Task 6: Database Integration and User Authentication.

Objective: Integrate a database and implement user authentication for secure data handling.

Node.js with Express and MongoDB

```
require('dotenv').config();

const express = require('express');

const mongoose = require('mongoose');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');


const app = express();

app.use(express.json());


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

const MONGO_URI = process.env.MONGO_URI;

const JWT_SECRET = process.env.JWT_SECRET;


// Connect to MongoDB

mongoose.connect(MONGO_URI, { useNewUrlParser: true,
useUnifiedTopology: true })

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

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


// User Schema

const userSchema = new mongoose.Schema({
```

```
    username: String,
    email: String,
    password: String
});

const User = mongoose.model('User', userSchema);

// User Registration Route
app.post('/register', async (req, res) => {
    const { username, email, password } = req.body;

    // Hash password
    const hashedPassword = await bcrypt.hash(password, 10);

    const user = new User({ username, email, password: hashedPassword });
    await user.save();

    res.status(201).json({ message: 'User registered successfully' });
});

// User Login Route
app.post('/login', async (req, res) => {
    const { email, password } = req.body;

    const user = await User.findOne({ email });
    if (!user) return res.status(400).json({ message: 'Invalid email or password' });
```

```
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(400).json({ message: 'Invalid email or password' });

    const token = jwt.sign({ userId: user._id }, JWT_SECRET, { expiresIn: '1h' });
    res.json({ token });
});

// Middleware for Authentication
const authMiddleware = (req, res, next) => {
    const token = req.header('Authorization');
    if (!token) return res.status(401).json({ message: 'Unauthorized' });

    try {
        const decoded = jwt.verify(token, JWT_SECRET);
        req.user = decoded;
        next();
    } catch (err) {
        res.status(401).json({ message: 'Invalid token' });
    }
};

// Protected API Route
app.get('/protected', authMiddleware, (req, res) => {
    res.json({ message: 'Welcome to the protected route', user: req.user });
});
```

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