Experiment 6.2: Implement Protected Routes with JWT Verification

Objective:

To implement protected routes in a Node.js application using JWT (JSON Web Token) verification for authentication.

Theory:

JSON Web Tokens (JWT) are a secure way to transmit information between a client and server. They are often used for authentication in RESTful APIs. In this experiment, users log in to receive a JWT, which must be included in the Authorization header to access protected routes.

Code Implementation:

```
const express = require('express');
const jwt = require('jsonwebtoken');
const bcrypt = require('bcryptjs');
require('dotenv').config();

const app = express();
app.use(express.json());
const PORT = process.env.PORT || 5000;

const users = [
  {
    id: 1,
    username: "ayush",
    password: "$2a$10$Qxz/jav5Hrq8Ds5HddQHve2eqGJmDYgZCsy5tz8K4OeG5duh9GdaK"
  }
];

app.post('/login', async (req, res) => {
  const { username, password } = req.body;
  const user = users.find(u => u.username === username);
  if (!user) return res.status(400).json({ message: "User not found" });

  const isMatch = await bcrypt.compare(password, user.password);
  if (!isMatch) return res.status(401).json({ message: "Invalid credentials" });

  const token = jwt.sign(
    { id: user.id, username: user.username },
    process.env.JWT_SECRET || "mysupersecretkey",
    { expiresIn: '1h' }
  );

  res.json({ message: "Login successful", token });
});

const verifyToken = (req, res, next) => {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token)
    return res.status(403).json({ message: "Access denied. No token provided." });

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET || "mysupersecretkey");
    req.user = decoded;
    next();
  } catch (err) {
    res.status(401).json({ message: "Invalid or expired token" });
```

```
  }
};

app.get('/user/profile', verifyToken, (req, res) => {
  res.json({
    message: `Welcome ${req.user.username}`,
    user: req.user
  });
});

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Learning Outcome:

By performing this experiment, students learn how to secure backend routes using JWT authentication, how to verify tokens using middleware, and how to control access to protected endpoints.