**Module 5: Assignment 1: Using Node middleware ORM modules and Middleware Security Modules to implement security**

Saurabh Verma

Arizona State University

IFT 598: Middleware Programming & Database Security

Professor. Dinesh Sthapit

October 27, 2022

JWT flowchart:

A picture containing table

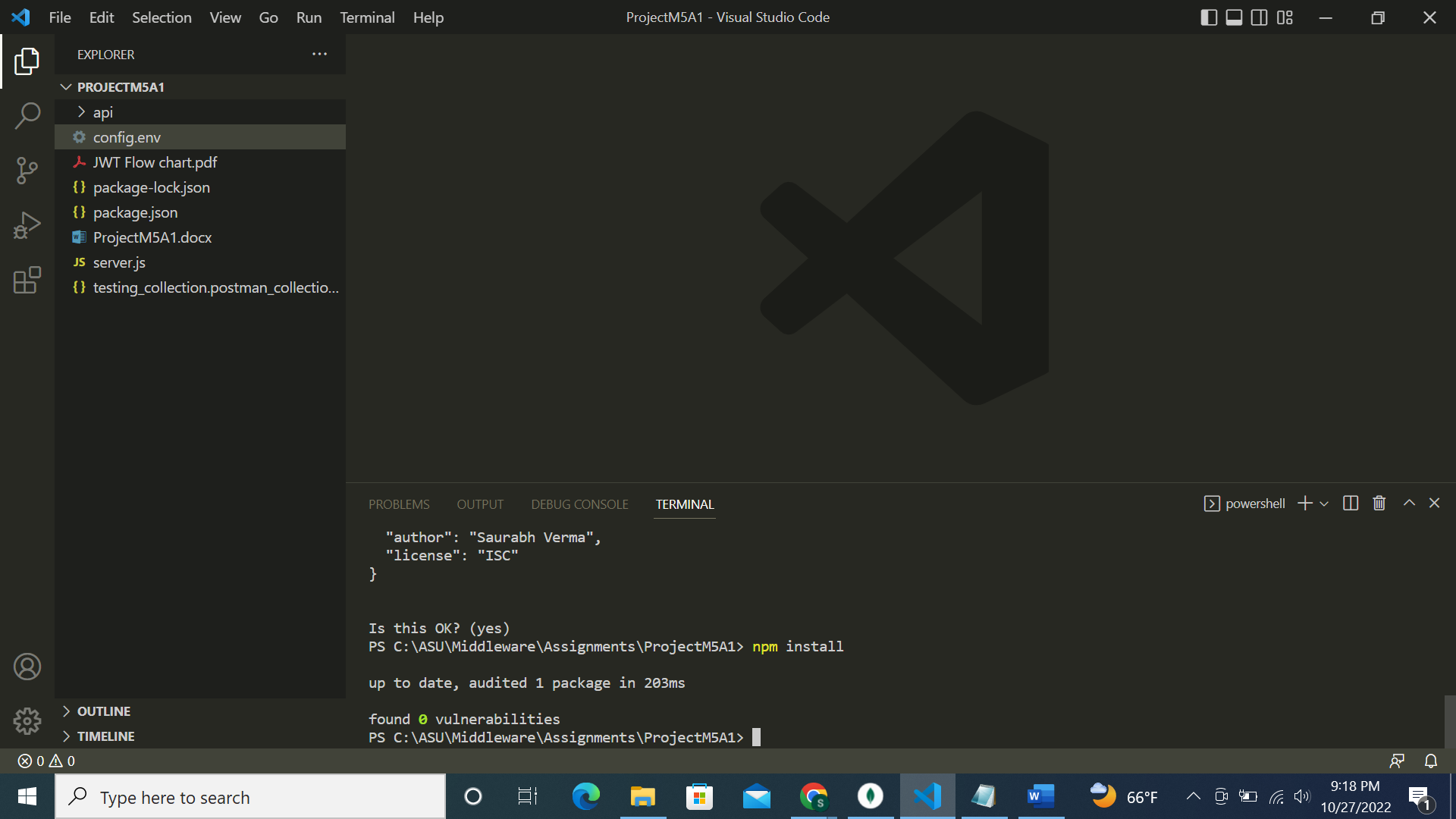
Description automatically generated

Running npm init command:

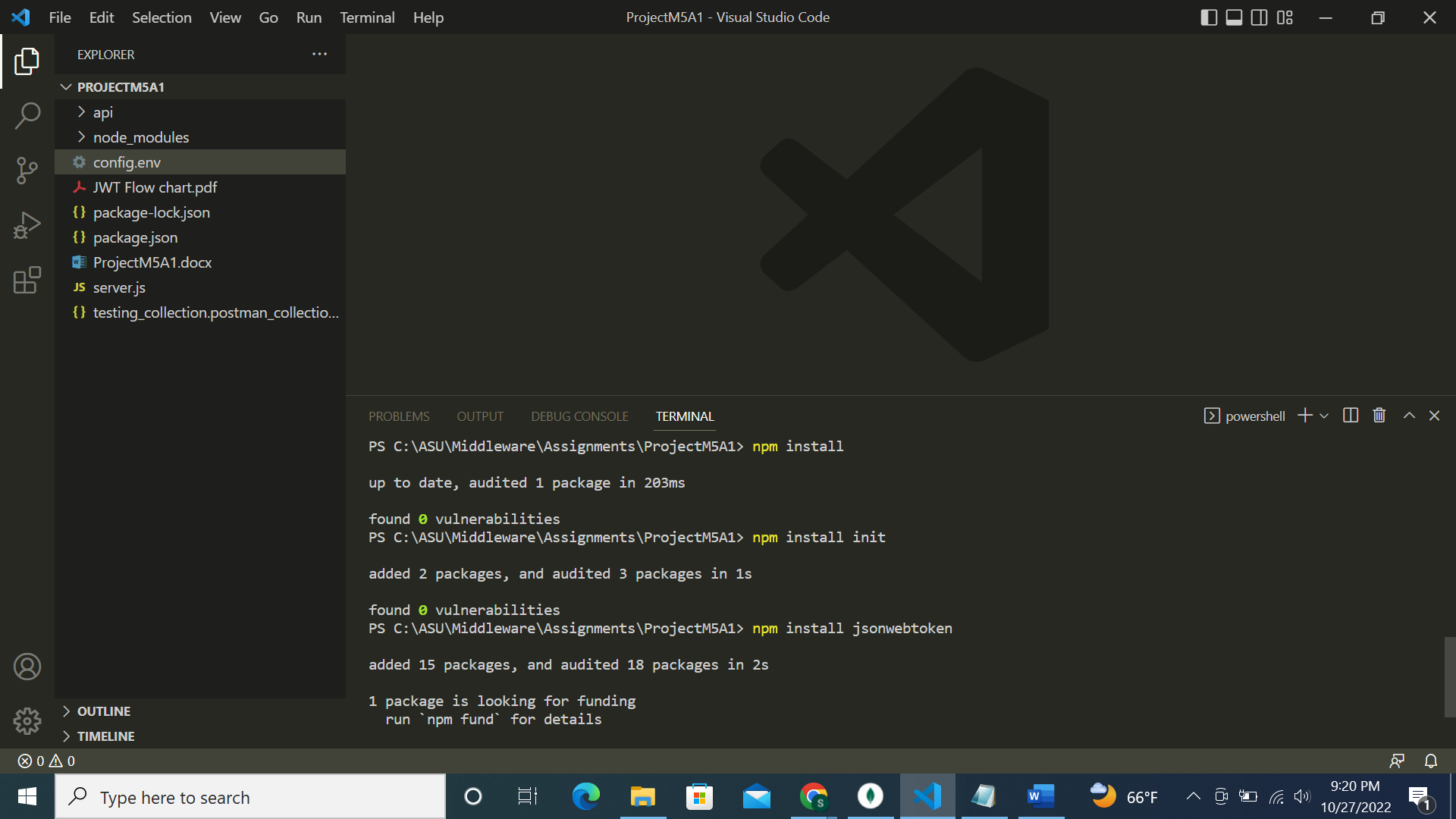
Text

Description automatically generated

Running npm install command:



Step 1: Running npm install jsonwebtoken command:



Step 2: Create user model

Code in the userModel.js:

**Code:**

'use strict';

var mongoose = require('mongoose'),

    bcrypt = require('bcrypt'),

    Schema = mongoose.Schema;

var UserSchema = new Schema({

    fullName: {

        type: String,

        trim: true,

        required: true

    },

    email: {

        type: String,

        unique: true,

        lowercase: true,

        trim: true,

        required: true

    },

    hash\_password: {

        type: String,

        required: true

    },

    created: {

        type: Date,

        default: Date.now

    }

});

UserSchema.methods.comparePassword = function(password){

    return bcrypt.compareSync(password, this.hash\_password);

};

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

A screenshot of a computer

Description automatically generated

Creating the todoModel.js:

**Code:**

'use strict';

var mongoose = require('mongoose'),

    Schema = mongoose.Schema;

var ToDoSchema = new Schema({

    name: {

        type: String,

        required: 'Kindly enter name of the task'

    },

    createdDate: {

        type: Date,

        default: Date.now

    },

    status: {

        type: String,

        default: 'pending'

    }

});

module.exports = mongoose.model('Tasks', ToDoSchema);

A screenshot of a computer

Description automatically generated

Step 3: Create user handlers

Coding in userController.js file:

**Code:**

'use strict';

var mongoose = require('mongoose'),

    jwt = require('jsonwebtoken'),

    bcrypt = require('bcrypt'),

    User = mongoose.model('User');

exports.register = function(req, res){

    var newUser = new User(req.body);

    newUser.hash\_password = bcrypt.hashSync(req.body.password, 10);

    newUser.save(function(err, user) {

        if (err) {

        return res.status(400).send({

            message: err

        });

        } else {

        user.hash\_password = undefined;

        return res.json(user);

        }

    });

};

exports.sign\_in = function(req, res){

    User.findOne({

        email: req.body.email

      }, function(err, user) {

        if (err) throw err;

        if (!user) {

          res.status(401).json({ message: 'Authentication failed. User not found.' });

        } else if (user) {

          if (!user.comparePassword(req.body.password)) {

            res.status(401).json({ message: 'Authentication failed. Wrong password.' });

          } else {

            return res.json({token: jwt.sign({ email: user.email, fullName: user.fullName, \_id: user.\_id}, 'RESTFULAPIs')});

          }

        }

    });

};

exports.loginRequired = function(req, res, next){

    if (req.user) {

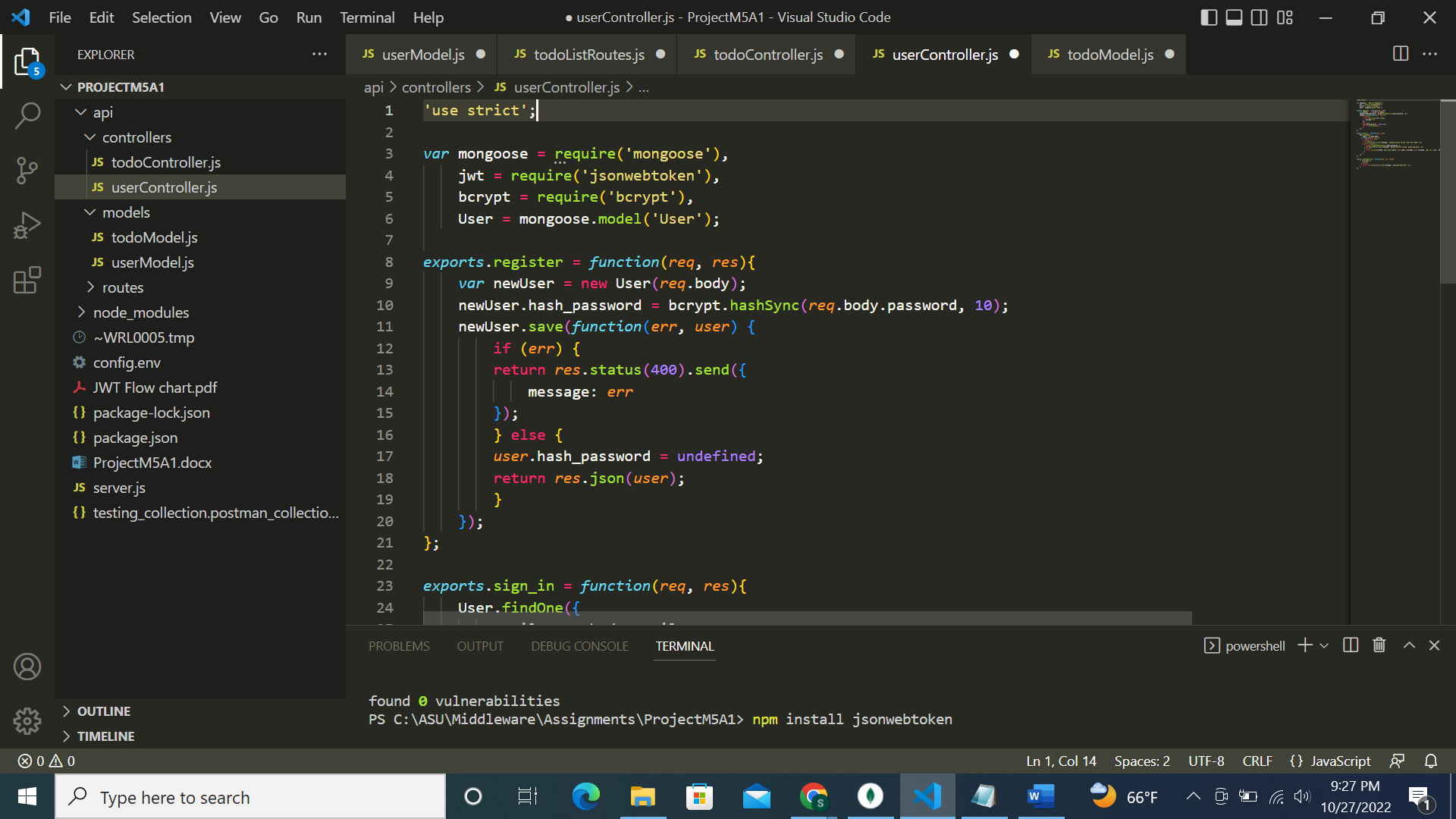
        next();

      } else {

        return res.status(401).json({ message: 'Unauthorized user!' });

    }

};



Coding in todoController.js file:

**Code:**

'use strict';

var mongoose = require('mongoose'),

    Task = mongoose.model('Tasks');

exports.getAlltask = function(req, res){

    Task.find({}, function(err, task) {

        if (err)

            res.send(err);

        res.json(task);

    });

};

exports.createTask = function(req, res){

    var new\_task = new Task(req.body);

    new\_task.save(function(err, task) {

    if (err)

        res.send(err);

    res.json(task);

    });

};

exports.getTask = function(req, res){

    Task.findById(req.params.taskId, function(err, task) {

    if (err)

        res.send(err);

    res.json(task);

    });

};

exports.updateTask = function(req, res) {

    Task.findOneAndUpdate({\_id:req.params.taskId}, req.body, {new: true}, function(err, task) {

        if (err)

            res.send(err);

        res.json(task);

    });

};

exports.deleteTask = function(req, res) {

    Task.remove({\_id: req.params.taskId}, function(err, task) {

            if (err)

                res.send(err);

            res.json({message: 'Task successfully deleted'});

        });

};

A screenshot of a computer

Description automatically generated

Step 4:

Adding todoListRoutes.js:

**Code:**

'use strict';

module.exports = function(app) {

    var todoList = require('../controllers/todoController'),

        userHandlers = require('../controllers/userController');

    // todo routes

    app.route('/tasks')

        .get(todoList.getAlltask)

        .post(userHandlers.loginRequired, todoList.createTask);

    app.route('/tasks/:taskId')

        .get(todoList.getTask)

        .put(todoList.updateTask)

        .delete(todoList.deleteTask);

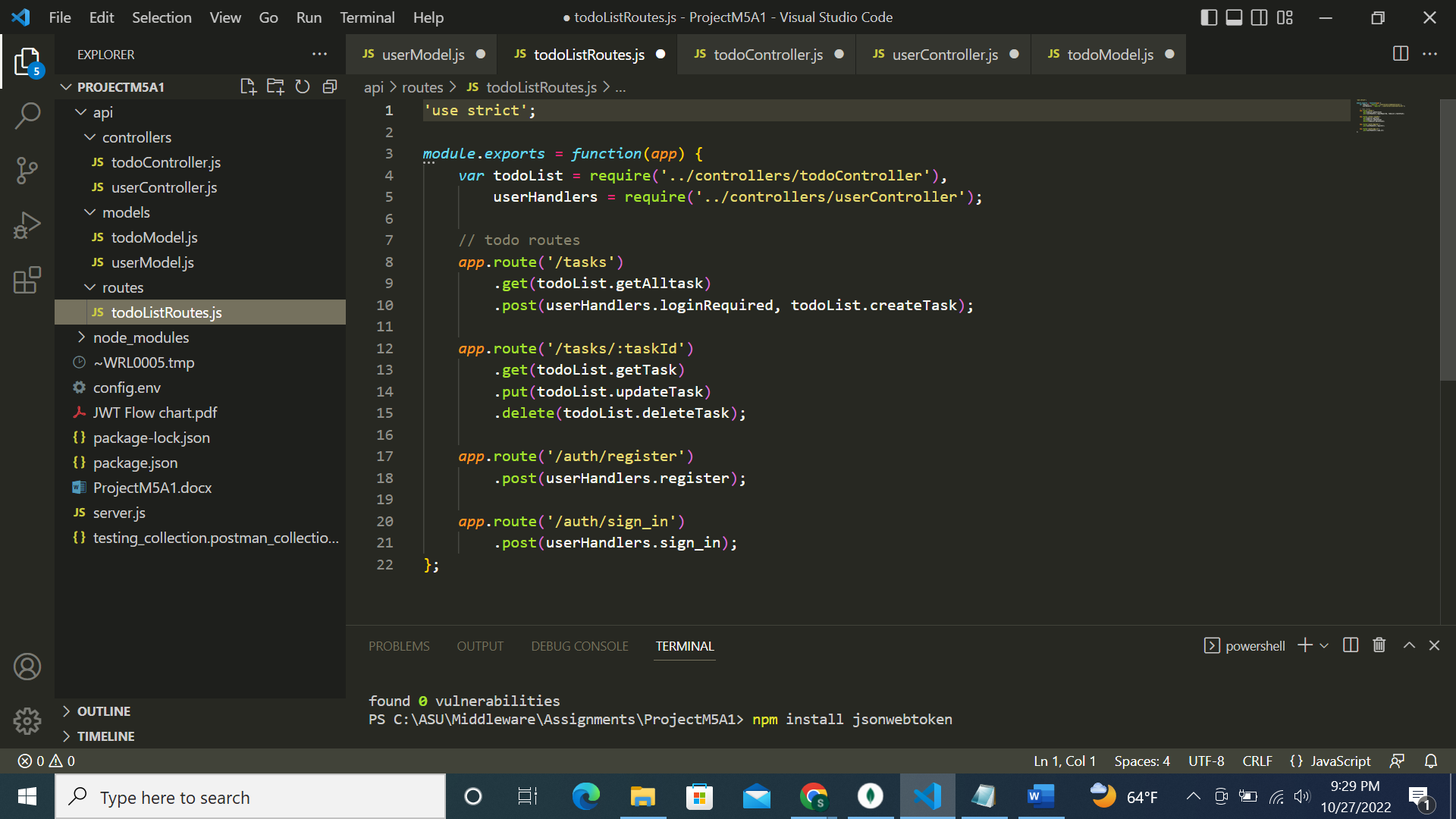
    app.route('/auth/register')

        .post(userHandlers.register);

    app.route('/auth/sign\_in')

        .post(userHandlers.sign\_in);

};



Creating server.js file:

**Code:**

const dotenv = require('dotenv');

dotenv.config({ path: './config.env' });

//listen port 3000

var express = require('express'),

  app = express(),

  port = process.env.PORT || 3000,

  mongoose = require('mongoose'),

  Task = require('./api/models/todoModel'),

  User = require('./api/models/userModel'),

  bodyParser = require('body-parser'),

  jsonwebtoken = require("jsonwebtoken");

mongoose.Promise = global.Promise;

// mongoose.connect('mongodb://localhost/Tododb');

const MONGO\_DATA\_BASE = process.env.DATABASE.replace('<password>', process.env.DB\_PASSWORD);

// Connect to the database

mongoose.connect(MONGO\_DATA\_BASE,

  //connection recipie

  {

    useNewUrlParser: true,

    useUnifiedTopology: true

    //useCreateIndex: true

  }).then(con=>{

    console.log(con.connection);// log connection properties

    console.log(`The Database connection was successful with ${process.env.DATABASE}`);// log connection properties

  });

//bodyParser

app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json());

app.use(function(req, res, next) {

  if (req.headers && req.headers.authorization && req.headers.authorization.split(' ')[0] === 'JWT') {

    jsonwebtoken.verify(req.headers.authorization.split(' ')[1], 'RESTFULAPIs', function(err, decode) {

      if (err) req.user = undefined;

      req.user = decode;

      next();

    });

  } else {

    req.user = undefined;

    next();

  }

});

//routes

var routes = require('./api/routes/todoListRoutes');

routes(app);

app.use(function(req, res) {

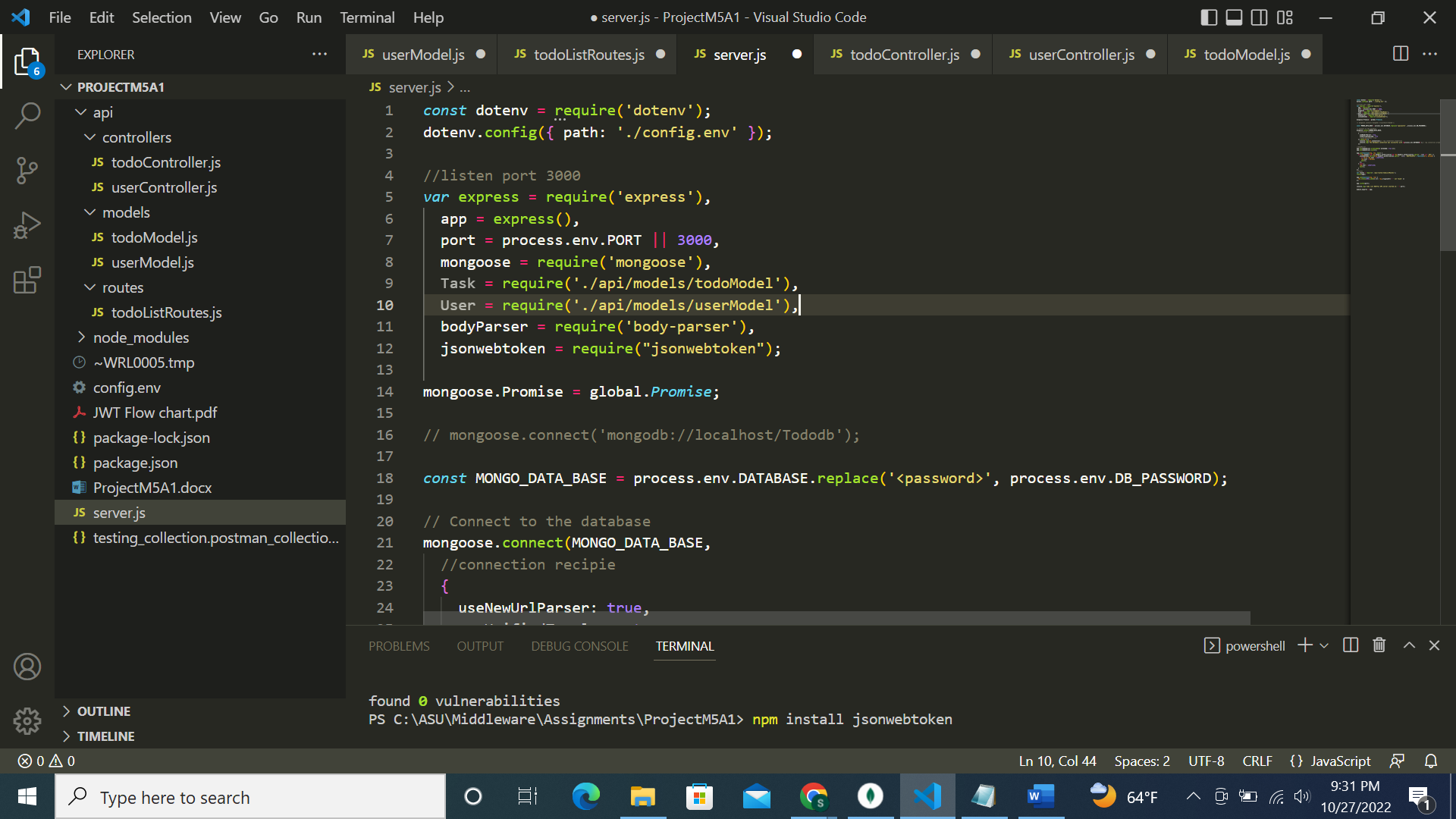
  res.status(404).send({ url: req.originalUrl + ' not found' })

});

app.listen(port);

console.log('todo list RESTful API server started on: ' + port);

module.exports = app;



Configuring config.env file:

**Code:**

NODE\_ENV=development

PORT=3000

DATABASE=mongodb+srv://sverma67:<password>@cluster0.qekugi2.mongodb.net/test

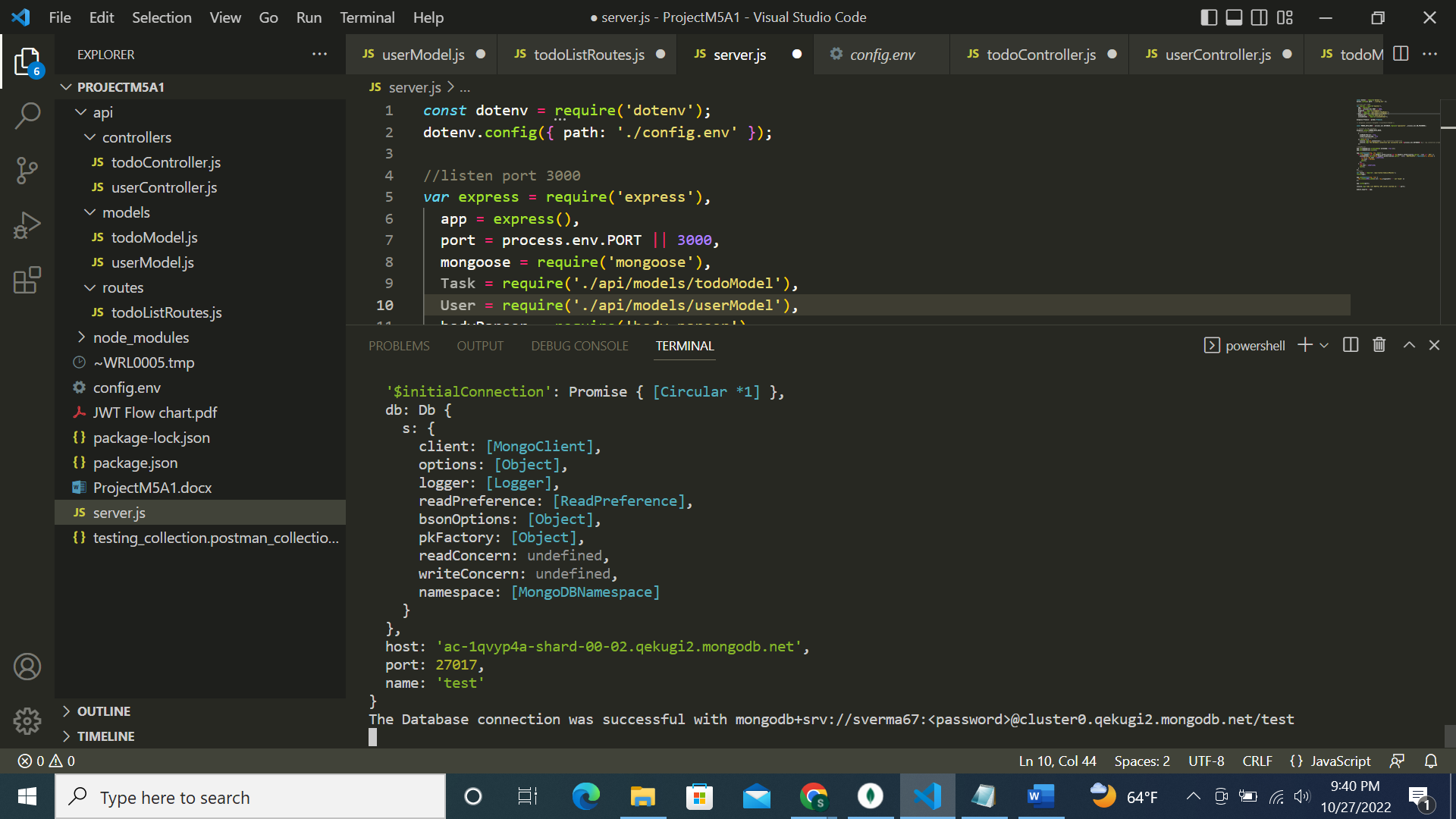
DB\_PASSWORD=MongoDBsdv200!

A screenshot of a computer

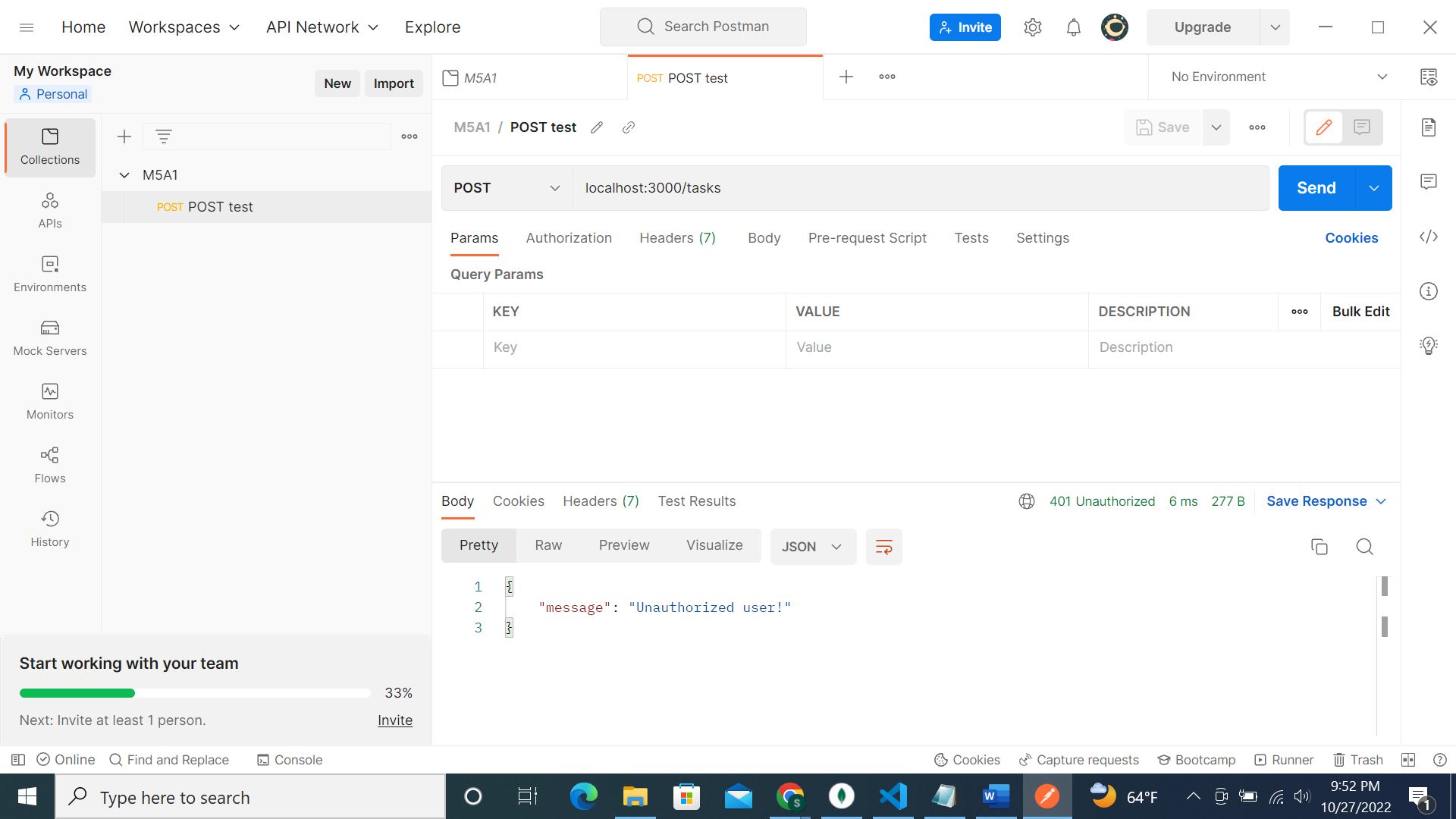
Description automatically generated

Step 5: Testing:

Running server.js:



Testing POST:



Creating a new user:

Graphical user interface, text, application, email

Description automatically generated

Checking MongoDB:

Graphical user interface, text, email

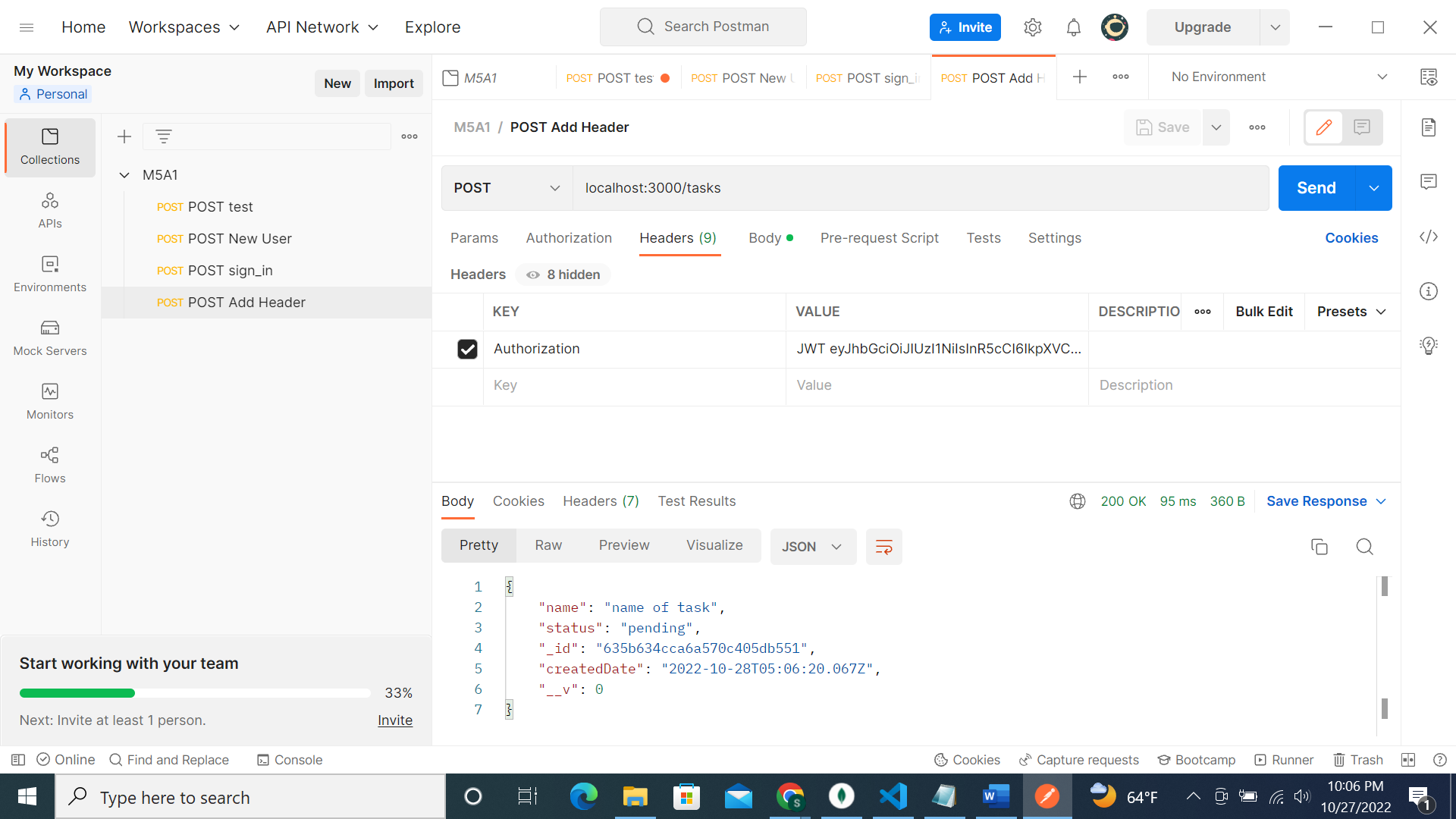
Description automatically generated

Signing in:

A screenshot of a computer

Description automatically generated

Testing the JWT token:



A screenshot of a computer

Description automatically generated

Task created in MongoDB:

Graphical user interface, text, email

Description automatically generated

Postman requests:

Text

Description automatically generated

**References**

[Authenticating Node.js with JWT](https://www.codementor.io/@olatundegaruba/5-steps-to-authenticating-node-js-with-jwt-7ahb5dmyr)

[JWT Flow chart](https://lucid.app/lucidchart/ba93ba2b-16cc-4a13-83aa-e2b18a2a028b/edit?beaconFlowId=3161EFF4A4FB8AB2&invitationId=inv_b0e6b702-3128-4260-9836-5c7fb1bd352e&page=0_0)