**Practical-15**

**Aim:-**

Create Secure Server using Nodejs and ExpressJs.   
Note: Explore nodemon package to handle server automatically.  
**Code:-**

const express = require("express");

const helmet = require("helmet");

const https = require("https");

const path = require("path");

const fs = require("fs");

const hostname = "127.0.0.81";

const port = 3000;

//express object

const app = express();

app.use(helmet());

//Urlencoder

app.use(

  express.urlencoded({

    extended: true,

  })

);

//Routes

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

  res.send("Welcom to SSL Server!");

});

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

  res.send("This is SSL Cretificate!");

});

//SSL Server Code

const sslServer = https.createServer({

  key: fs.readFileSync(path.join(\_\_dirname, 'certi', "key.pem")),

  cert: fs.readFileSync(path.join(\_\_dirname, 'certi', "cert.pem"))

},app)

sslServer.listen(port, () => {

  console.log(`server is running 🚀 on https://${hostname}:${port}/`);

});

**Practical-16**

**Aim:-**

Create Login and Registration using ExpressJs.

• Apply Session, Cookies and JWT Token Concept.   
• Apply Passport.js for Authentication middleware

**Code:-**

const express = require("express");

const app = express();

const port = 3000;

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

const session = require("express-session");

const cookieParser = require("cookie-parser");

const jwt = require("jsonwebtoken");

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

app.use(cookieParser());

app.use(

  session({

    secret: "secret",

    resave: true,

    saveUninitialized: true,

  })

);

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

  res.send("Hello World!");

});

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

  res.sendFile(\_\_dirname + "/Views/login.html");

});

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

  res.sendFile(\_\_dirname + "/Views/register.html");

});

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

  const username = req.body.username;

  const password = req.body.password;

  if (username && password) {

    const user = {

      username: username,

      password: password,

    };

    const token = jwt.sign(user, "secretkey", (err, token) => {

      res.cookie("token", token);

      res.redirect("/dashboard");

    });

  } else {

    res.send("Please enter Username and Password!");

  }

});

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

  const username = req.body.username;

  const password = req.body.password;

  if (username && password) {

    const user = {

      username: username,

      password: password,

    };

    const token = jwt.sign(user, "secretkey", (err, token) => {

      res.cookie("token", token);

      res.redirect("/dashboard");

    });

  } else {

    res.send("Please enter Username and Password!");

  }

});

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

  if (req.cookies.token) {

    const token = req.cookies.token;

    const decoded = jwt.verify(token, "secretkey");

    res.send(`Welcome ${decoded.username}`);

  } else {

    res.send("Please login to view this page!");

  }

});

app.listen(port, () => {

  console.log(`Example app listening at http://localhost:${port}`);

});

**Practical-17**

**Aim:-**

Create Coure page which contains University Name, Institute Name,   
Department Name, CoursName, CourseCode, Semester. Handle.   
   
Note:   
• Create Middleware to Sanitize and validate All fileds  
**Code:-**

const express = require("express");

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

const { body, validationResult } = require("express-validator");

const cors = require("cors");

const app = express();

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

app.use(

  cors({

    origin: "http://localhost:3000",

    credentials: true,

  })

);

// Middleware for sanitization and validation

const sanitizeAndValidate = [

  body("university").trim().escape(),

  body("institute").trim().escape(),

  body("department").trim().escape(),

  body("courseName").trim().escape(),

  body("courseCode").trim().escape(),

  body("semester").trim().escape(),

  (req, res, next) => {

    const errors = validationResult(req);

    if (!errors.isEmpty()) {

      return res.status(400).json({ errors: errors.array() });

    }

    next();

  },

];

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

  const {

    university,

    institute,

    department,

    courseName,

    courseCode,

    semester,

  } = req.body;

  const newCourse = {

    university,

    institute,

    department,

    courseName,

    courseCode,

    semester,

  };

  // Handle the course data (e.g., store it in a database)

  res.json({

    success: true,

    message: "Course details saved successfully.",

    course: newCourse,

  });

});

app.listen(4000, () => {

  console.log("Server started on port 3000");

});