**Project Development Phase**

**Model Performance Test**

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

**JWT Authentication:**

**1. Introduction to JWT Authentication:**

JWT (JSON Web Token) is a secure, compact, and self-contained way of transmitting information between parties. In web development, JWT is commonly used for authentication and authorization, especially in stateless RESTful APIs.

JWT allows the backend to verify requests based on tokens instead of storing session information on the server.

**2. Why JWT Was Used:**

JWT was implemented in this MERN stack application to:

* Enable **stateless user authentication**.
* Secure access to protected routes based on user roles (Admin, Owner, Renter).
* Transmit user identity securely between frontend (React) and backend (Node.js).
* Allow token-based login without the need to store sessions in memory.

**3.** **How JWT Was Used in the Project:**

JWT Authentication was used in the following way:

* After a successful login, the server generates a **JWT token** signed with a secret key.
* The token is sent to the frontend and stored (e.g., in localStorage).
* On protected route access, the token is passed via **Authorization header** (Bearer <token>).
* The backend middleware verifies the token before granting access.

**4. Sample Code Snippets:**

**Generating a Token**

const jwt = require('jsonwebtoken');

const generateToken = (userId, role) => {

return jwt.sign({ id: userId, role }, process.env.JWT\_SECRET, {

expiresIn: '1d',

});

};

**Verifying a Token (Middleware)**

const jwt = require('jsonwebtoken');

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

const token = req.headers.authorization?.split(' ')[1];

if (!token) return res.status(401).json({ message: 'No token provided' });

try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

req.user = decoded;

next();

} catch (error) {

return res.status(401).json({ message: 'Invalid token' });

}

};

**5. Conclusion:**

JWT Authentication provided a secure and scalable method for managing user sessions in this MERN stack application. It helped restrict access based on roles and ensured API security without maintaining session data on the server.