_details, project\_links, project\_tags } =

      req.body;

    let project;

    project = await Project.findById(projectId);

    if (project\_name !== project.project\_name) {

      project.project\_name = project\_name;

    }

    if (project\_details !== project.project\_details) {

      project.project\_details = project\_details;

    }

    if (project\_tags !== project.project\_tags) {

      project.project\_tags = project\_tags;

    }

    if (project\_links !== project.project\_links) {

      project.project\_links = project\_links;

    }

    project.updatedAt = Date.now();

    const updatedProject = await project.save();

    // console.log(updatedProject);

    return res.status(200).json({ success: true, updatedProject });

  } catch (error) {

    console.log(error);

    return res

      .status(500)

      .json({ success: false, message: "Internal server error.", error });

  }

};

exports.DeleteProject = async (req, res) => {

  try {

    const projectId = req.params.id;

    const project = await Project.findById(projectId);

    if (!project) {

      console.log(1);

      return res

        .status(400)

        .json({ success: false, message: "Project not found" });

    }

    const deletedProject = await Project.findByIdAndDelete(projectId);

    return res

      .status(200)

      .json({ success: true, message: "Project deleted successfully" });

  } catch (error) {

    console.log(error);

    return res

      .status(500)

      .json({ success: false, message: "Internal server error.", error });

  }

};

**OutPut:-**

