Name: Hun Ravit

ID: e20180328

Exercise1: connect with mongooseDB

DB and model

connect.js U

ex1_api_mongo > db > connect.js > ...

```js
const mongoose = require('mongoose')
require("dotenv").config();

const uri= "mongodb://localhost:27017/test";
const option = {
  autoIndex:false,
  maxPoolsize:10,
  serverSelectionTimeoutMS:5000,
  socketTimeoutMS:45000,
  family:4
};

const connectDB = (url) => {
  return mongoose.connect(uri,option)
}

module.exports = connectDB;
```

user.js U

ex1_api_mongo > model > user.js > UserSchema

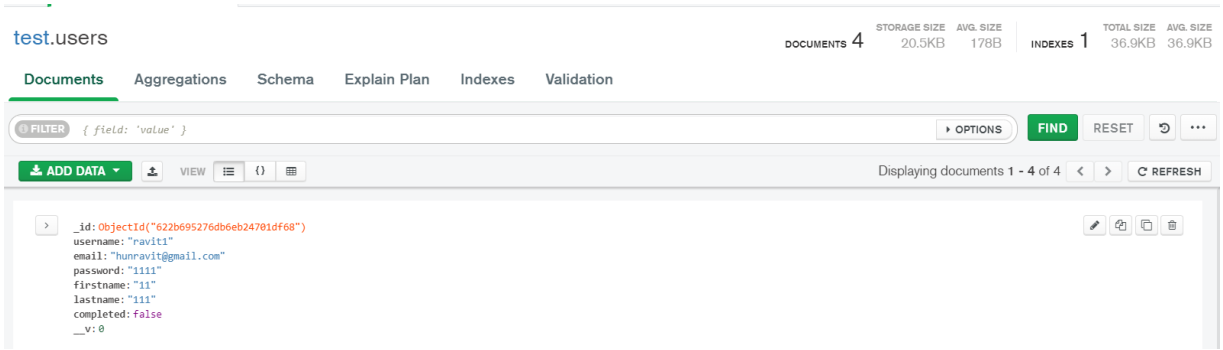
```js
const mongoose = require('mongoose')

const UserSchema = new mongoose.Schema({
  username: {
    type: String,
    required: [true, 'must provide username'],
    trim: true,
  },
  email: {
    type: String,
    required: [true, 'must provide email'],
  },
  password: {
    type: String,
    required: [true, 'must provide password'],
  },
  firstname: {
    type: String,
    required: [false, 'must provide firstname'],
    default:''
  },
  lastname: {
    type: String,
    required: [false, 'must provide lastname'],
    default:''
  },
  completed: {
    type: Boolean,
    default: false,
  },
  versionKey: false
})

module.exports = mongoose.model('User', UserSchema)
```

Controller

```js
const createUser = async (req, res) => {


  const username = req.body.username;
  try {
    const userExist = await User.findOne({ username: username})
    if(userExist){
        res.status(400).json({msg:'success',command:'Username is already exits'});
    }else{

      const user = new User({
        username: req.body.username,
        password:req.body.password,
        email:req.body.email,
        firstname:req.body.firstname,
        lastname:req.body.lastname
      });

      try {
        const newUser = await user.save();
        res.status(201).json({msg:'success', newUser });
      } catch (err) {
        res.status(500).json({msg:'error', message: err.message });
      }

    }

  } catch (err) {
    res.status(500).json({msg:'error', message: err.message });
  }



}
```

```
const login = async (req,res)=>{

  const { username,password } = req.body;
  var nameFind = username;

  try {
    const user = await User.findOne({ username: nameFind})
    if(user){
      var info = user;
      if(info.password==password){
        res.status(200).json({msg:'success',user});
      }else{
        res.status(400).json({msg:'success',command:'wrong password'});
      }
    }
  } catch (err) {
    res.status(500).json({ message: err.message });
  }

}
```

result:



Exercise2: Encrypt password when register

Controller

```
const createUser = async (req, res) => {

  const username = req.body.username;
  try {
    const userExist = await User.findOne({ username: username})
    if(userExist){
        res.status(400).json({msg:'success',command:'Username is already exits'});
    }else{
      //encryp password
      const hashPass = await bcrypt.hash(req.body.password, 12);

      const user = new User({
        username: req.body.username,
        password: hashPass,
        email:req.body.email,
        firstname:req.body.firstname,
        lastname:req.body.lastname
      });
      try {
        const newUser = await user.save();
        res.status(201).json({msg:'success', newUser });
      } catch (err) {
        res.status(500).json({msg:'error', message: err.message });
      }
    }
  } catch (err) {
    res.status(500).json({msg:'error', message: err.message });
  }

}
```

```
const login = async (req,res)=>{

  const { username,password } = req.body;
  var nameFind = username;

  try {
    const user = await User.findOne({ username: nameFind})
    if(user){
      var info = user;
      //bcrypt
      const checkPass = await bcrypt.compare(password, info.password)
      if(checkPass){
        res.status(200).json({msg:'success',user});
      }else{
        res.status(400).json({msg:'Error',command:'wrong password'});
      }
    }
  } catch (err) {
    res.status(500).json({ message: err.message });
  }



}
```

Result:


```
_id: ObjectId("622c15b43e9c51e57a8381ac")
username: "ravit27121"
email: "hunravit@gmail.com"
password: "$2a$12$gF4iUYHzrPPk5u4H6wPgney7yXg9OcDdS2hXtvHlXddKkoyzgECbe"
firstname: "11"
lastname: "111"
completed: false
__v: 0
```

```
_id: ObjectId("622c219ef8849c71cdb59d08")
username: "ravit27"
email: "hunravit@gmail.com"
password: "$2a$12$GeKFPJ3mmpLBkizOLMulOeonZ/iXj1i9eAvHzKc9JH7OLGRlJKAUS"
firstname: "11"
lastname: "111"
completed: false
__v: 0
```

Exercise3: Token

Controller:

```
const login = async (req,res)=>{

  const { username,password } = req.body;
  var nameFind = username;

  try {
    const user = await User.findOne({ username: nameFind})
    if(user){
      var info = user;
      //bcrypt
      const checkPass = await bcrypt.compare(password, info.password);
      if(checkPass){

        const token = await jwt.sign({username},process.env.SECRETE,{ expiresIn: '2h' });
        store.set(process.env.SECRETE,token);

        res.status(200).json({msg:'success',user});
      }else{
        res.status(400).json({msg:'Error',command:'wrong password'});
      }
    }
  } catch (err) {
    res.status(500).json({ message: err.message });
  }

}
```

Middleware:

ex3_api_token > middleware > auth.js > ...

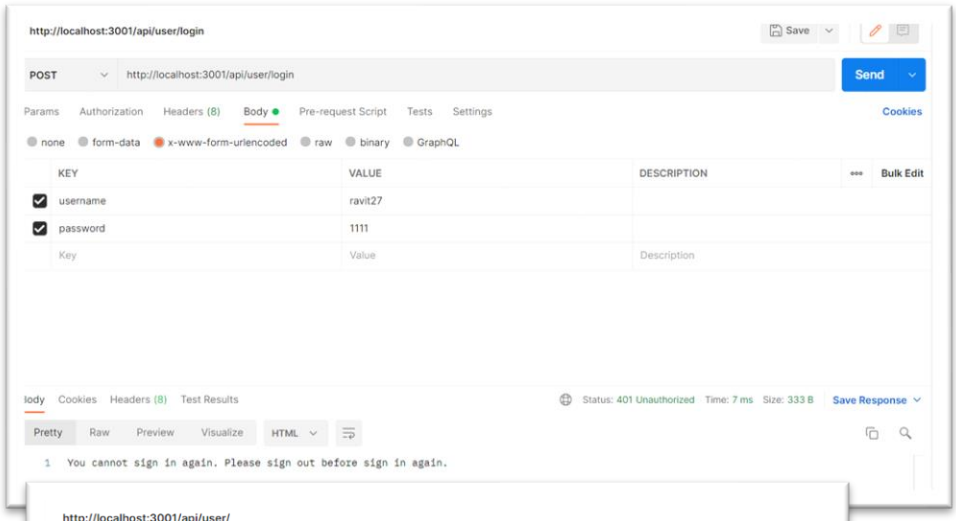
```
const jwt = require("jsonwebtoken");
const store = require("store2")
require("dotenv").config();

const verifyToken = (req, res,next) => {

  const token = store.get(process.env.SECRETE);
  // console.log(token);
  try {
    const decoded = jwt.verify(token, process.env.SECRETE);
    req.user = decoded;

  } catch (e) {}
  return next();
};
module.exports = verifyToken

```

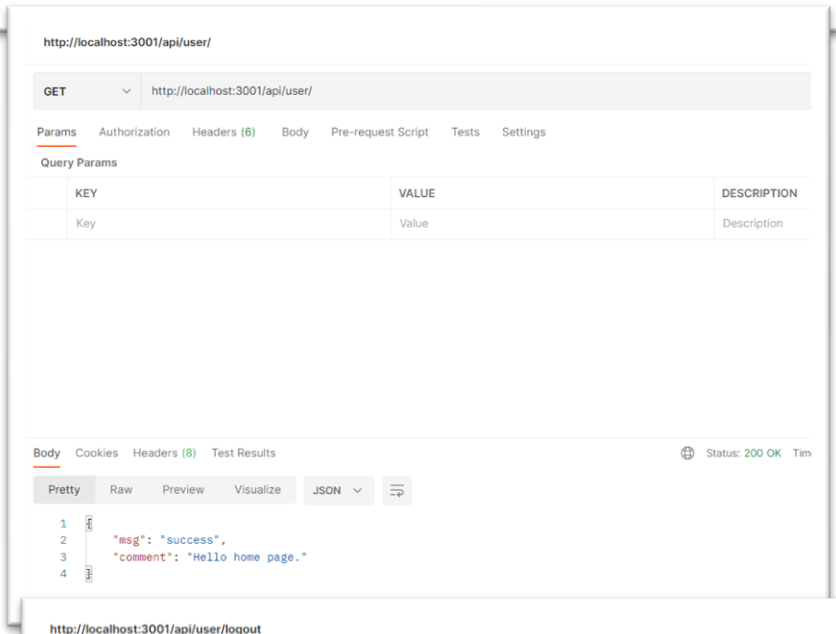
Route:

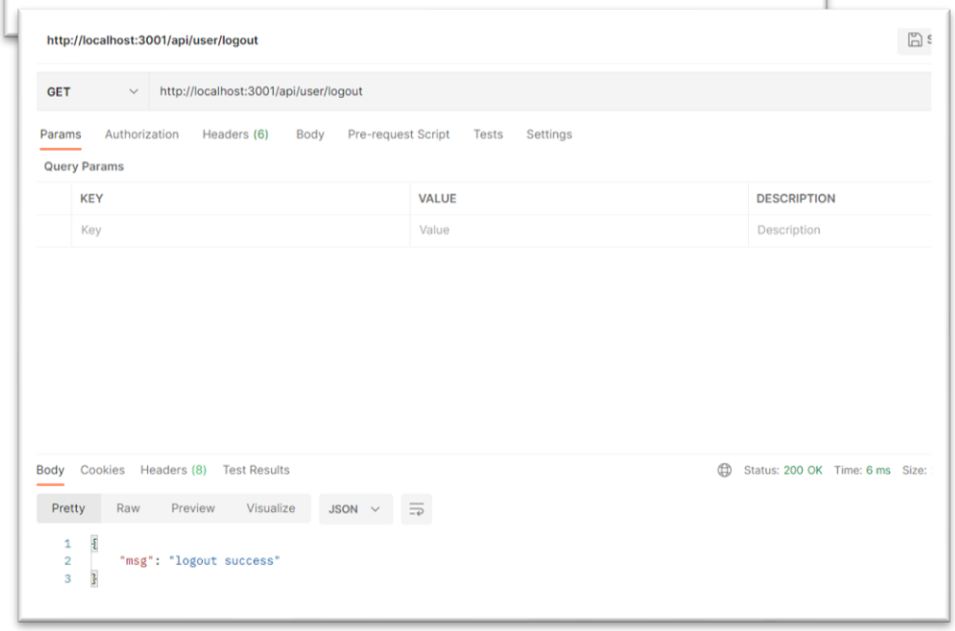
```
const checkLogout = (req,res,next)=>{
  if(req.user){
      //  console.log(req.user);
      return next();
  }else{
    return res.status(401).send("Attem to sign out fial.");
  }
}
const checkLogin = (req,res,next)=>{
    if(!req.user){
      return next();
    }else{
      return res.status(401).send("You cannot sign in again. Please sign out before sign in again.");
    }
}
const checkRegister = (req,res,next)=>{
  if(!req.user){
    return next();
  }else{
    return res.status(401).send("You cannot register a user.");
  }
}
const checkHomepage= (req,res,next)=>{
  if(!req.user){
    return next();
  }else{
    return res.status(401).send("You cannot go to home page. PLease log in.");
  }
}

```

```
router.post('/login',checkAuth,checkLogin,login);
router.post('/register',checkAuth,checkRegister,createUser);
router.get('/logout',checkAuth,checkLogout, (req, res, next) => {
  store.clear();
  res.status(200).json({msg:'logout success'})
});
router.get('/',checkAuth,checkHomepage, (req, res, next) => {
  res.status(200).json({msg:'success',comment:'Hello home page.'})
});
```

Result:


http://localhost:3001/api/user/login
Save
POST
http://localhost:3001/api/user/login
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
username
ravit27
password
1111
Key
Value
Description
Body
Cookies
Headers (8)
Test Results
Status: 401 Unauthorized
Time: 7 ms
Size: 333 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 You cannot sign in again. Please sign out before sign in again.


http://localhost:3001/api/user/
GET
http://localhost:3001/api/user/
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Key
Value
Description
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "msg": "success",
3 "comment": "Hello home page."
4 }


http://localhost:3001/api/user/logout
GET
http://localhost:3001/api/user/logout
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Key
Value
Description
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Time: 6 ms
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "msg": "logout success"
3 }