**IFT 598: IFT 458: Middleware Prog & Database Sec (2022 Fall)**

**Assignment**

**Name: ABHISHEK MATHUR**

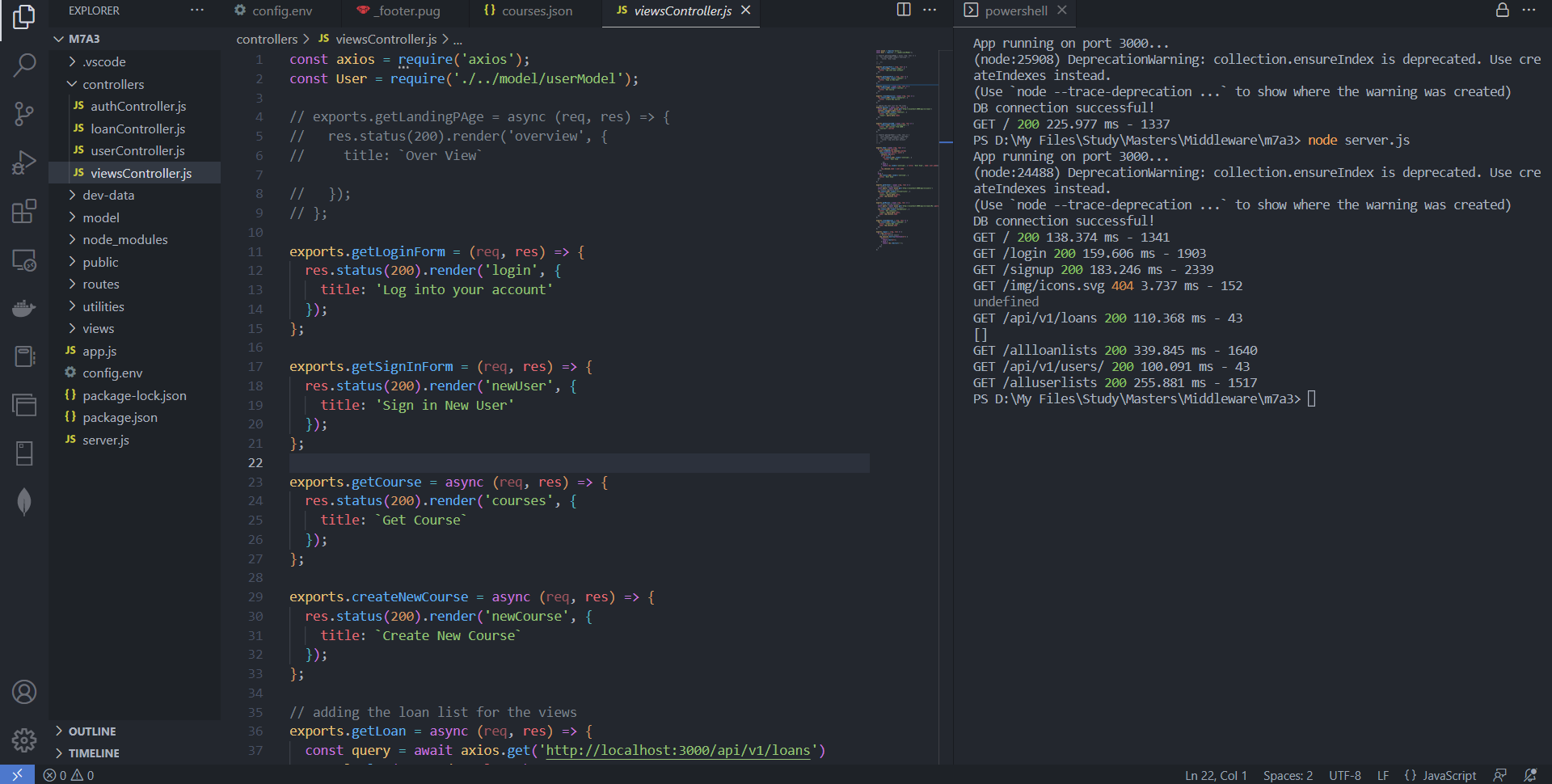
**Professor: Dinesh Sthapit**

Replicated the module 7 assignment 2 code for this assignment and downloaded the dependencies

**Text

Description automatically generated**

**Step 1: Sign In a new user**

****

**Code:**

const axios = require('axios');

const User = require('./../model/userModel');

// exports.getLandingPAge = async (req, res) => {

//   res.status(200).render('overview', {

//     title: `Over View`

//   });

// };

exports.getLoginForm = (req, res) => {

  res.status(200).render('login', {

    title: 'Log into your account'

  });

};

exports.getSignInForm = (req, res) => {

  res.status(200).render('newUser', {

    title: 'Sign in New User'

  });

};

exports.getCourse = async (req, res) => {

  res.status(200).render('courses', {

    title: `Get Course`

  });

};

exports.createNewCourse = async (req, res) => {

  res.status(200).render('newCourse', {

    title: `Create New Course`

  });

};

// adding the loan list for the views

exports.getLoan = async (req, res) => {

  const query = await axios.get('http://localhost:3000/api/v1/loans')

  console.log(query.data.loans)

  res.status(200).render('loanlist', {

    title: `Get loanlists`,

    "loans": query.data.loans

  });

};

exports.getCoursesJSON = async (req, res) => {

  res.status(200).render('allCourses', {

    title: `Get courses from JSON`,

    "courses": courses

  });

};

// exports.getLoginUser = (req, res) => {

//   const {email, password} = req.body;

//   res.status(200).render('login', {

//     title: 'Log into your account'

//   });

// };

exports.home = async (req, res) => {

  if (req.session.userId){

    User.findById(req.session.userId)

    .exec(function (error, user) {

      console.log(user)

      if (error) {

        res.status(200).render('overview', {

          title: `Over View`

        });

      } else {

        return res.render('overview', { title: 'Over View', name: user.name});

      }

      req.session.name = user.name

    });

  }

  else {

    res.status(200).render('overview', {

    title: `Over View`

  });

  }

};

exports.getAllUser = async (req, res) => {

  // console.log(req.session)

  const query = await axios.get('http://localhost:3000/api/v1/users/')

  // console.log(query.data)

  res.status(200).render('alluserlists', {

    title: `Get userlists`,

    "users": query.data.users,

    name: req.session.name

  });

};

exports.getMyLoan = async (req, res) => {

  // console.log(req.session)

  const query = await axios.get('http://localhost:3000/api/v1/loans/My',{params:{name:req.session.name}})

  // console.log(query.data.loans)

  res.status(200).render('myloanlists', {

    title: `Get loanlists`,

    "loans": query.data.loans,

    name: req.session.name

  });

};

exports.CreateNewLoan = (req, res) => {

  res.status(200).render('newLoan', {

    title: 'Create New Loan',

    name: req.session.name

  });

};

exports.logout = (req, res) => {

  if (req.session) {

    // delete session object

    req.session.destroy(function(err) {

      if(err) {

        return next(err);

      } else {

        return res.redirect('/');

      }

    });

  }

};

Pug file for signing up the user

**Text

Description automatically generated**

**Code:**

extends baseTemplate

block content

  main.main

    .card-container

      .card

        .card\_\_header

          h3.heading-tertirary

            span Sign In

        .card\_\_details

          h4.card\_\_sub-heading New Student

          p.card\_\_text

            | Please enter the following Details

          .card\_\_data

            svg.card\_\_icon

              use(xlink:href='img/icons.svg#icon-map-pin')

            form(action='/api/v1/users/signup' method='post')

              label(for='name') Full name:

              input#name(type='text' name='name')

              br

              br

              label(for='email') Email:

              input#email(type='email' name='email')

              br

              br

              label(for='role') Role:

              input#role(type='text' name='role')

              br

              br

              label(for='password') Password:

              input#password(type='password' name='password')

              br

              br

              label(for='passwordConfirm') Confirm Password:

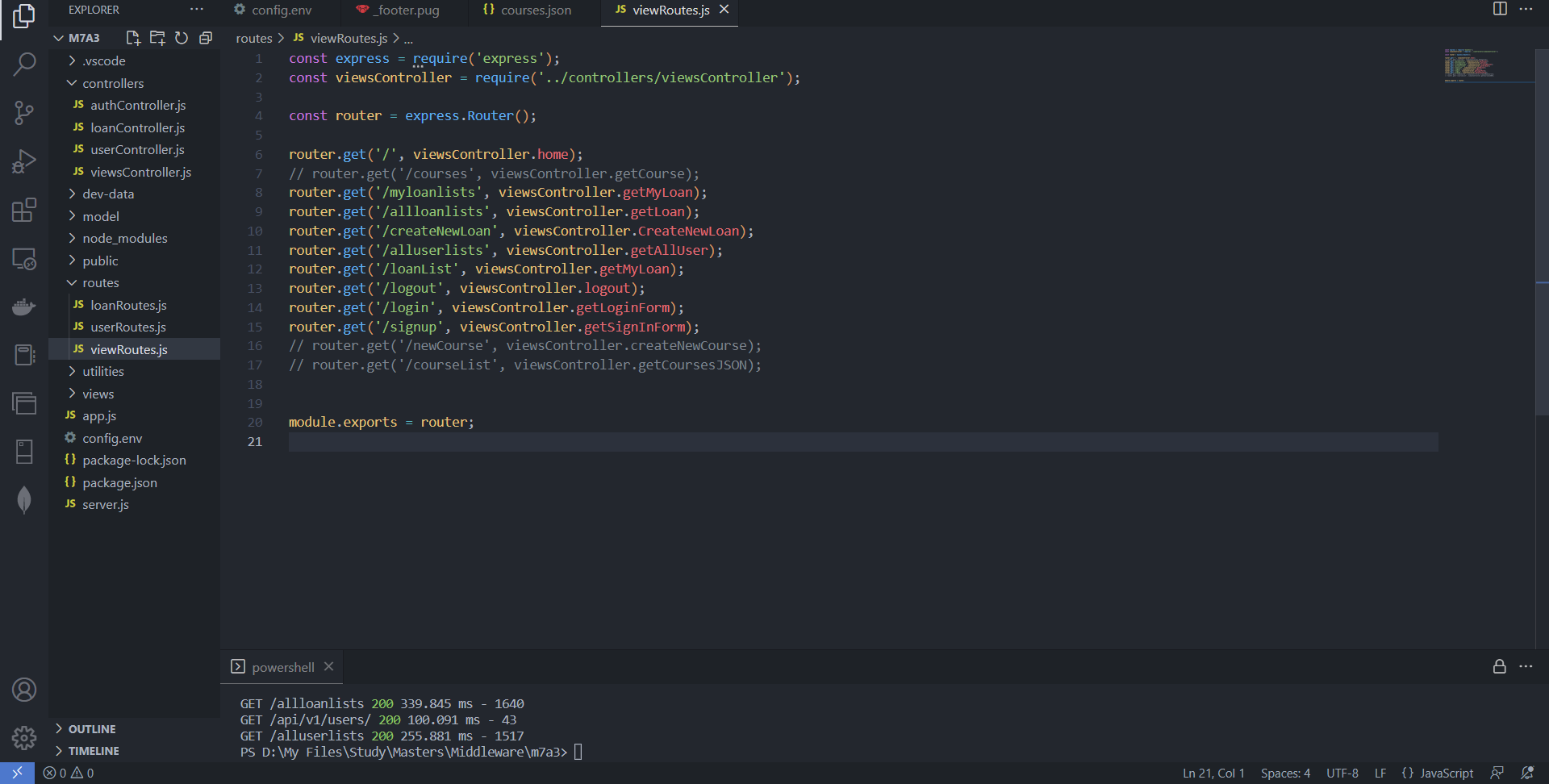
              input#passwordConfirm(type='password' name='passwordConfirm')

              br

              br

              input.card\_\_sub-heading(type='submit' value='Submit')

viewRoutes.js calling the view controller



Code:

const express = require('express');

const viewsController = require('../controllers/viewsController');

const router = express.Router();

router.get('/', viewsController.home);

// router.get('/courses', viewsController.getCourse);

router.get('/myloanlists', viewsController.getMyLoan);

router.get('/allloanlists', viewsController.getLoan);

router.get('/createNewLoan', viewsController.CreateNewLoan);

router.get('/alluserlists', viewsController.getAllUser);

router.get('/loanList', viewsController.getMyLoan);

router.get('/logout', viewsController.logout);

router.get('/login', viewsController.getLoginForm);

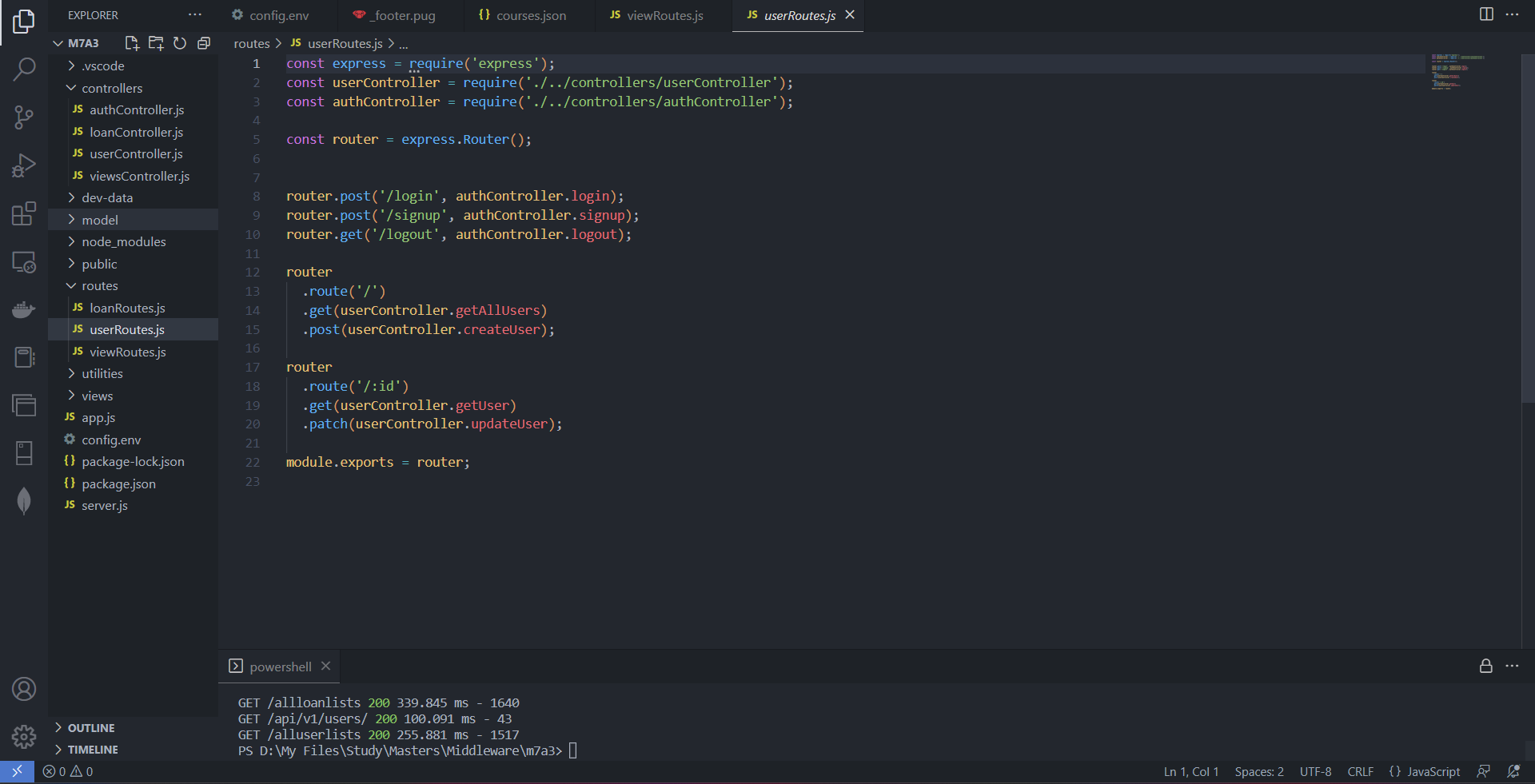
router.get('/signup', viewsController.getSignInForm);

// router.get('/newCourse', viewsController.createNewCourse);

// router.get('/courseList', viewsController.getCoursesJSON);

module.exports = router;

userRoutes.js calling the auth controller to sign up new user



Code:

const express = require('express');

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

const authController = require('./../controllers/authController');

const router = express.Router();

router.post('/login', authController.login);

router.post('/signup', authController.signup);

router.get('/logout', authController.logout);

router

  .route('/')

  .get(userController.getAllUsers)

  .post(userController.createUser);

router

  .route('/:id')

  .get(userController.getUser)

  .patch(userController.updateUser);

module.exports = router;

Authcontroller.js creating a new user

A screenshot of a computer

Description automatically generated with medium confidence

Code:

const crypto = require('crypto');

const jwt = require('jsonwebtoken');

const User = require('./../model/userModel');

const AppError = require('./../utilities/appError');

const util = require('util');

const express = require('express');

const alert =require('alert')

require('util.promisify').shim();

const signToken = id => {

  return jwt.sign({ id }, process.env.JWT\_SECRET, {

    expiresIn: process.env.JWT\_EXPIRES\_IN

  });

};

const createSendToken = (user, statusCode, res) => {

  const token = signToken(user.\_id);

  const cookieOptions = {

    expires: new Date(

      Date.now() + process.env.JWT\_COOKIE\_EXPIRES\_IN \* 24 \* 60 \* 60 \* 1000

    ),

    httpOnly: true

  };

  if (process.env.NODE\_ENV === 'production') {

    cookieOptions.secure = true;

  }

  res.cookie('jwt', token, cookieOptions);

  // Remove password from output

  user.password = undefined;

};

exports.signup = async (req, res, next) => {

  console.log(req.body)

  const newUser = await User.create({

    name: req.body.name,

    email: req.body.email,

    password: req.body.password,

    passwordConfirm: req.body.passwordConfirm

  });

  createSendToken(newUser, 201, res);

  console.log('sign up successfully')

  req.session.email = newUser.email

  req.session.name = newUser.name

  req.session.userId = newUser.\_id;

  res.redirect('/')

};

exports.login = async (req, res, next) => {

  const { email, password } = req.body;

  // 1) Check if email and password exist

  if (!email || !password) {

    return next(new AppError('Please provide email and password!', 400));

  }

  //authentication

  // 2) Check if user exists && password is correct

  const user = await User.findOne({ email }).select('+password');

  if (!user || !(user.correctPassword(password, user.password))) {

    return next(new AppError('Incorrect email or password', 401));

  }

  //end of authentication

  // 3) If everything ok, send token to client

  createSendToken(user, 200, res);

  req.session.email = user.email

  req.session.name = user.name

  req.session.userId = user.\_id;

  alert("successfully!!!")

  res.redirect('/')

};

exports.logout = (req, res) => {

  res.cookie('jwt', 'loggedout', {

    expires: new Date(Date.now() + 10 \* 1000),

    httpOnly: true

  });

  res.status(200).json({ status: 'success' });

};

exports.protect = async (req, res, next) => {

  // 1) Getting token and check of it's there

  let token;

  if (

    req.headers.authorization &&

    req.headers.authorization.startsWith('Bearer')

  ) {

    token = req.headers.authorization.split(' ')[1];

  } else if (req.cookies.jwt) {

    token = req.cookies.jwt;

  }

  if (!token) {

    return next(

      new AppError('You are not logged in! Please log in to get access.', 401)

    );

  }

  // 2) Verification token

  const decoded = await util.promisify(jwt.verify)(token, process.env.JWT\_SECRET);

  // 3) Check if user still exists

  const currentUser = await User.findById(decoded.id);

  if (!currentUser) {

    return next(

      new AppError(

        'The user belonging to this token does no longer exist.',

        401

      )

    );

  }

  // 4) Check if user changed password after the token was issued

  if (currentUser.changedPasswordAfter(decoded.iat)) {

    return next(

      new AppError('User recently changed password! Please log in again.', 401)

    );

  }

  // GRANT ACCESS TO PROTECTED ROUTE

  req.user = currentUser;

  res.locals.user = currentUser;

  next();

  res.status(201).json({

    status: 'success',

    token,

    data: {

      currentUser

    }

  });

};

// Only for rendered pages, no errors!

exports.isLoggedIn = async (req, res, next) => {

  if (req.cookies.jwt) {

    try {

      // 1) verify token

      const decoded = await promisify(jwt.verify)(

        req.cookies.jwt,

        process.env.JWT\_SECRET

      );

      // 2) Check if user still exists

      const currentUser = await User.findById(decoded.id);

      if (!currentUser) {

        return next();

      }

      // 3) Check if user changed password after the token was issued

      if (currentUser.changedPasswordAfter(decoded.iat)) {

        return next();

      }

      // THERE IS A LOGGED IN USER

      res.locals.user = currentUser;

      return next();

    } catch (err) {

      return next();

    }

  }

  next();

};

Usercontroller.js creating a new user

Text

Description automatically generated

Code:

const User = require('./../model/userModel');

exports.getAllUsers = async (req, res, next) => {

  const users = await User.find();

  // SEND RESPONSE

  res.status(200).json({

    status: 'success',

    results: users.length,

    users

  });

};

exports.createUser = async (req, res) => {

  const newUser = await User.create({

    name: req.body.name,

    email: req.body.email,

    password: req.body.password,

    passwordConfirm: req.body.password // this needs to be changed

  });

  res.status(201).json({

    status: 'success',

    data: newUser

  });

};

exports.deleteMe = async (req, res) => {

  await User.findByIdAndUpdate(req.user.id, { active: false });

  res.status(204).json({

    status: 'success',

    data: null

  });

};

exports.getUser = (req, res) => {

  res.status(500).json({

    status: 'error',

    message: 'This route is not yet defined!'

  });

};

exports.updateUser = (req, res) => {

  res.status(500).json({

    status: 'error',

    message: 'This route is not yet defined!'

  });

};

exports.deleteUser = (req, res) => {

  res.status(500).json({

    status: 'error',

    message: 'This route is not yet defined!'

  });

};

shows the code executed successfully without any errors.

Text

Description automatically generated

shows the MongoDB database created successfully

Graphical user interface, text, application

Description automatically generated

shows the overview page obtained when I hit localhost:3000/

Graphical user interface, application, Teams

Description automatically generated

shows the sign up page for new user.

Graphical user interface

Description automatically generated

shows that after I submit the information in sign up page the new user gets logged in with their name on the top right side of the screen.

Graphical user interface, application, Teams

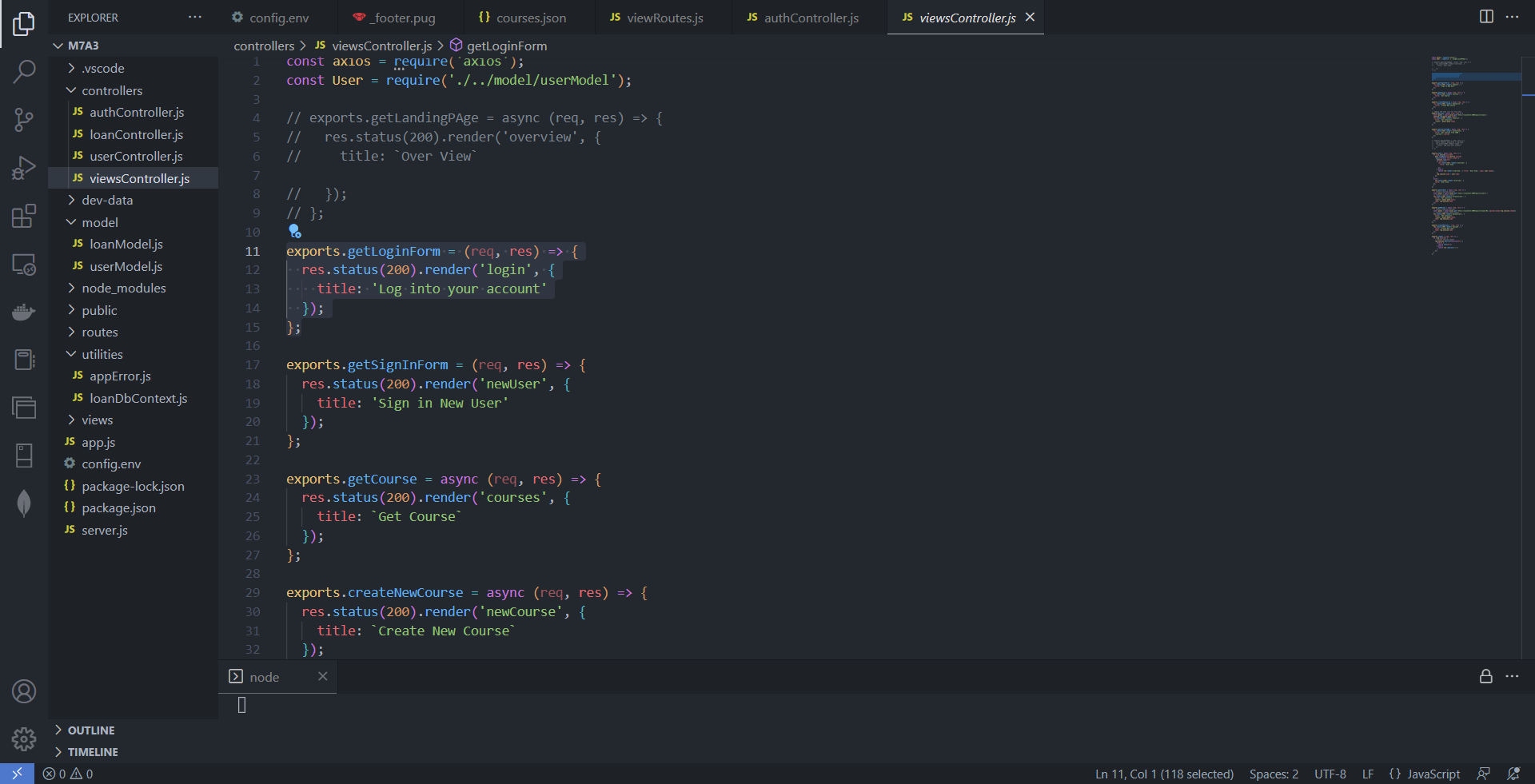
Description automatically generated

New user added into MongoDB successfully Graphical user interface, text, application

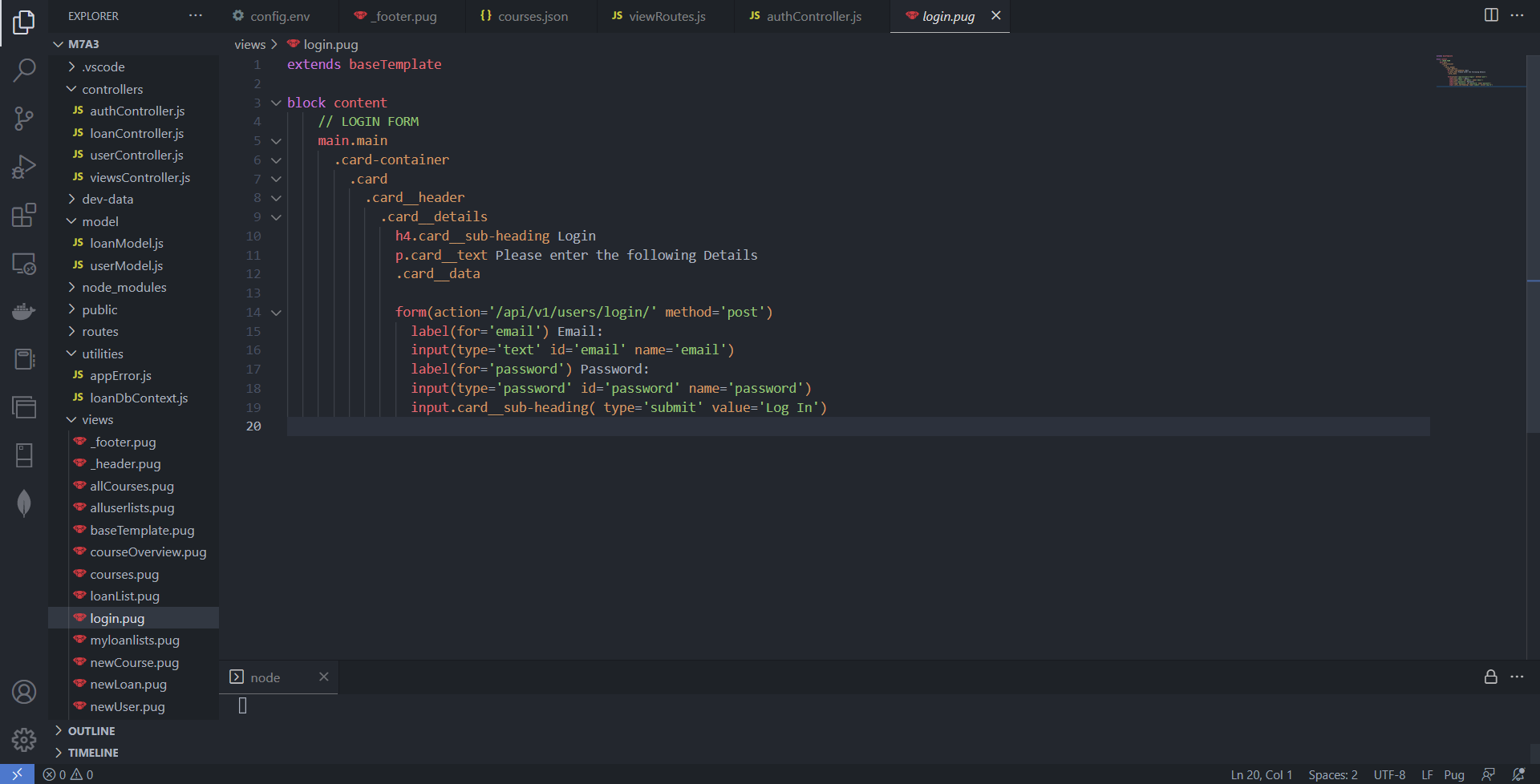
Description automatically generated

**LogIn the newly created user**

viewController.js navigating to the pug file



PUG file for the login page in the UI



Code:

extends baseTemplate

block content

    // LOGIN FORM

    main.main

      .card-container

        .card

          .card\_\_header

            .card\_\_details

              h4.card\_\_sub-heading Login

              p.card\_\_text Please enter the following Details

              .card\_\_data

              form(action='/api/v1/users/login/' method='post')

                label(for='email') Email:

                input(type='text' id='email' name='email')

                label(for='password') Password:

                input(type='password' id='password' name='password')

                input.card\_\_sub-heading( type='submit' value='Log In')

viewRoutes.js for the routing to the login page

Text

Description automatically generated

userRoutes.js for calling authentication on the login page

A screenshot of a computer

Description automatically generated with medium confidence

authController.js function showing login authentication

A screenshot of a computer

Description automatically generated with medium confidence

shows the login page when we click from overview page to login button.

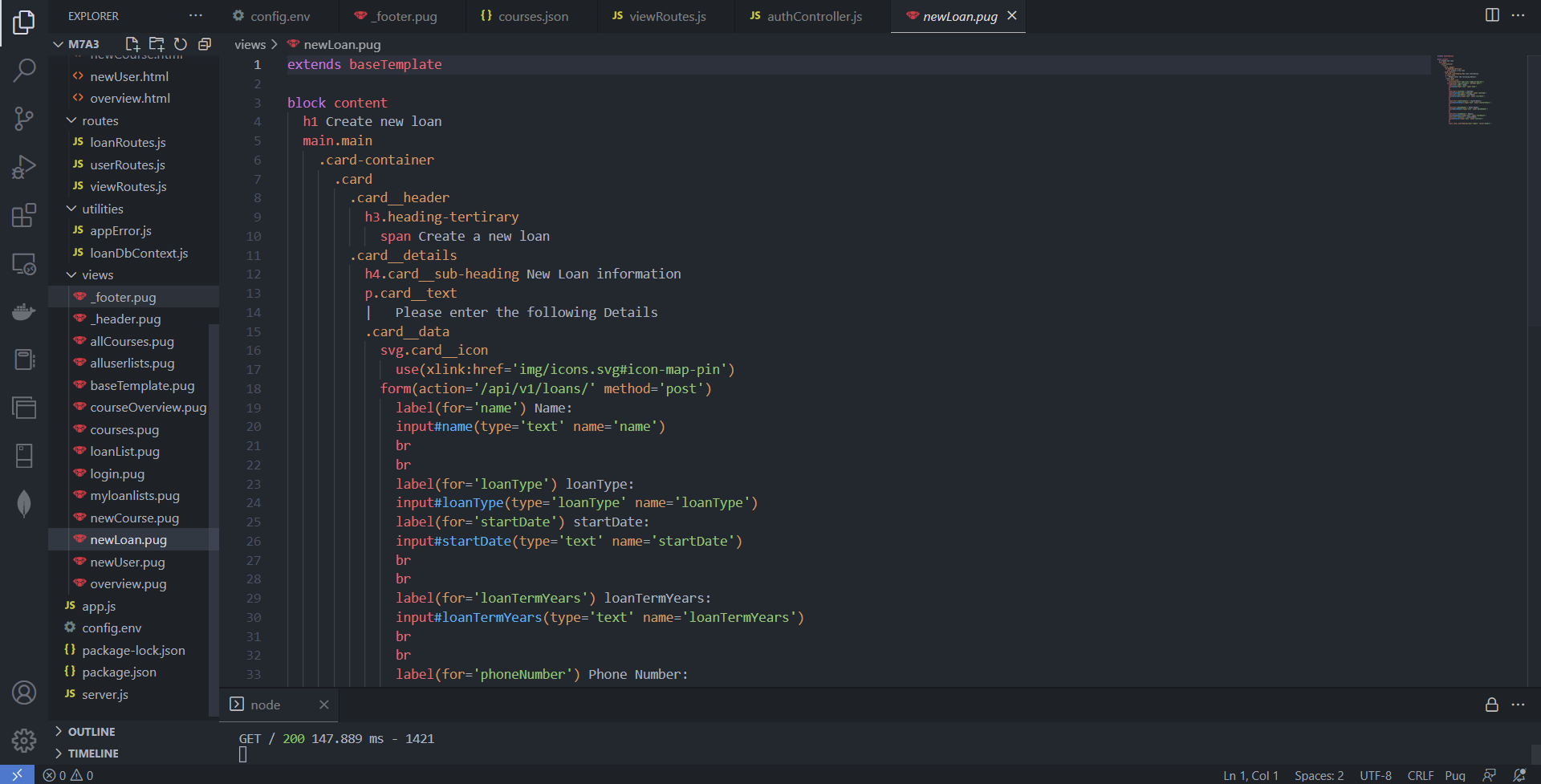
Graphical user interface, application

Description automatically generated Graphical user interface

Description automatically generated

**Step 3: The listing of loans**

PUG file for creating a new loan



Code:

extends baseTemplate

block content

  h1 Create new loan

  main.main

    .card-container

      .card

        .card\_\_header

          h3.heading-tertirary

            span Create a new loan

        .card\_\_details

          h4.card\_\_sub-heading New Loan information

          p.card\_\_text

          |   Please enter the following Details

          .card\_\_data

            svg.card\_\_icon

              use(xlink:href='img/icons.svg#icon-map-pin')

            form(action='/api/v1/loans/' method='post')

              label(for='name') Name:

              input#name(type='text' name='name')

              br

              br

              label(for='loanType') loanType:

              input#loanType(type='loanType' name='loanType')

              label(for='startDate') startDate:

              input#startDate(type='text' name='startDate')

              br

              br

              label(for='loanTermYears') loanTermYears:

              input#loanTermYears(type='text' name='loanTermYears')

              br

              br

              label(for='phoneNumber') Phone Number:

              input#phoneNumber(type='text' name='phoneNumber')

              br

              br

              label(for='loanAmount') Amount:

              input#loanAmount(type='text' name='loanAmount')

              label(for='interest') interestRate:

              input#interest(type='text' name='interest')

              br

              br

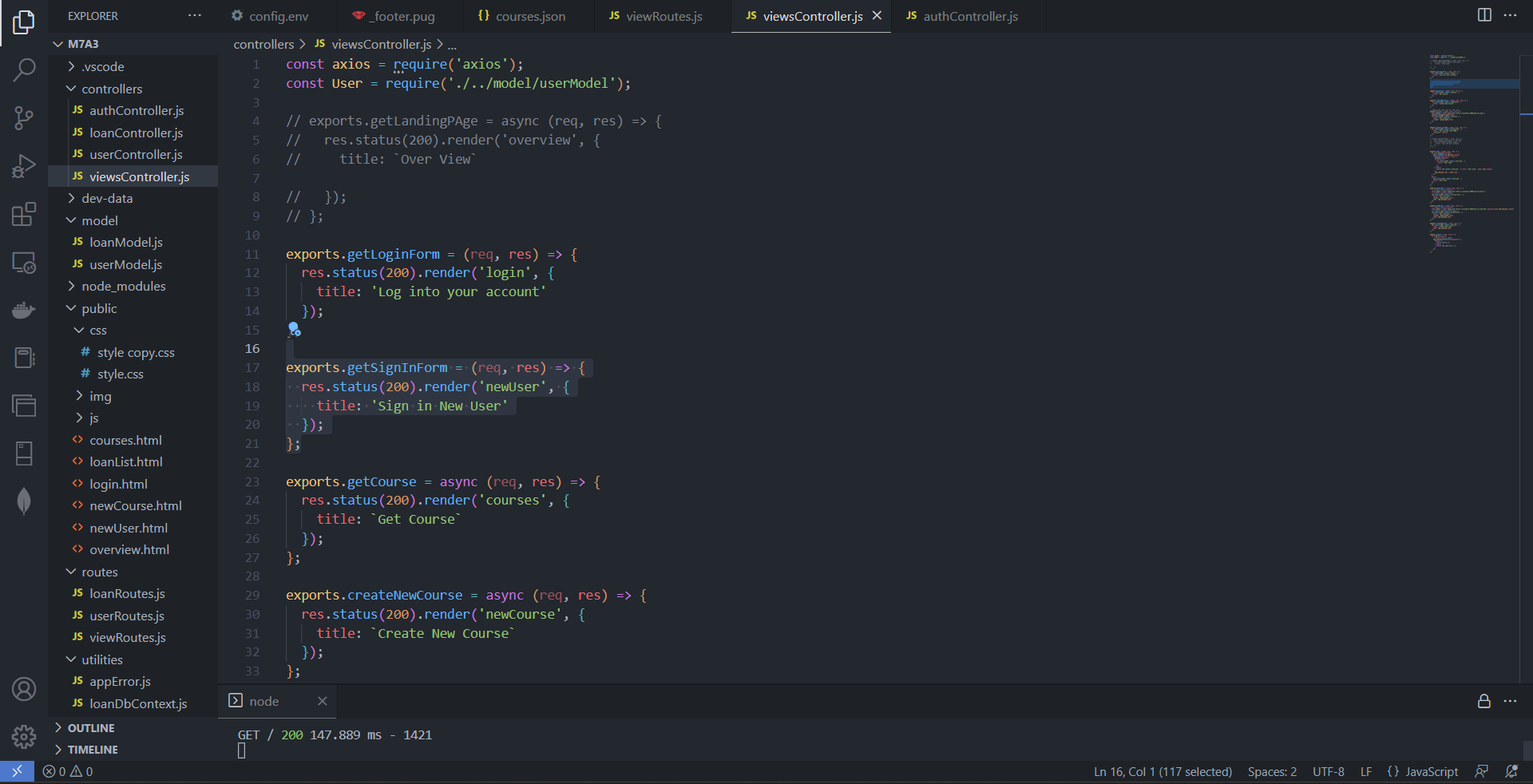
              input.card\_\_sub-heading(type='submit' value='Submit')

viewRoute.js for users loan list

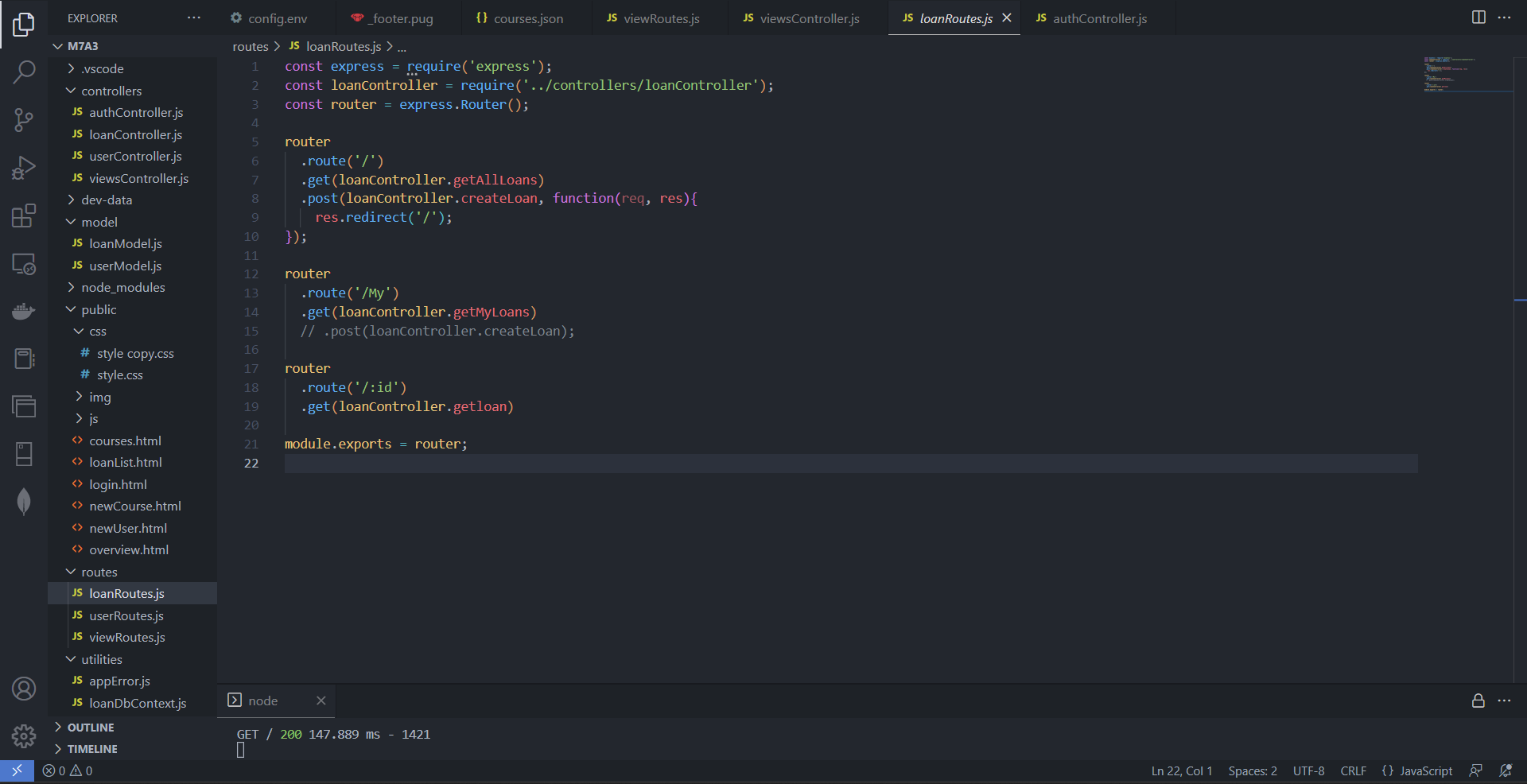
A screenshot of a computer

Description automatically generated with medium confidence

viewController.js hitting the loan routes for showing the list of loans user has (Figure 23)



LoanRoutes.js for showing user loans



Code:

const express = require('express');

const loanController = require('../controllers/loanController');

const router = express.Router();

router

  .route('/')

  .get(loanController.getAllLoans)

  .post(loanController.createLoan, function(req, res){

    res.redirect('/');

});

router

  .route('/My')

  .get(loanController.getMyLoans)

  // .post(loanController.createLoan);

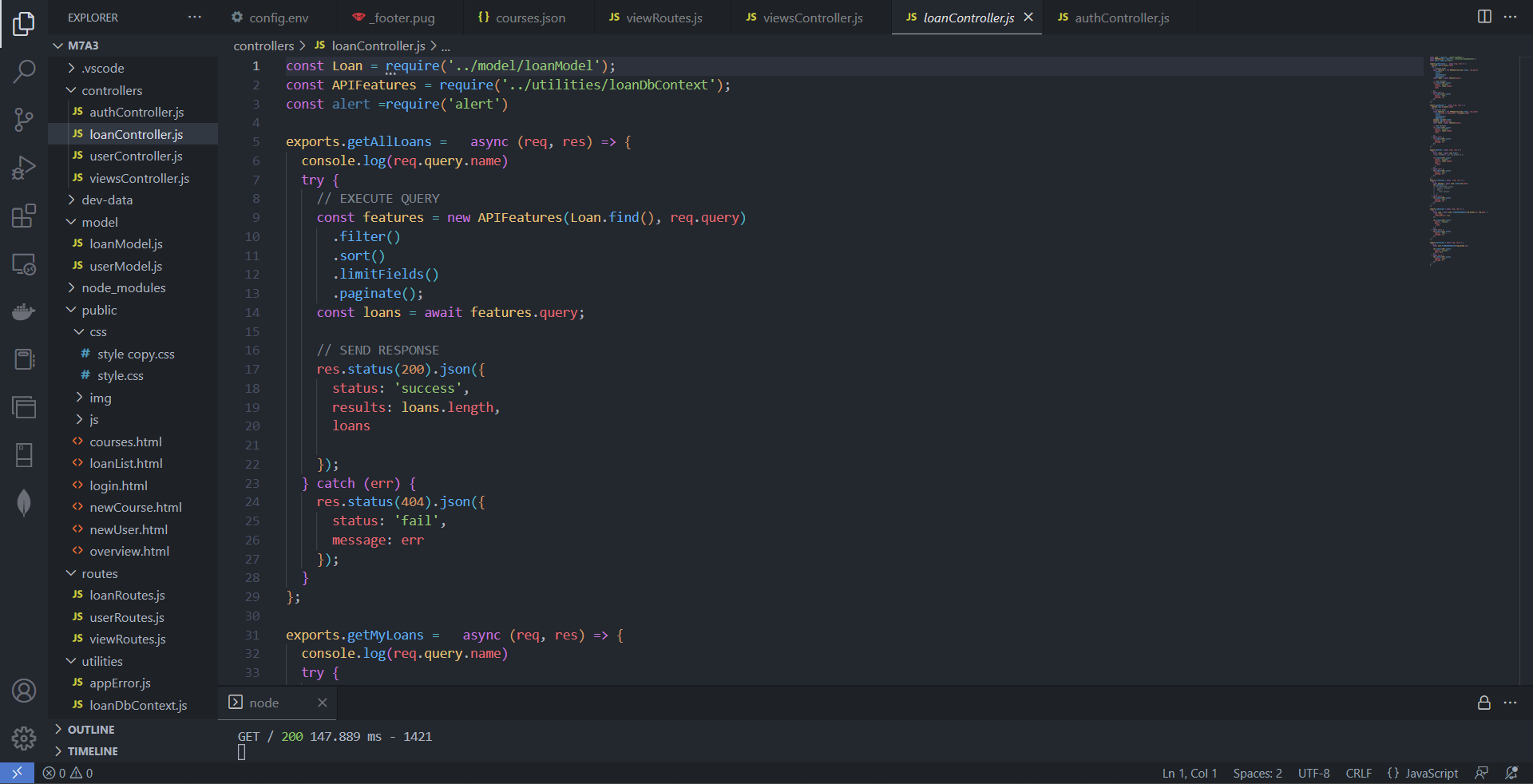
router

  .route('/:id')

  .get(loanController.getloan)

module.exports = router;

loanController.js for getting user loan list (Figure 25)



Code

const Loan = require('../model/loanModel');

const APIFeatures = require('../utilities/loanDbContext');

const alert =require('alert')

exports.getAllLoans =   async (req, res) => {

  console.log(req.query.name)

  try {

    // EXECUTE QUERY

    const features = new APIFeatures(Loan.find(), req.query)

      .filter()

      .sort()

      .limitFields()

      .paginate();

    const loans = await features.query;

    // SEND RESPONSE

    res.status(200).json({

      status: 'success',

      results: loans.length,

      loans

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

exports.getMyLoans =   async (req, res) => {

  console.log(req.query.name)

  try {

    // EXECUTE QUERY

    const features = new APIFeatures(Loan.find(), req.query)

      .filter(Loan => Loan.Name = req.query.name)

      .sort()

      .limitFields()

      .paginate();

    console.log(Loan.find);

    console.log(req.query);

    const loans = await features.query;

    // SEND RESPONSE

    res.status(200).json({

      status: 'success',

      results: loans.length,

      loans

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

exports.getloan = async (req, res) => {

  try {

    const loans = await Loan.find();

    //Loan.findOne({ \_id: req.params.id })

    res.status(200).json({

      status: 'success',

      results: loans.length,

      data: {

        loans

      }

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

exports.createLoan = async  (req, res) => {

  try {

    const newLoan = await Loan.create(req.body);

    res.redirect('/')

    // res.status(201).json({

    //   status: 'success',

    //   data: {

    //     loan: newLoan

    //   }

    // })

  } catch (err) {

    res.status(400).json({

      status: 'fail',

      message: err

    });

  }

};

exports.updateLoan = async (req, res) => {

  try {

    const loan = await Loan.findByIdAndUpdate(req.params.id, req.body, {

      new: true,

      runValidators: true

    });

    res.status(200).json({

      status: 'success',

      data: {

        loan

      }

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

      message: err

    });

  }

};

exports.deleteLoan = async (req, res) => {

  try {

    await Loan.findByIdAndDelete(req.params.id);

    res.status(200).json({

      status: 'success',

      data: null

    });

  } catch (err) {

    res.status(404).json({

      status: 'fail',

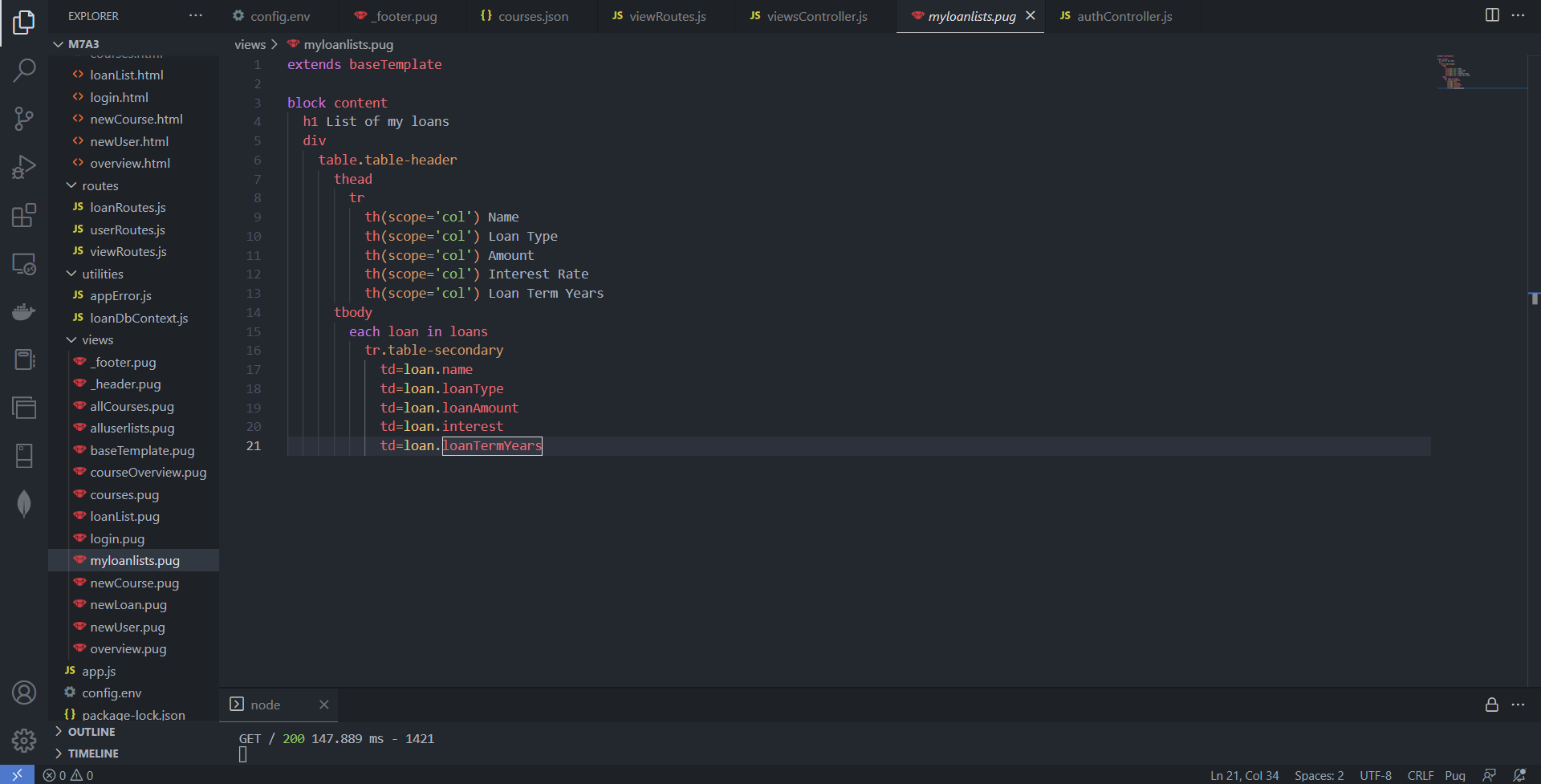
      message: err

    });

  }

};

PUG file showing the list of loans of the user



Code:

extends baseTemplate

block content

  h1 List of my loans

  div

    table.table-header

      thead

        tr

          th(scope='col') Name

          th(scope='col') Loan Type

          th(scope='col') Amount

          th(scope='col') Interest Rate

          th(scope='col') Loan Term Years

      tbody

        each loan in loans

          tr.table-secondary

            td=loan.name

            td=loan.loanType

            td=loan.loanAmount

            td=loan.interest

            td=loan.loanTermYears

Demonstrating the creation of loan by the user

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

Step 4: The MongoDB Atlas Database

Adding more users in the users collection in Mongo DB

Graphical user interface, text, application, email

Description automatically generated

Adding more loans for these users

Graphical user interface, text, application

Description automatically generated

Step 5: The users list of users

viewRoutes.js showing the route for listing all users

Text

Description automatically generated

userRoutes.js for user Controller function

Text

Description automatically generated

userController.js for all users

Text

Description automatically generated

PUG file for listing all users

Graphical user interface, text

Description automatically generated

extends baseTemplate

block content

  h1 This is user lists Overview

  div

    table.table-header

      thead

        tr

          th(scope='col') name

          th(scope='col') email

          th(scope='col') role

      tbody

        each user in users

          tr.table-secondary

            td=user.name

            td=user.email

            td=user.role

Demonstrating the listing of all users

Graphical user interface, application, website

Description automatically generated

Step 6: The loans

viewRoutes.js showing the get all loan route

Text

Description automatically generated

viewController.js file showing the functionality when the route is called

Text

Description automatically generated

loanRoute.js file showing which route will get all the loans

A screenshot of a computer

Description automatically generated with medium confidence

loanController.js file showing the functionality to implement for bringing all the loan

Text

Description automatically generated

PUG File for showing the UI of all loan list

A screenshot of a computer

Description automatically generated with medium confidence

Code:

extends baseTemplate

block content

  h3.heading-primary

    span List of Loans

  table.table-header

    thead

      tr

        th(scope='col') Name

        th(scope='col') Type of Loan

        th(scope='col') Phone Number

        th(scope='col') Amount

        th(scope='col') Interest Rate (%)

        th(scope='col') Term (Yrs)

    tbody

      each loan in loans

        tr.table-secondary

          td=loan.name

          td=loan.loanType

          td=loan.phoneNumber

          td=loan.loanAmount

          td=loan.interest

          td=loan.loanTermYears

Demonstrating the displaying of all loan list

Graphical user interface, text, application

Description automatically generated