Practical No. 8

Study and implementation of node.js

Perform following problem statements using Node.js

Problem Statement 1: Introduction to Node.js

- What is Node.js, and how does it differ from traditional server-side platforms like Apache or PHP?
- What is the purpose of the V8 engine in Node.js?
- Explain the single-threaded, event-driven architecture of Node.js.
- Why is Node.js considered non-blocking?
- What is npm, and how is it used in Node.js?
- What is a module in Node.js? How do you export and import modules?
- What is the difference between require() and import in Node.js?
- How can you create a custom module in Node.js?
- What is the role of the package.json file in a Node.js project?
- How do you install a package globally and locally using npm?
- What is the difference between asynchronous and synchronous programming in Node.js?
- How do you create an HTTP server in Node.js?
- What is the difference between http.createServer() and using frameworks like Express.js?
- How do you handle GET and POST requests in Node.js?

Problem Statement 2: Middleware (Express.js)

- What is middleware in Node.js, particularly in the context of Express.js?
- How do you create custom middleware in Express.js?
- Explain how middleware is executed in order in an Express.js application.

Problem Statement 3: File System (fs) Module

- How do you read and write files using the fs module in Node.js?
- What is the difference between fs.readFile() and fs.readFileSync()?
- How can you check if a file or directory exists in Node.js?
- How do you handle file operations in an asynchronous manner?

Problem Statement 4: Database Connectivity

- How do you connect to a MongoDB database from a Node.js application?
- What is the purpose of the mongoose library in Node.js?

- Explain how you would perform basic CRUD operations (Create, Read, Update, Delete) using MongoDB and Node.js.

Problem Statement 5: Building a RESTful API

Develop a RESTful API using Node.js and Express.js for a library management system. The system should allow users to:

- Add new books (title, author, genre, year of publication).
- Update book details.
- Delete books from the collection.
- Fetch a list of books with pagination and filtering by genre and author.
- Add a user authentication system to restrict access to certain API routes.

File Upload and Management System

Build a file upload and management system using Node.js and Multer (or any other file upload middleware). The system should allow users to:

- Upload files (images, PDFs, etc.).
- View the list of uploaded files.
- Download or delete specific files.
- Implement user authentication so that only authorized users can upload and manage files.

1. Create a **document** of the above questions with screenshots wherever necessary.
2. Scan the document and **create a pdf file** with **"ExamSeatNum_P#PS#" as its name**.
3. Upload the file on the **WCE /ERP** before the given deadline.