**Batch T7**

**Practical No. 9**

**Title of Assignment: Study and implementation of node.js**

**Student Name: Shreyash Patil**

**Student PRN: 22510039**

Problem Statement 1: Database Connectivity using SQL or Oracle

Write a Node.js program that connects to an Oracle/SQL database, retrieves data

from a table, and displays the results.

Ans.



Problem Statement 2: Middleware (Express.js)

What is middleware in Node.js, particularly in the context of Express.js?

Functions that have access to request, response objects and next() function

* Executes between receiving request and sending response
* Can modify req/res objects, end request cycle, call next middleware

How do you create custom middleware in Express.js?

const customMiddleware = (req, res, next) => {

// Do something

next();

};

Explain how middleware is executed in order in an Express.js application.

Executes sequentially from top to bottom

* app.use() applies middleware to all routes
* Route-specific middleware only executes for matched routes
* Error handling middleware has 4 parameters (err, req, res, next)

Problem Statement 3: File System (fs) Module

How do you read and write files using the fs module in Node.js?

fs.readFile() - Async read

* fs.writeFile() - Async write
* fs.appendFile() - Append content

What is the difference between fs.readFile() and fs.readFileSync()?

readFile(): Asynchronous, non-blocking

* readFileSync(): Synchronous, blocks execution

How can you check if a file or directory exists in Node.js?

fs.existsSync() - Sync check

* fs.access() - Async check

How do you handle file operations in an asynchronous manner?

Use promises with fs.promises

* Use async/await
* Handle errors with try/catch
* Use callbacks with proper error handling

Problem Statement 4: File Upload and Download API

Develop a file upload and download API using Node.js and Express. The API should allow

users to upload files (e.g., images, documents) and download them later.

Create an API to upload files to the server.

Implement routes to retrieve and download files.

Ensure proper error handling (e.g., file size limits, invalid file formats).

Implement file versioning to allow multiple uploads of the same file name

without overwriting.

Key aspects to implement:

1. Upload:

* Use multer middleware
* Validate file types
* Set size limits
* Generate unique filenames

1. Download:

* Stream large files
* Set proper headers
* Handle missing files

1. Error handling:

* Check file existence
* Validate file sizes
* Handle corrupt files
* Proper HTTP status codes

1. File versioning:

* Use timestamps in filenames
* Store metadata in database
* Implement version tracking
* Keep version history

Problem Statement 5: Real-time Chat Application with Socket.io

Create a real-time chat application using Node.js, Express, and Socket.io that allows

multiple users to join and communicate in a chat room.

Set up a Node.js server with Socket.io for real-time bi-directional

communication.

Implement event listeners to handle user connections, disconnections, and

message broadcasting to all connected users.

Server setup with Socket.io:

* Handle connections
* Manage rooms
* Track active users
* Implement message broadcasting

1. Event handling: Key events to implement:

* connection: New user joins
* disconnect: User leaves
* message: Broadcast to room
* typing: User typing indicator
* joinRoom: Room management
* privateMessage: Direct messages

Best practices:

* Implement error handling
* Add user authentication
* Store chat history
* Handle reconnections
* Implement rate limiting
* Add message validation