**Project Development Phase**

**Model Performance Test**

|  |  |
| --- | --- |
| Date | 16 July 2025 |
| Team ID | LTVIP2025TMID59690 |
| Project Name | HouseHunt – Rental Property Recommender |
| Maximum Marks | 2 Marks |

**Node.js:**

**1. Introduction to Node.js:**

Node.js is a JavaScript runtime built on Chrome’s V8 engine that enables server-side execution of JavaScript code. It allows developers to build scalable and efficient network applications using a single programming language across the entire stack.

As part of the MERN stack, Node.js forms the core of the backend environment, powering Express.js and enabling seamless communication with MongoDB and the frontend React application.

**2. Why Node.js Was Used:**

Node.js was selected for the backend of this MERN stack application due to:

* Its **non-blocking, event-driven architecture** for handling concurrent requests.
* Ability to use **JavaScript on both frontend and backend** (single language full stack).
* High performance for real-time web apps.
* Strong **NPM ecosystem** for using modules like jsonwebtoken, mongoose, cors, etc.
* Easy integration with MongoDB and Express.js

**3. How Node.js Was Used in the Project:**

In this project, Node.js was used to:

* Serve as the runtime environment for the backend Express server.
* Handle server-side logic for user authentication, data handling, and business logic.
* Connect to the MongoDB database using **Mongoose**.
* Implement middleware for parsing requests (body-parser, cors).
* Organize API routing and connect controllers for each user role (Admin, Owner, Renter).

**4.** **Sample Server Code Snippet:**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 5000;

// Middleware

app.use(cors());

app.use(express.json());

// MongoDB Connection

mongoose.connect(process.env.MONGO\_URI)

.then(() => console.log('MongoDB connected'))

.catch(err => console.log(err));

// Routes

app.use('/api/users', require('./routes/userRoutes'));

app.use('/api/properties', require('./routes/propertyRoutes'));

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

**5. Conclusion:**

Node.js formed the backbone of the backend in this full stack MERN application. Its efficient, event-driven model enabled scalable backend operations while allowing consistent use of JavaScript across the stack**.**