JWT Implementation – Short Explanation (Interview Answer)

In this application, JWT is used for **stateless authentication**.
During **login**, the user's credentials are verified using **bcrypt** by comparing the entered password with the hashed password stored in the database.
If the credentials are valid, a **JWT token** is generated using jsonwebtoken, where the payload contains the user's **id and role**.
This token is signed using a **secret key** (JWT_SECRET) and has an **expiry of 1 day**.
The generated token is then sent to the client as an **HTTP-only cookie**, which improves security by preventing access from JavaScript.
On subsequent requests, this token can be verified to authenticate the user and perform **role-based authorization** without storing session data on the server.

One-line flow (very effective in interviews)

Register → hash password → store user → login → verify password → generate JWT → send token in cookie → use token for protected routes

---

Key points to highlight if asked

- Passwords are **hashed with bcrypt**
- JWT contains **user ID and role**
- Token is **signed and expires**
- Stored in **HTTP-only cookie** (prevents XSS)
- Enables **stateless authentication**

If you want, I can also give:

- **JWT middleware explanation**
- **Role-based authorization example**
- **Why cookie vs localStorage**
- **JWT vs session (interview comparison)**

Login – Create JWT

```
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');

router.post('/login', async (req, res) => {
  const user = await User.findOne({ email: req.body.email });

  if (!user || !(await bcrypt.compare(req.body.password, user.password)))
    return res.status(401).json({ msg: "Invalid credentials" });

  const token = jwt.sign(
    { id: user._id, role: user.role },
    process.env.JWT_SECRET,
    { expiresIn: '1d' }
  );

  res.cookie('token', token, { httpOnly: true });
  res.json({ msg: "Login successful" });
});
```

JWT Middleware – Verify Token

```
const jwt = require('jsonwebtoken');

const authMiddleware = (req, res, next) => {
  const token = req.cookies.token;
  if (!token) return res.status(401).json({ msg: "No token" });

  const decoded = jwt.verify(token, process.env.JWT_SECRET);
  req.user = decoded;
  next();
};
```

3 Protected Route

```
router.get('/profile', authMiddleware, (req, res) => {
  res.json({ userId: req.user.id, role: req.user.role });
});
```