

## User.js

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  }
});
module.exports = mongoose.model('User', userSchema);
```

## userRoutes.js

```
const express = require('express');
const router = express.Router();
const User = require('../models/User');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');

// Register User
router.post('/register', async (req, res) => {
  try {
    const { username, password, email } = req.body;
    // Check if user already exists
    const existingUser = await User.findOne({ $or: [{ username }, { email }] });
    if (existingUser) {
      return res.status(400).json({ message: 'Username or email already exists' });
    }
    // Encrypt password
    const hashedPassword = await bcrypt.hash(password, 10);
    // Create new user
    const user = new User({
      username,
      password: hashedPassword,
      email
    });
    await user.save();
    res.status(201).json({ message: 'User registered successfully' });
  } catch (error) {
    res.status(500).json({ message: 'Server error' });
  }
});
```

```

// Login User
router.post('/login', async (req, res) => {
  try {
    const { username, password } = req.body;
    const user = await User.findOne({ username });
    if (!user || !(await bcrypt.compare(password, user.password))) {
      return res.status(401).json({ message: 'Invalid credentials' });
    }
    const token = jwt.sign({ id: user._id }, 'your_jwt_secret', { expiresIn: '1h' });
    res.json({ token, user: { _id: user._id, username: user.username, email: user.email } });
  } catch (error) {
    res.status(500).json({ message: 'Server error' });
  }
});

// Get User Data
router.get('/profile/:id', async (req, res) => {
  try {
    const user = await User.findById(req.params.id).select('-password');
    if (!user) {
      return res.status(400).json({ message: 'User not found' });
    }
    res.json(user);
  } catch (error) {
    res.status(500).json({ message: 'Server error' });
  }
});

// Update User
router.put('/update/:id', async (req, res) => {
  try {
    const { username, email } = req.body;
    const user = await User.findByIdAndUpdate(
      req.params.id,
      { username, email },
      { new: true }
    ).select('-password');
    if (!user) {
      return res.status(400).json({ message: 'User not found' });
    }
    res.json({ message: 'User updated successfully', user });
  } catch (error) {
    res.status(500).json({ message: 'Server error' });
  }
});

module.exports = router;

```

### server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const app = express();

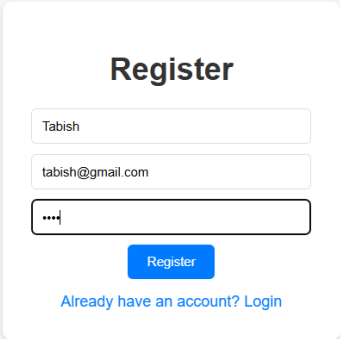
// Middleware
app.use(express.json()); // Lets the server read JSON data from requests
app.use(cors()); // Allows Angular to connect to this server

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/userdb', {
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(() => console.log('MongoDB connected'))
.catch(err => console.log('MongoDB error:', err));

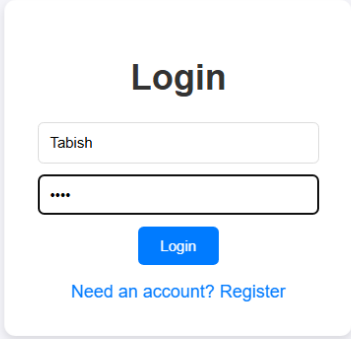
// Load routes
const userRoutes = require('./routes/userRoutes');
app.use('/api/users', userRoutes);

// Start the server
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

### Screenshots of Application:



The screenshot shows a 'Register' form with a light gray background. The form is white and contains three input fields: a text field with 'Tabish', an email field with 'tabish@gmail.com', and a password field with '\*\*\*'. Below the inputs is a blue 'Register' button. At the bottom, there is a link that says 'Already have an account? Login'.



The screenshot shows a 'Login' form with a light gray background. The form is white and contains two input fields: a text field with 'Tabish' and a password field with '\*\*\*'. Below the inputs is a blue 'Login' button. At the bottom, there is a link that says 'Need an account? Register'.

