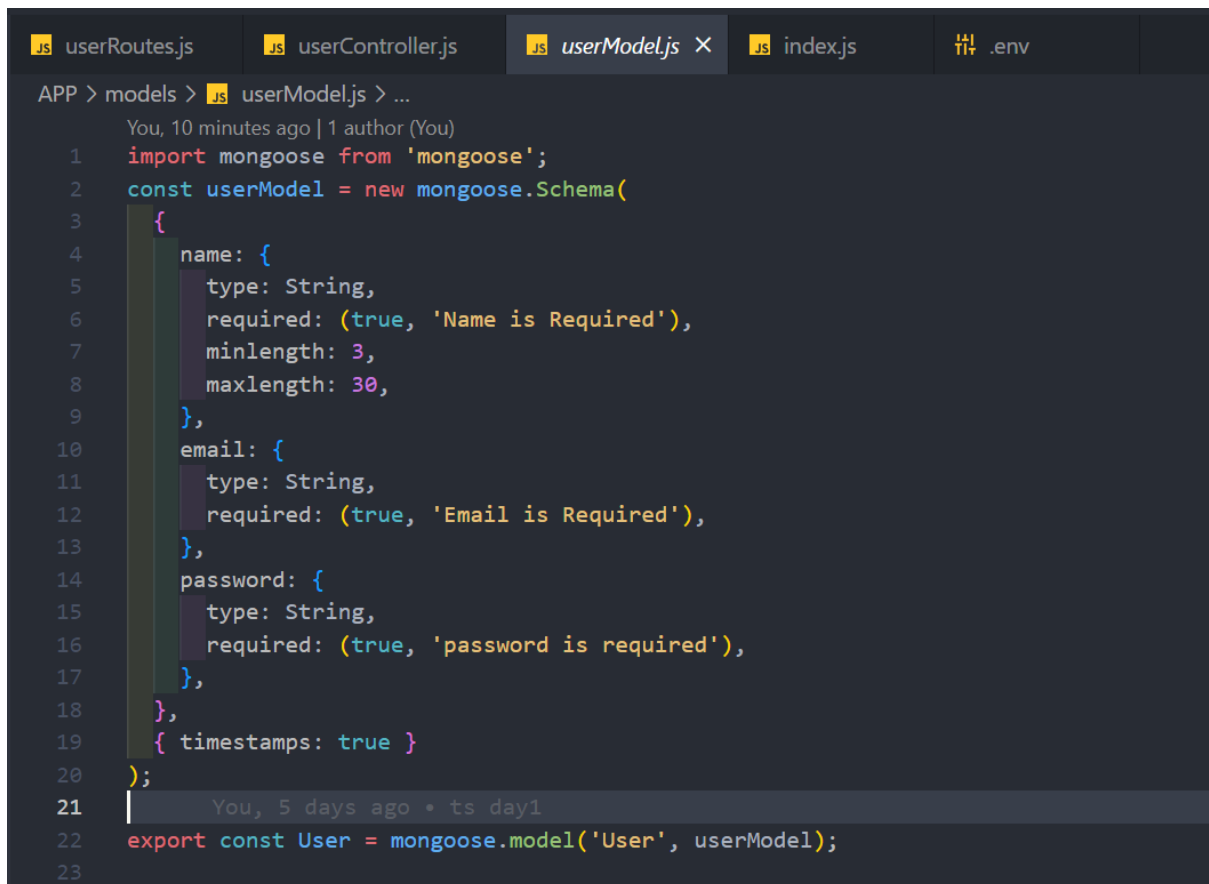


Update password:



```
userRoutes.js  userController.js  userModel.js X  index.js  .env

APP > models > userModel.js > ...
You, 10 minutes ago | 1 author (You)
1  import mongoose from 'mongoose';
2  const userModel = new mongoose.Schema(
3    {
4      name: {
5        type: String,
6        required: (true, 'Name is Required'),
7        minlength: 3,
8        maxlength: 30,
9      },
10     email: {
11       type: String,
12       required: (true, 'Email is Required'),
13     },
14     password: {
15       type: String,
16       required: (true, 'password is required'),
17     },
18   },
19   { timestamps: true }
20 );
21   You, 5 days ago • ts day1
22 export const User = mongoose.model('User', userModel);
23
```

```
JS userRoutes.js JS userController.js X JS userModel.js JS index.js .env
APP > controllers > JS userController.js > ...
72 const updatePassword = async (req, res) => {
73   try {
74     const { id } = req.params;
75     const { password } = req.body;
76
77     // const user = await User.find({ _id: id });
78     // console.log(user[0].password);
79
80     const user = await User.findById(id);
81     if (user.password === password) {
82       res.json({
83         message: 'old and new password are same. Enter a New Password',
84       });
85     } else {
86       const updatePwd = await User.findByIdAndUpdate(
87         id,
88         { password },
89         { new: true }
90       );
91       res.json({ message: 'password updated successfully', data: updatePwd });
92     }
93   } catch (err) {
94     res.json({
95       msg: err.message,
96     });
97   }
98 };
99
```

```
JS userRoutes.js X JS userController.js JS userModel.js JS index.js
APP > routes > JS userRoutes.js > ...
5  import {
6      createUser,
7      getUsers,
8      getUser,
9      deleteUser,
10     updateUser,
11     updatePassword,
12 } from '../controllers/userController.js';
13
14 userRouter.post('/create-user', createUser);
15 userRouter.get('/all-users', getUsers);
16 userRouter.get('/user/:id', getUser);
17 userRouter.delete('/user-delete/:id', deleteUser);
18 userRouter.put('/user-update/:id', updateUser);
19 userRouter.put('/update-pwd/:id', updatePassword);
20
21 export default userRouter;
22
```

Usage of bcrypt:

Step1:install bcrypt -> npm i bcrypt

Step2: inside the model , create a method to hash the password

Step3:use this method to validate the password

JS userRoutes.js JS userController.js JS userModel.js X JS index.js .env

APP > models > JS userModel.js > ...

You, 5 minutes ago | 1 author (You)

```
1 import mongoose from 'mongoose';
2 import bcrypt from 'bcrypt';
3 const userModel = new mongoose.Schema(
4   {
5     >   name: { ...
10     },
11     >   email: { ...
14     },
15     >   password: { ...
18     },
19     >   city: { ...
22     },
23   },
24   { timestamps: true }
25 );
26
27 //next is middleware
28 userModel.pre('save', async function (next) {
29   const salt = await bcrypt.genSalt(10);
30   this.password = await bcrypt.hash(this.password, salt);
31   next();
32 });
33 You, 5 minutes ago • Uncommitted changes
34 userModel.methods.isPasswordMatched = async function (enteredPassword) {
35   return await bcrypt.compare(enteredPassword, this.password);
36 };
37 export const User = mongoose.model('User', userModel);
38
```

```
JS userRoutes.js JS userController.js X JS userModel.js JS index.js .env
APP > controllers > JS userController.js > loginUser
98
99 const loginUser = async (req, res) => {
100   const { email, password } = req.body;
101   try {
102     const findUser = await User.findOne({ email: email });
103     // res.json(findUser);
104     if (findUser && (await findUser.isPasswordMatched(password))) {
105       res.json({ msg: 'Login Successful' });
106     } else {
107       res.json({ msg: 'Enter Valid Credentials' });
108     }
109   } catch (err) {
110     console.log(err);
111     res.json(err);
112   }
113 };
114 export {
115   createUser,
116   getUsers,
117   getUser,
118   deleteUser,
119   updateUser,
120   updatePassword,
121   loginUser,
122 };
123
```

Then install jsonweb token

npm i jsonwebtoken

import jwt

```
JS userRoutes.js JS userController.js X .env
APP > controllers > JS userController.js > loginUser > token
You, 6 minutes ago | 1 author (You)
1 import { User } from '../models/userModel.js';
2 import jwt from 'jsonwebtoken';
3
```

```
JS userRoutes.js  JS userController.js X  .env
APP > controllers > JS userController.js > loginUser > token
99
100 const loginUser = async (req, res) => {
101   try {
102     const { email, password } = req.body;
103     const findUser = await User.findOne({ email: email });
104     // res.json(findUser);
105     console.log(password);
106     if (findUser && (await findUser.isPasswordMatched(password))) {
107       // res.json({ msg: 'Login Successful' });
108       const token = jwt.sign({ id: findUser?._id }, process.env.JWT_SECRET, {
109         expiresIn: '1d',
110       });
111       res.json({ name: findUser?.name, email: findUser?.email, token: token });
112     } else {
113       res.json({ user: findUser, msg: 'Enter Valid Credentials' });
114     }
115   } catch (err) {
116     console.log(err);
117     res.json(err);
118   }
119 };
```

```
JS userRoutes.js  JS userController.js  .env X
APP > .env
You, 30 minutes ago | 1 author (You)
1  PORT = 4000
2  MONGODB_URL = mongodb://localhost:27017/lms1
3  JWT_SECRET = "mysecret"  You, 30 minutes ago
```

We have seen

Password hashing

Password matching

Generating json web token