Practical-8

Demonstrate Node.js using MySQL to perform the below operations: Create a database, create a table, insert a record, and update the record.

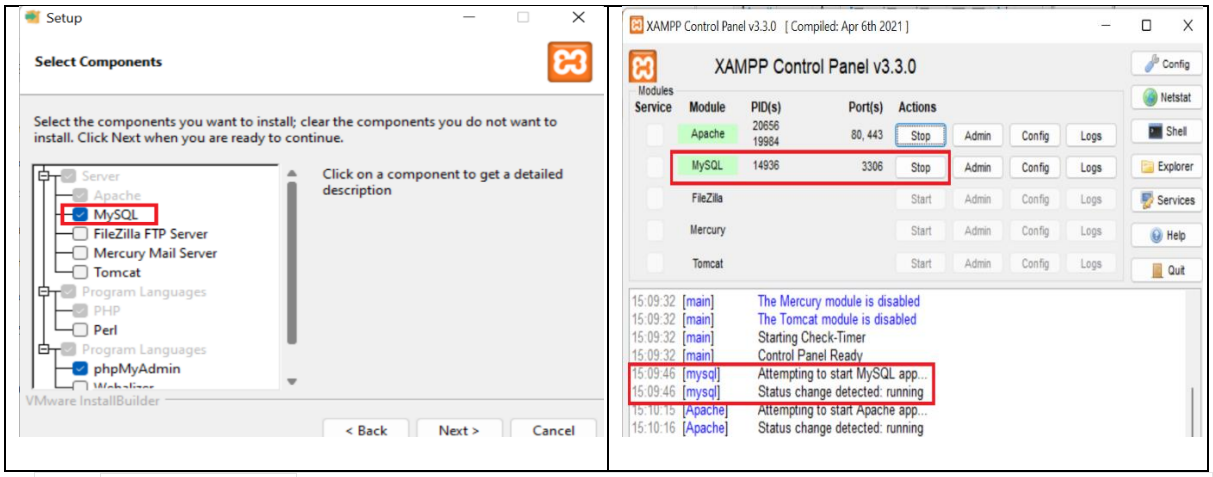
Step-1. Install Node.js

Step-2. Verify that Node.js was properly installed with the following

Command in CMD: node --version

Step-3. Verify whether NPM (Node Package Manager) is installed or not with the following command in CMD: npm –version

Step-4. MySQL Installation : download it from https://www.apachefriends.org/index.html



1. **Start XAMPP**: Open the XAMPP control panel and start the Apache and MySQL services if they are not already running.
2. **Access phpMyAdmin**: Open your web browser and go to `http://localhost/phpmyadmin/`. This will open phpMyAdmin, a web-based MySQL administration tool that is included with XAMPP.

Step-5 Create a new folder for your project: Open Visual Studio Code and create a new folder where you want to store your project files.

Step-6 Open Terminal in Visual Studio Code: Once you have your folder open in Visual Studio Code, open the integrated terminal. You can do this by selecting Terminal > New Terminal from the menu bar.

Step-7 Initialize a new Node.js project: In the terminal, navigate to your project folder and run the following command to initialize a new Node.js project:

```
npm init -y
```

Step-8 Install the MySQL package: Install the mysql package using npm by running the following command in the terminal:

```
npm install mysql
```

Step-9 Create a new JavaScript file: Create a new file app.js. This file will contain your Node.js code.

File name: app.js

```
// Import required modules
const mysql = require('mysql');

// Create a connection to the MySQL server
const connection = mysql.createConnection({
  host: 'localhost', // Change to your MySQL host
  user: 'root', // Change to your MySQL username
  password: '', // Change to your MySQL password
  database: 'my_database' // Change to your MySQL database name
});

// Connect to MySQL server
connection.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL server: ' + err.stack);
    return;
  }
  console.log('Connected to MySQL server as ID ' +
connection.threadId);
});

// Create a database
```

```
connection.query('CREATE DATABASE IF NOT EXISTS my_database', (err,
results) => {
  if (err) throw err;
  console.log('Database created or already exists');
});

// Create a table
connection.query(`CREATE TABLE IF NOT EXISTS my_table (id INT
AUTO_INCREMENT PRIMARY KEY,name VARCHAR(255))`,
(err, results) => {
  if (err) throw err;
  console.log('Table created or already exists');
});

// Insert a record
connection.query('INSERT INTO my_table (name) VALUES (?)', ['John
Doe'], (err, results) => {
  if (err) throw err;
  console.log('Record inserted:', results);
});

// Update a record
connection.query('UPDATE my_table SET name = ? WHERE id = ?',
['Devendra Vashi', 1], (err, results) => {
  if (err) throw err;
  console.log('Record updated:', results);
});

// Close the connection
connection.end((err) => {
  if (err) {
    console.error('Error closing connection: ' + err.stack);
    return;
  }
  console.log('Connection closed');
});
```

Step-10 Modify the MySQL connection details: Modify the MySQL connection details in the code to match your MySQL server configuration. Update the host, user, password, and database properties in the createConnection function call.

Step-11 Run the script: Save your app.js file and return to the terminal in Visual Studio Code. Run the script by executing the following command:

```
node app.js
```

Verify the output: Check the output in the terminal to see if the script executed successfully. You should see messages indicating the connection status, database creation, table creation, record insertion, record update, and the closure of the connection.