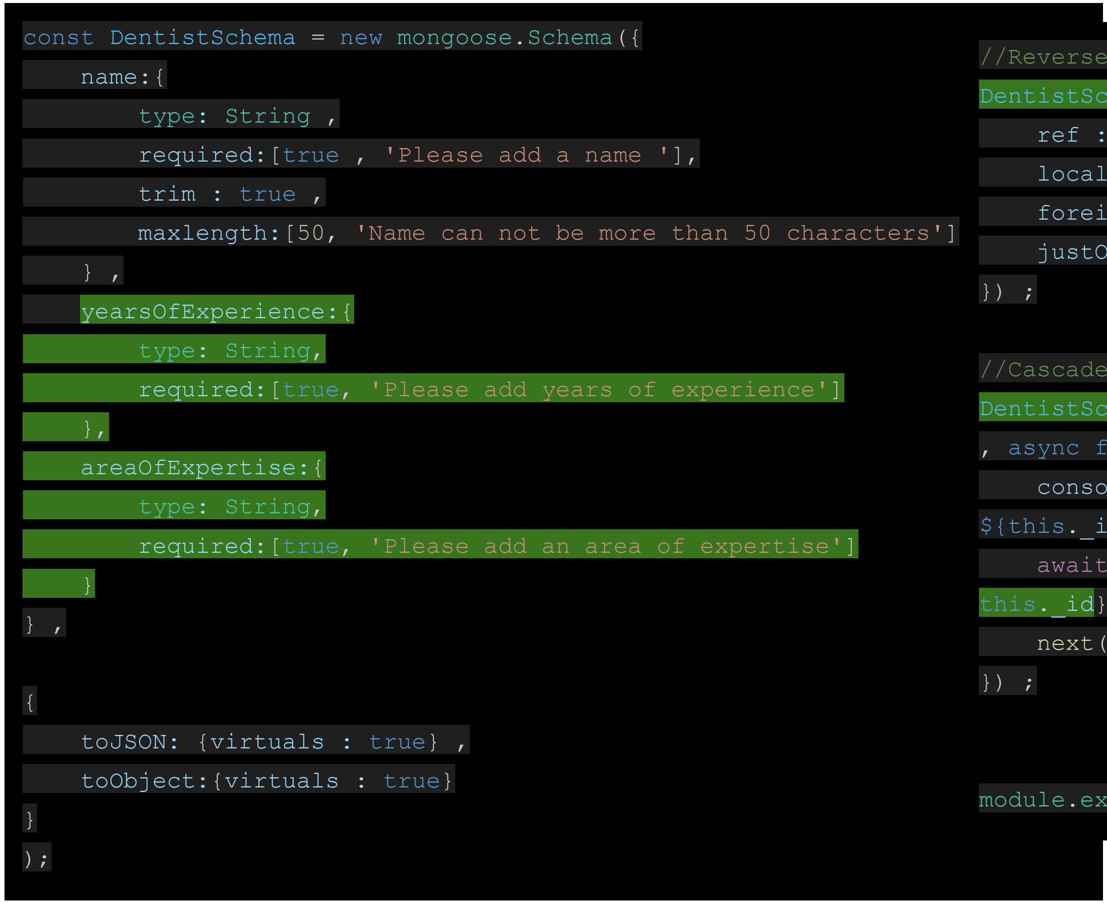
This is My Kingdom Come

Project#4: Dentist Booking

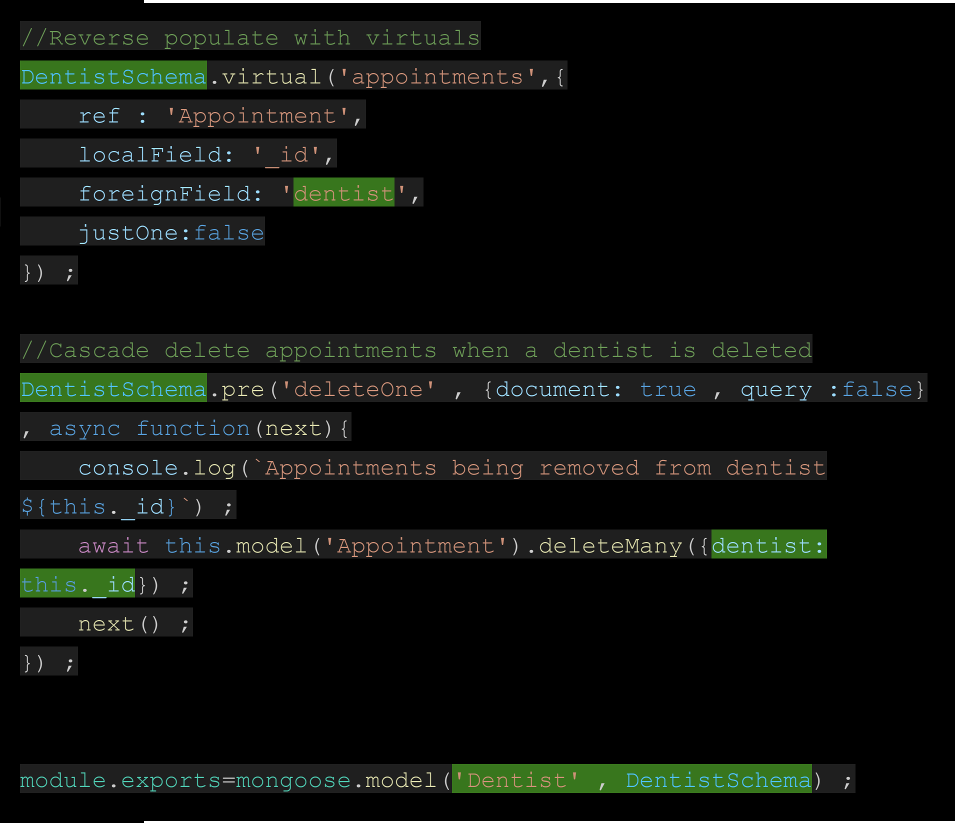
Member

1. 6633115821 ธีรุตม์ สุวรรณระดา
2. 6633143321 ปฤณสิษฐ์ จำปา
3. 6633186321 ภัทรดนัย อาศรมเงิน

models/Dentist.js

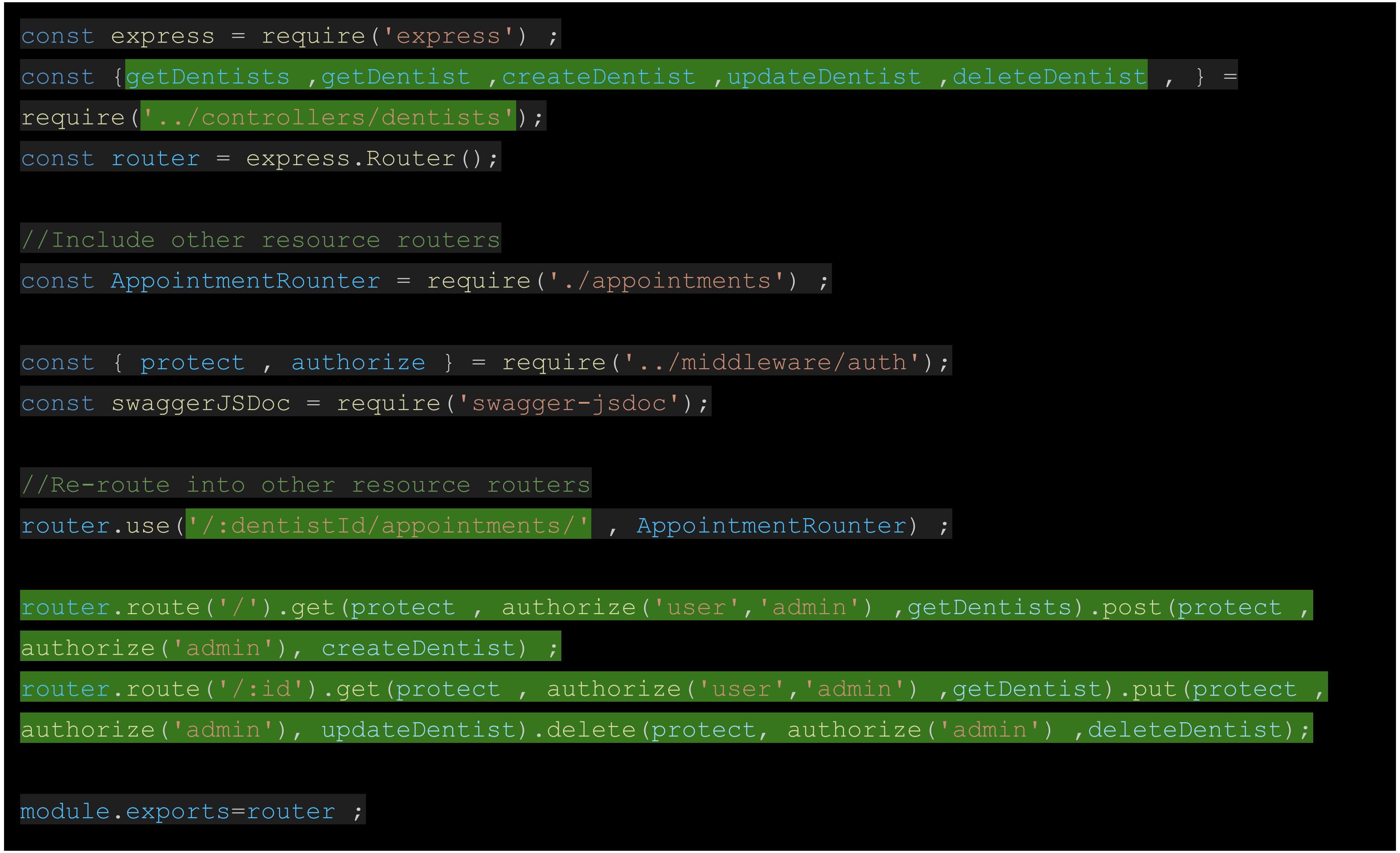


```
const DentistSchema = new mongoose.Schema({
    name:{
        type: String ,
        required:[true , 'Please add a name '],
        trim : true ,
        maxlength:[50, 'Name can not be more than 50 characters']
    } ,
    yearsOfExperience:{
        type: String,
        required:[true, 'Please add years of experience']
    },
    areaOfExpertise:{
        type: String,
        required:[true, 'Please add an area of expertise']
    }
} ,

{
    toJSON: {virtuals : true} ,
    toObject:{virtuals : true}
}
);
```



```
//Reverse populate with virtuals
DentistSchema.virtual('appointments',{
    ref : 'Appointment',
    localField: '_id',
    foreignField: 'dentist',
    justOne:false
}) ;

//Cascade delete appointments when a dentist is deleted
DentistSchema.pre('deleteOne' , {document: true , query :false}
, async function(next){
    console.log(`Appointments being removed from dentist
${this._id}`) ;
    await this.model('Appointment').deleteMany({dentist:
this._id}) ;
    next() ;
}) ;


module.exports=mongoose.model('Dentist' , DentistSchema) ;
```

routes/dentists.js



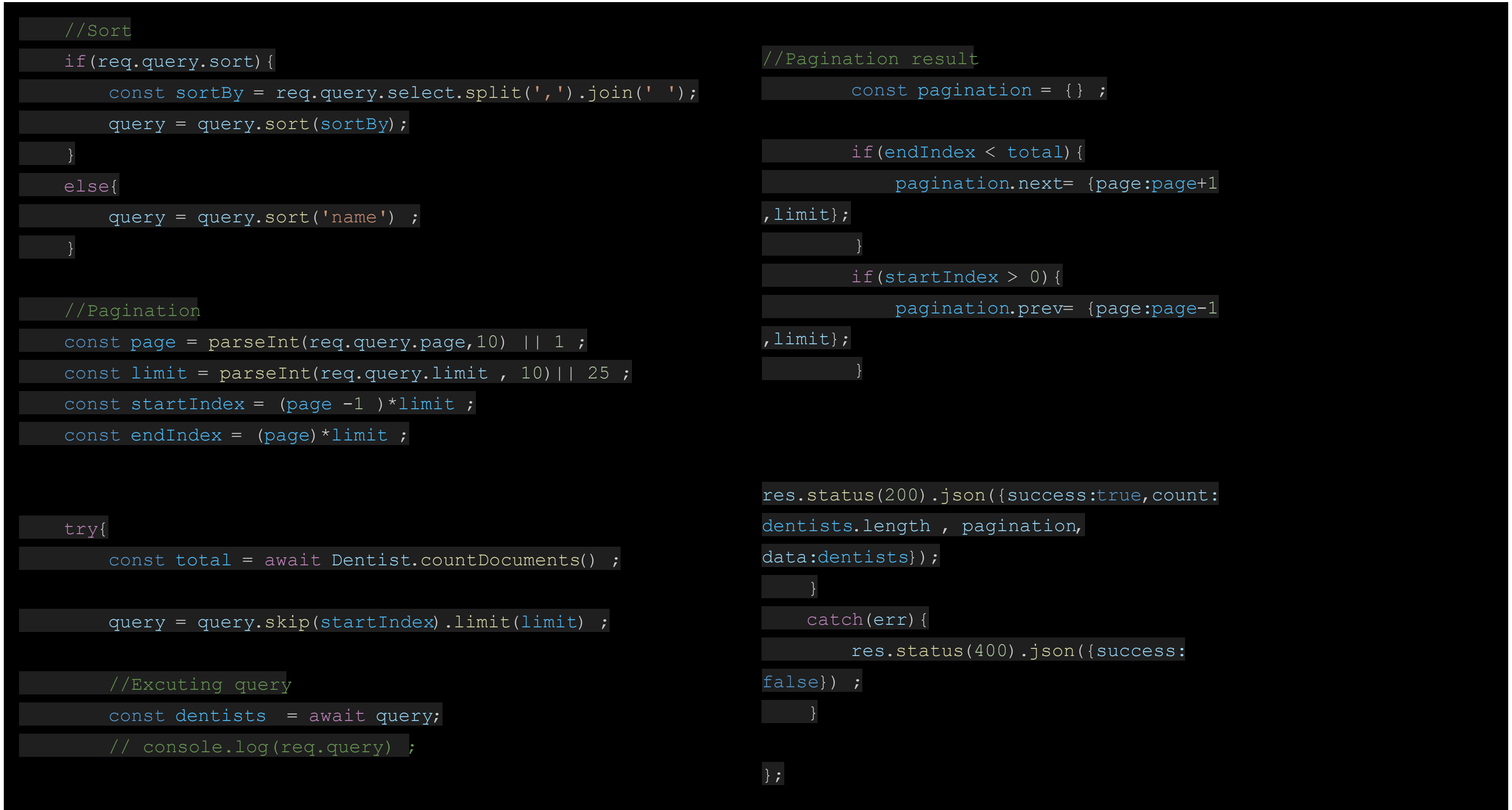
```
const express = require('express') ;
const {getDentists ,getDentist ,createDentist ,updateDentist ,deleteDentist , } =
require('../controllers/dentists');
const router = express.Router();

//Include other resource routers
const AppointmentRounter = require('./appointments') ;

const { protect , authorize } = require('../middleware/auth');
const swaggerJSDoc = require('swagger-jsdoc');

//Re-route into other resource routers
router.use('/:dentistId/appointments/' , AppointmentRounter) ;

router.route('/').get(protect , authorize('user','admin') ,getDentists).post(protect ,
authorize('admin'), createDentist) ;
router.route('/:id').get(protect , authorize('user','admin') ,getDentist).put(protect ,
authorize('admin'), updateDentist).delete(protect, authorize('admin') ,deleteDentist);

module.exports=router ;
```

controllers/dentists.js

```
const Dentist = require('../models/Dentist');

//@desc  Get all dentists
//@route GeT /api/v1/dentists
//@access Public
exports.getDentists  = async (req , res , next)=>{
    let query ;
    //copy req.query
    const reqQuery = {...req.query};
    //Fields to exclude
    const removeFields= ['select' , 'sort' , 'page' , 'limit'] ;

    //Loop over remove  frelds and delete them from req Query
    removeFields.forEach(param=> delete reqQuery[param]);
    console.log(reqQuery) ;
```

```
//Create query String
    let queryStr = JSON.stringify(reqQuery) ;
    queryStr = queryStr.replace(/\b(gt|gte|lt|lte|in)\b/g ,
match=> `$${match}`);

    if(req.user.role === "admin"){
        query =
Dentist.find(JSON.parse(queryStr)).populate('appointments');
    }
    else{
        query = Dentist.find(JSON.parse(queryStr)) ;
    }
('appointments');

    //Select Fields
    if(req.query.select){
        const fields = req.query.select.split(',').join(' ');
        query = query.select(fields) ;
    }
```

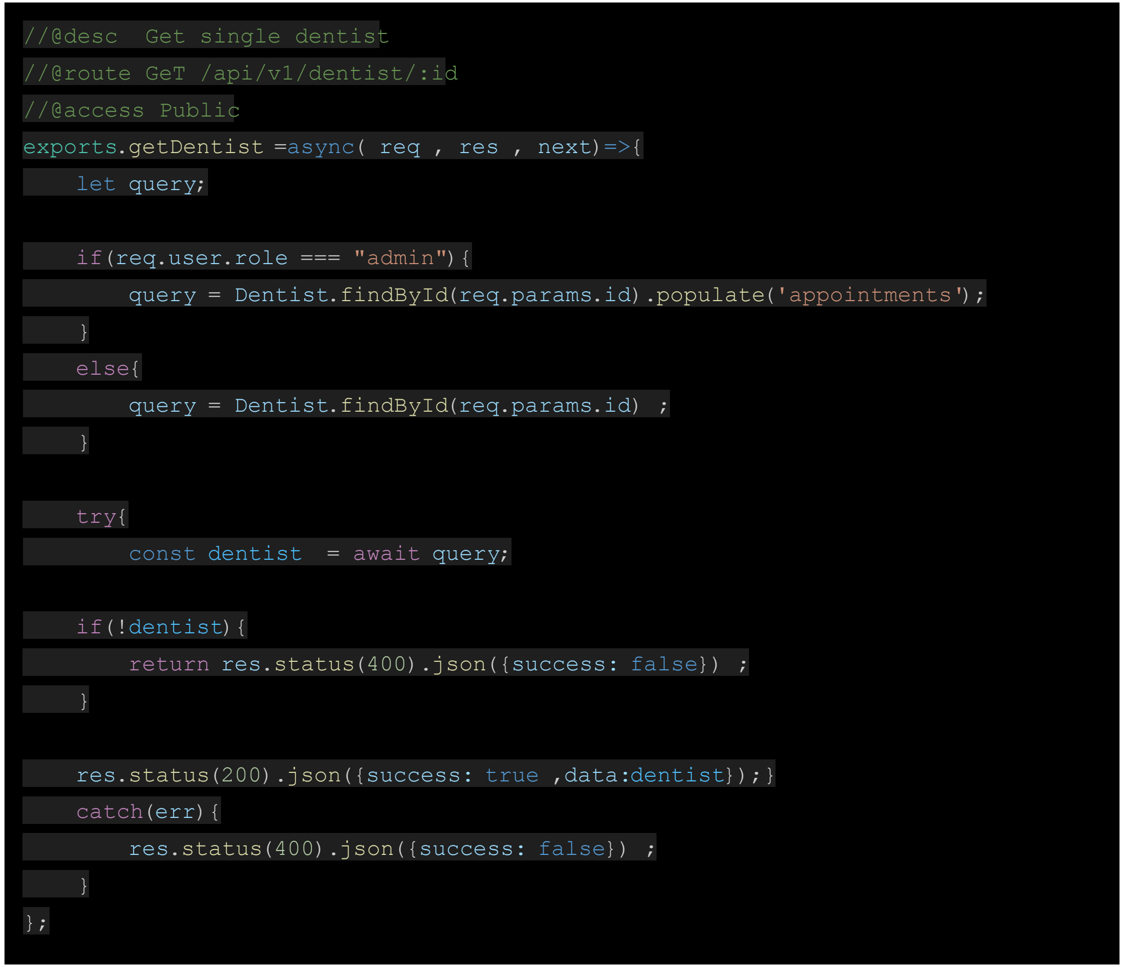
controllers/dentists.js



```
    //Sort
    if(req.query.sort){
        const sortBy = req.query.select.split(',').join(' ');
        query = query.sort(sortBy);
    }
    else{
        query = query.sort('name') ;
    }

    //Pagination
    const page = parseInt(req.query.page,10) || 1 ;
    const limit = parseInt(req.query.limit , 10)|| 25 ;
    const startIndex = (page -1 )*limit ;
    const endIndex = (page)*limit ;


    try{
        const total = await Dentist.countDocuments() ;

        query = query.skip(startIndex).limit(limit) ;

        //Excuting query
        const dentists  = await query;
        // console.log(req.query) ;
```

```
//Pagination result
        const pagination = {} ;

        if(endIndex < total){
            pagination.next= {page:page+1
,limit};
        }
        if(startIndex > 0){
            pagination.prev= {page:page-1
,limit};
        }



res.status(200).json({success:true,count:
dentists.length , pagination,
data:dentists});
    }
    catch(err){
        res.status(400).json({success:
false}) ;
    }

};
```

controllers/dentists.js



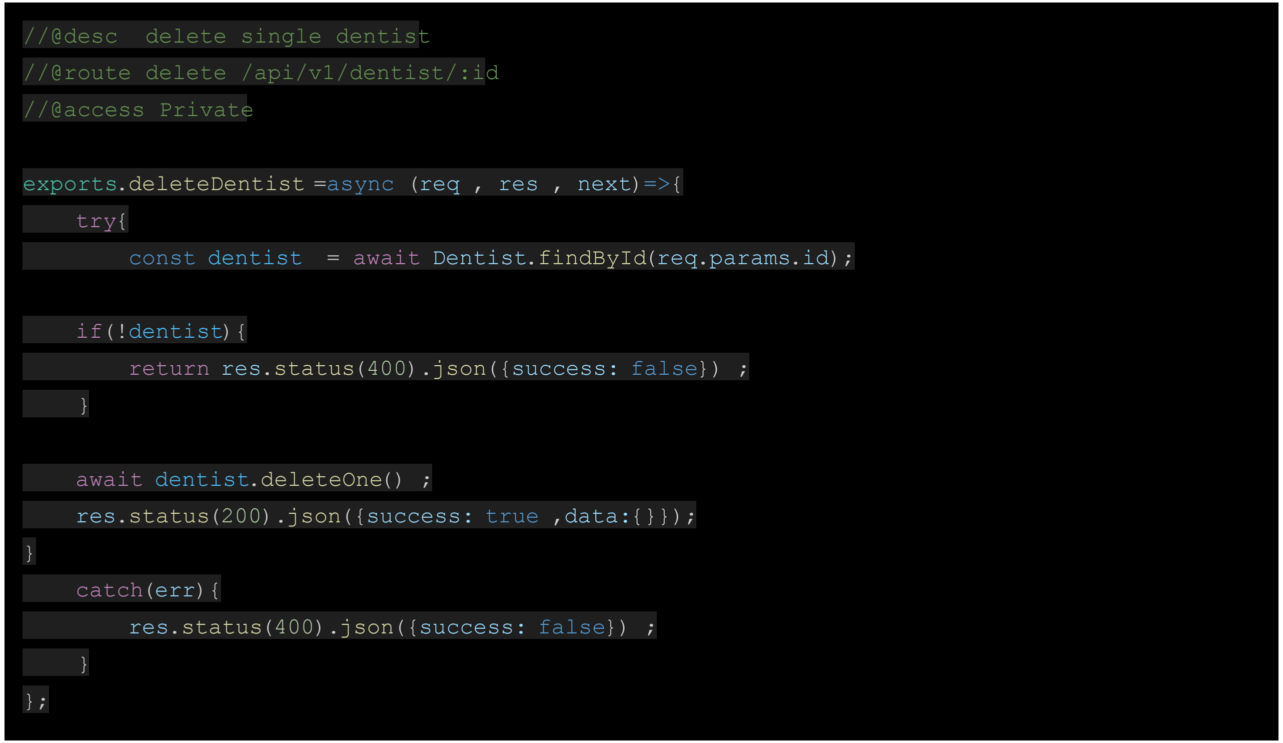
```
//@desc  Get single dentist
//@route GeT /api/v1/dentist/:id
//@access Public
exports.getDentist =async( req , res , next)=>{
    let query;

    if(req.user.role === "admin"){
        query = Dentist.findById(req.params.id).populate('appointments');
    }
    else{
        query = Dentist.findById(req.params.id) ;
    }

    try{
        const dentist  = await query;

    if(!dentist){
        return res.status(400).json({success: false}) ;
    }

    res.status(200).json({success: true ,data:dentist});}
    catch(err){
        res.status(400).json({success: false}) ;
    }
};
```

controllers/dentists.js

```javascript
//@desc  update single dentist
//@route put /api/v1/dentist/:id
//@access Private

exports.updateDentist =async (req , res , next)=>{
    try{
        const dentist  = await Dentist.findByIdAndUpdate(req.params.id ,
req.body ,{
            new: true ,
            runValidators: true

        });

    if(!dentist){
        return res.status(400).json({success: false}) ;
    }

    res.status(200).json({success: true ,data:dentist});}
    catch(err){
        res.status(400).json({success: false}) ;
    }
};
```

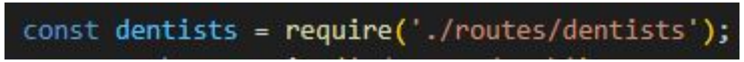
```javascript
//@desc  Creaet a  Dentist
//@route Post /api/v1/dentists
//@access Private

exports.createDentist = async (req , res , next)=>{
    const dentist = await Dentist.create(req.body);
    res.status(201).json({success:true, data:dentist});
};
```

controllers/dentists.js

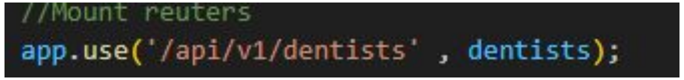


```javascript
//@desc  delete single dentist
//@route delete /api/v1/dentist/:id
//@access Private

exports.deleteDentist =async (req , res , next)=>{
    try{
        const dentist  = await Dentist.findById(req.params.id);

    if(!dentist){
        return res.status(400).json({success: false}) ;
    }

    await dentist.deleteOne() ;
    res.status(200).json({success: true ,data:{}});
}
    catch(err){
        res.status(400).json({success: false}) ;
    }
};
```

server.js

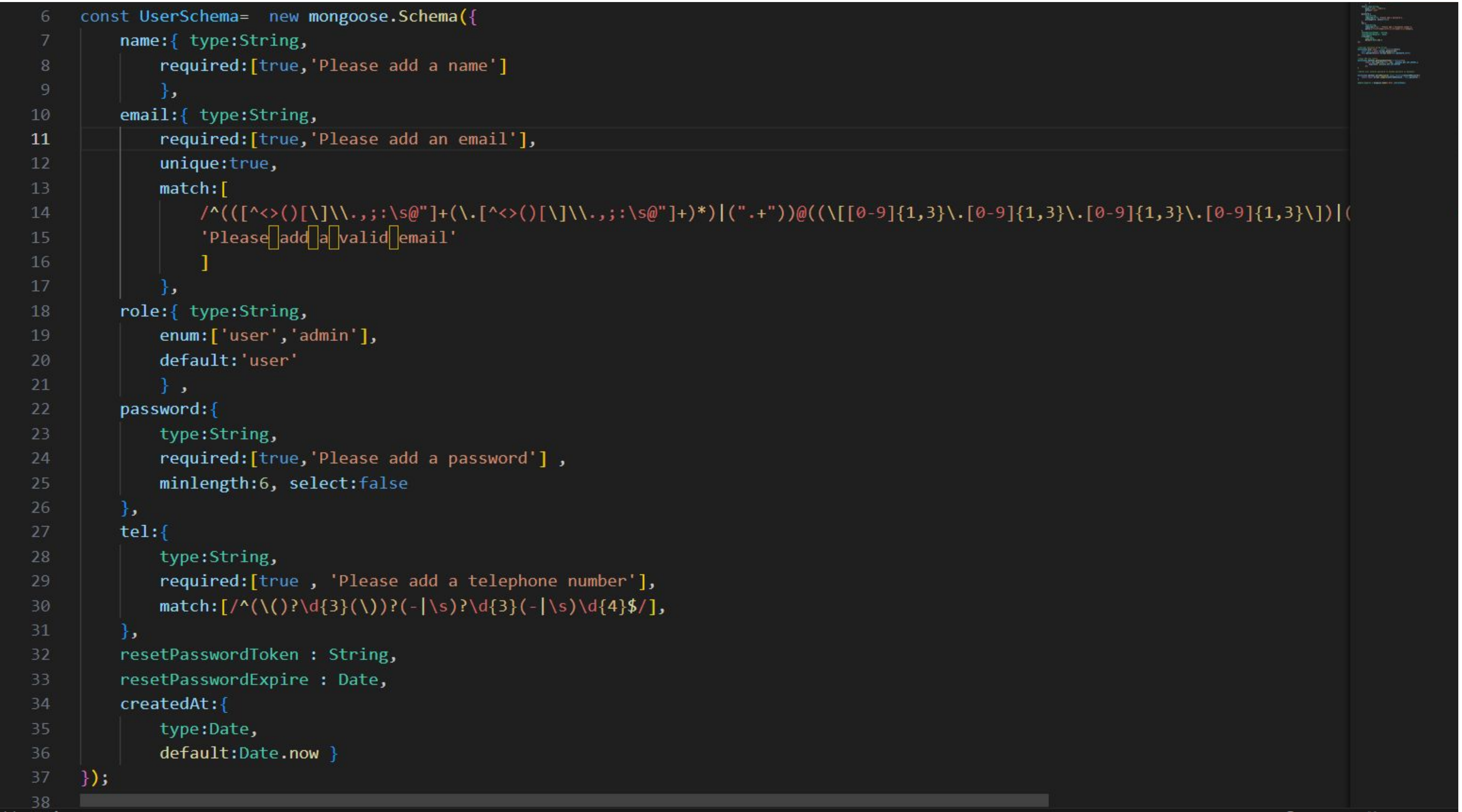


```
const dentists = require('./routes/dentists');
```



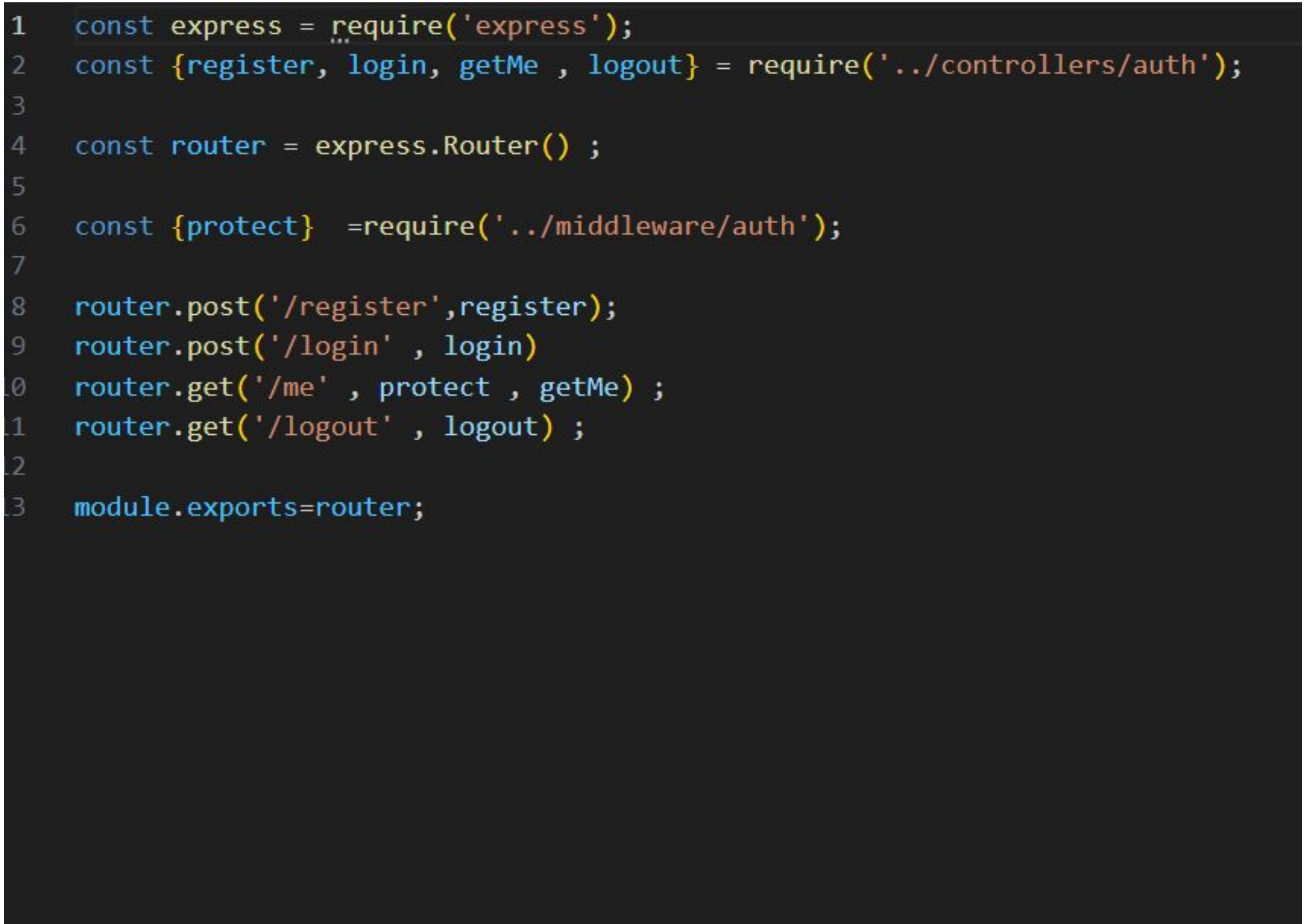
```
//Mount reuters
app.use('/api/v1/dentists' , dentists);
```

models/User.js



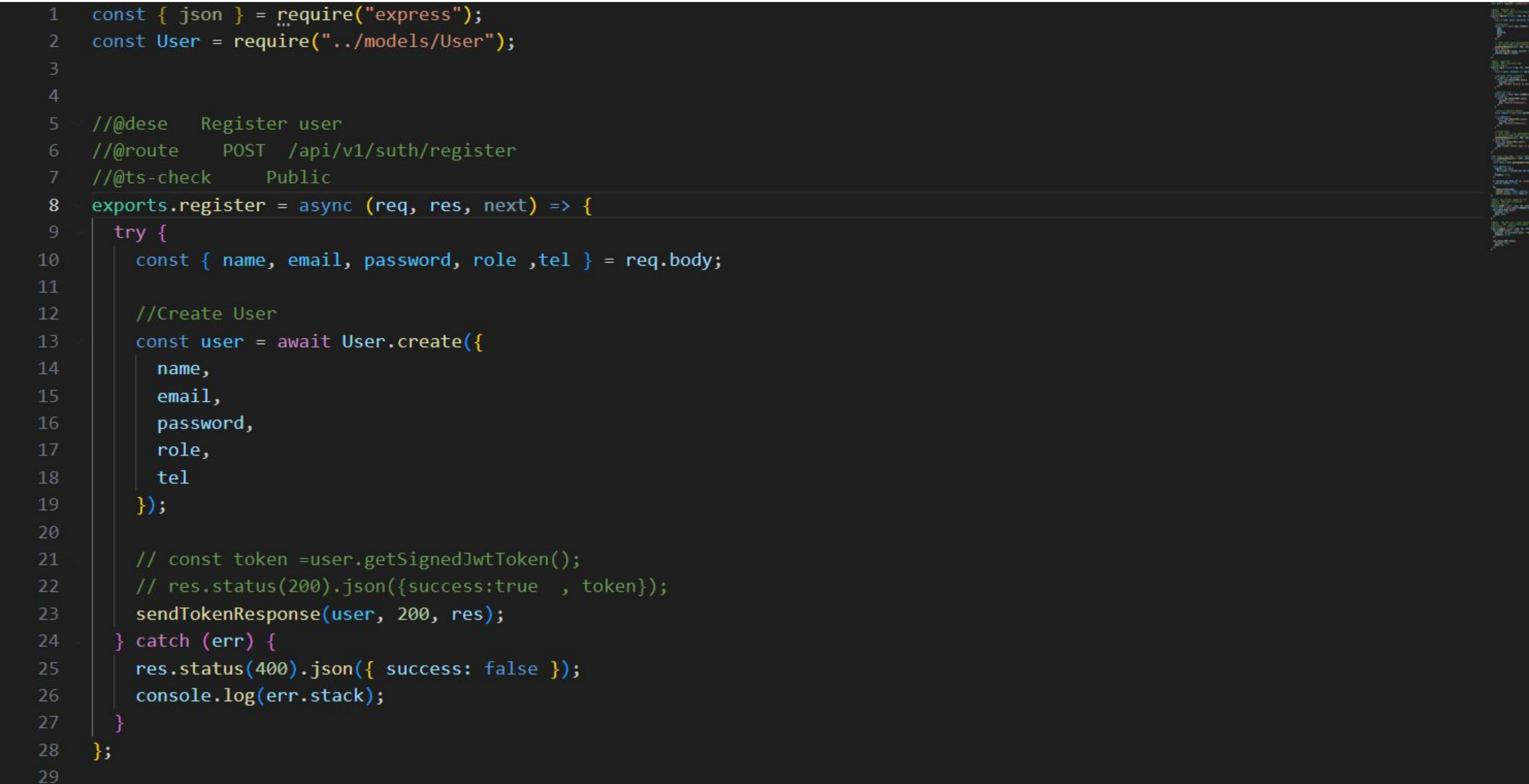
```
const UserSchema=  new mongoose.Schema({
    name:{ type:String,
        required:[true,'Please add a name']
        },
    email:{ type:String,
        required:[true,'Please add an email'],
        unique:true,
        match:[
            /^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(
            'Please add a valid email'
            ]
        },
    role:{ type:String,
        enum:['user','admin'],
        default:'user'
        } ,
    password:{
        type:String,
        required:[true,'Please add a password'] ,
        minlength:6, select:false
    },
    tel:{
        type:String,
        required:[true , 'Please add a telephone number'],
        match:[/^(\()?\d{3}(\))?(-|\s)?\d{3}(-|\s)\d{4}$/],
    },
    resetPasswordToken : String,
    resetPasswordExpire : Date,
    createdAt:{
        type:Date,
        default:Date.now }
});
```

routes/auth.js



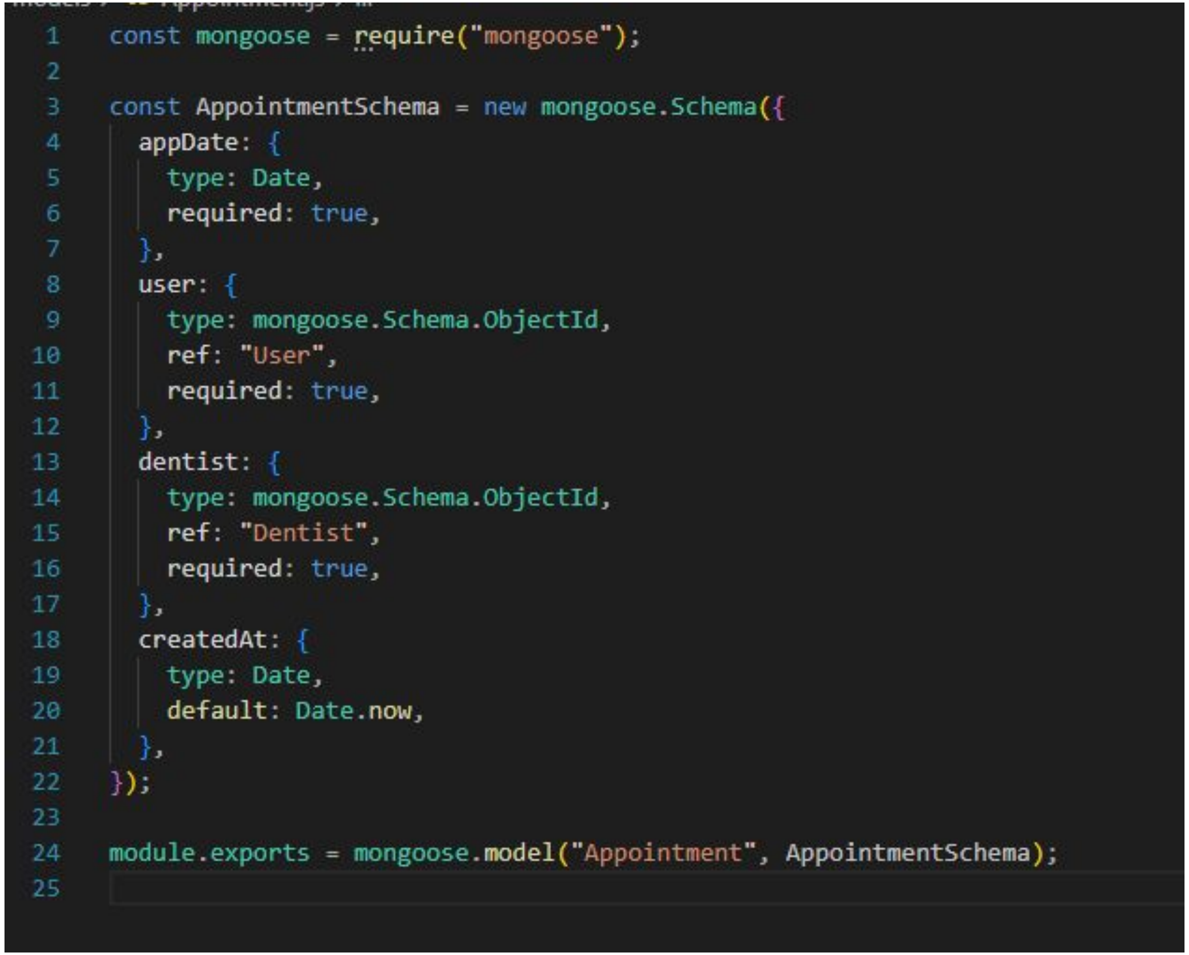
```javascript
const express = require('express');
const {register, login, getMe , logout} = require('../controllers/auth');

const router = express.Router() ;

const {protect}  =require('../middleware/auth');

router.post('/register',register);
router.post('/login' , login)
router.get('/me' , protect , getMe) ;
router.get('/logout' , logout) ;

module.exports=router;
```

controllers/auth.js



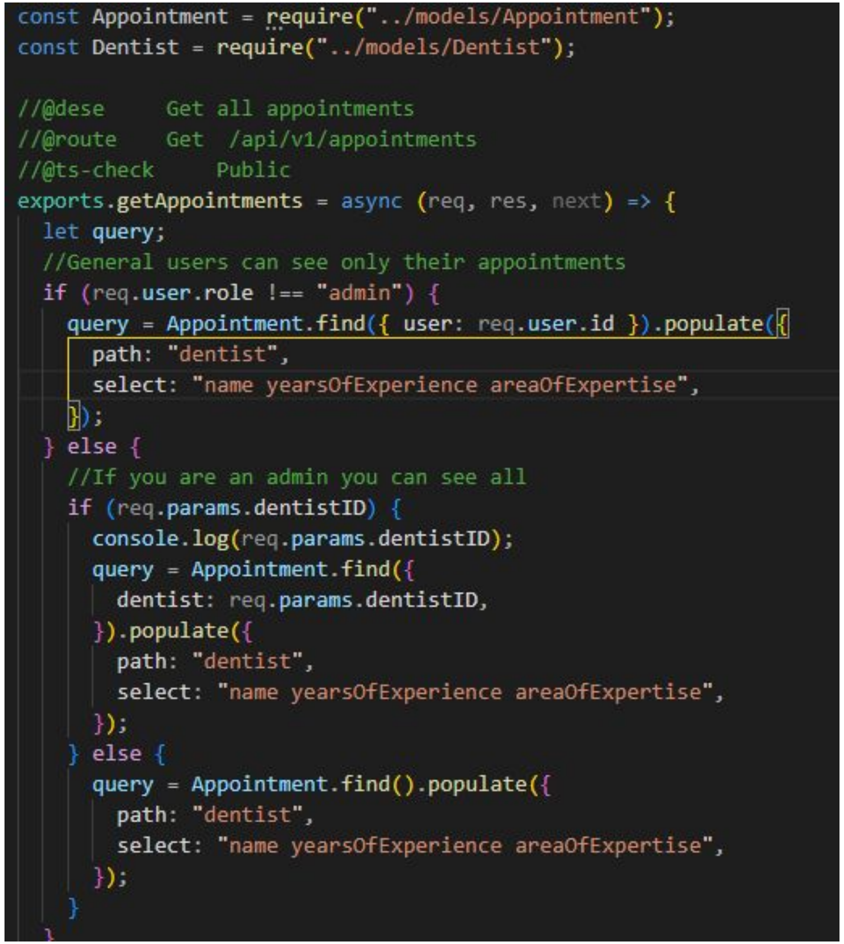
```javascript
const { json } = require("express");
const User = require("../models/User");


//@dese   Register user
//@route    POST  /api/v1/suth/register
//@ts-check     Public
exports.register = async (req, res, next) => {
  try {
    const { name, email, password, role ,tel } = req.body;

    //Create User
    const user = await User.create({
      name,
      email,
      password,
      role,
      tel
    });

    // const token =user.getSignedJwtToken();
    // res.status(200).json({success:true   , token});
    sendTokenResponse(user, 200, res);
  } catch (err) {
    res.status(400).json({ success: false });
    console.log(err.stack);
  }
};

```

models/Appointment.js

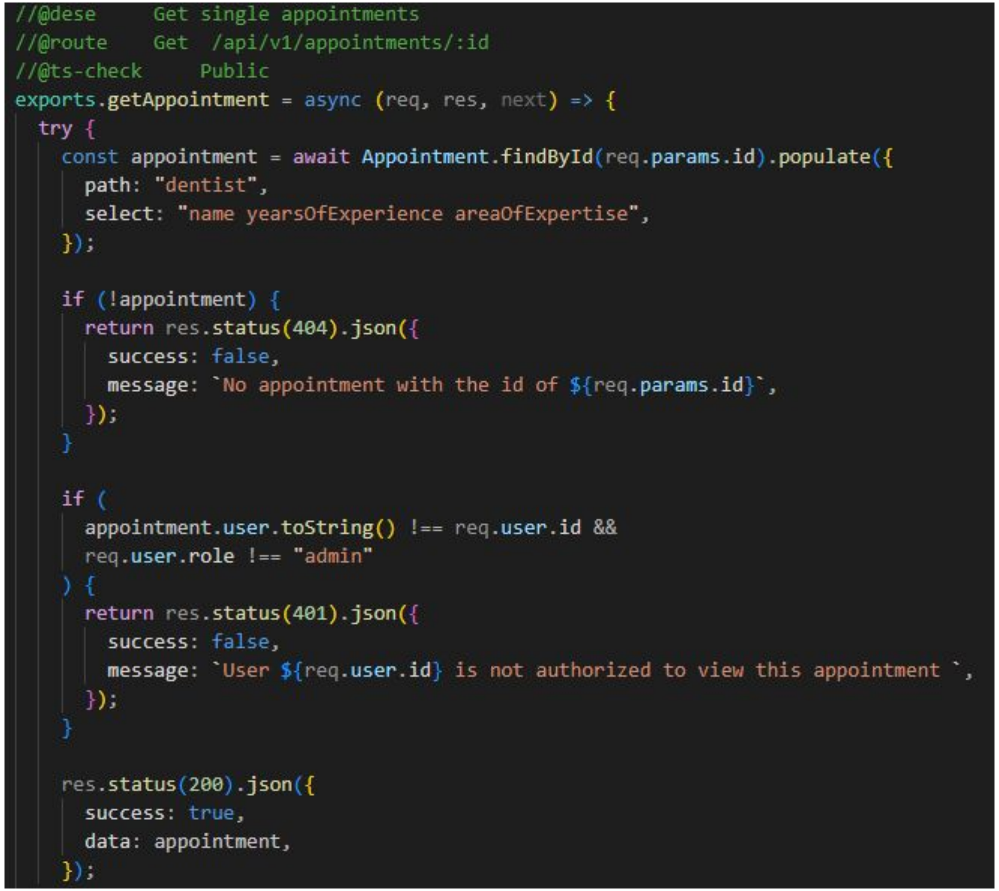


```javascript
const mongoose = require("mongoose");

const AppointmentSchema = new mongoose.Schema({
  appDate: {
    type: Date,
    required: true,
  },
  user: {
    type: mongoose.Schema.ObjectId,
    ref: "User",
    required: true,
  },
  dentist: {
    type: mongoose.Schema.ObjectId,
    ref: "Dentist",
    required: true,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});

module.exports = mongoose.model("Appointment", AppointmentSchema);

```

controllers/appointments.js

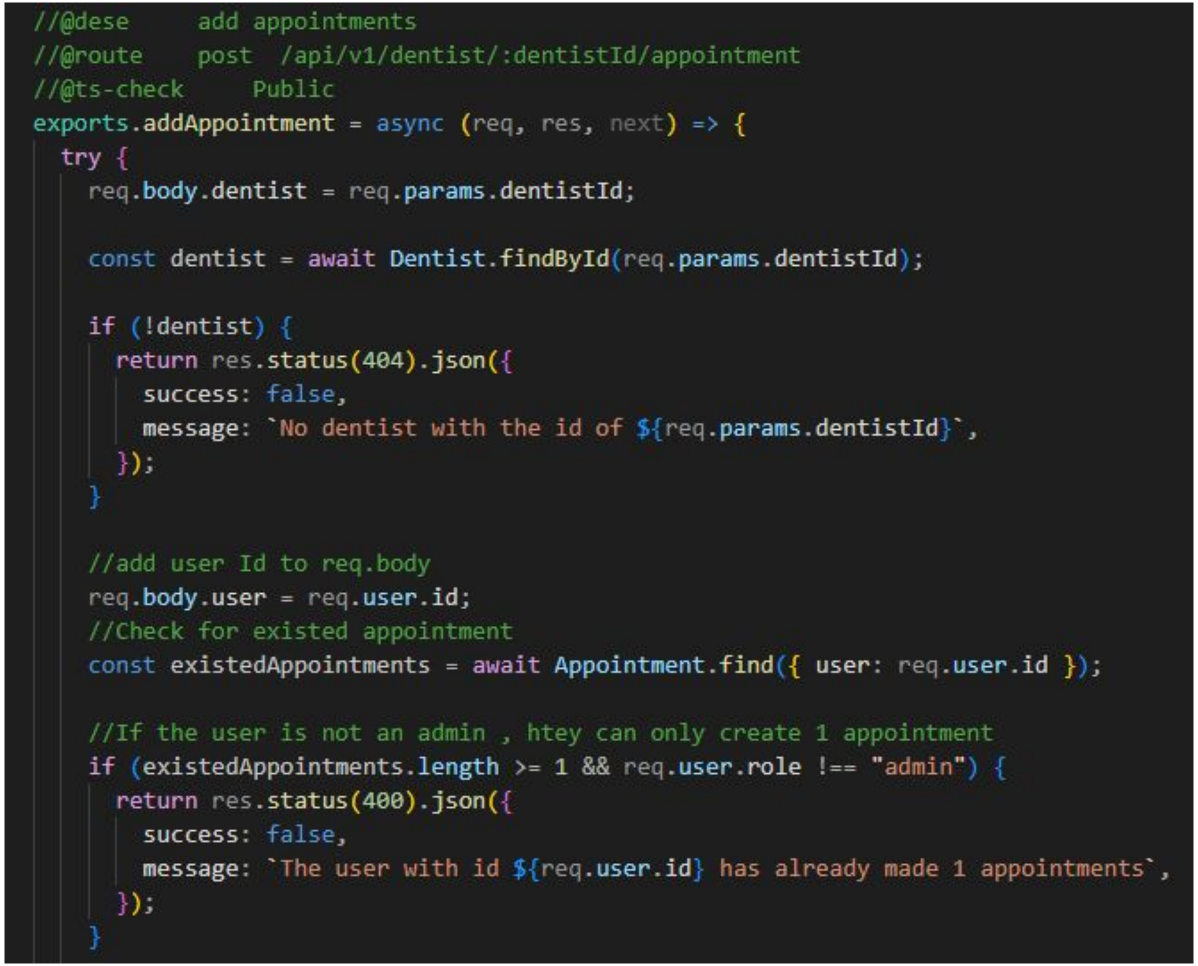


```javascript
const Appointment = require("../models/Appointment");
const Dentist = require("../models/Dentist");

//@dese     Get all appointments
//@route    Get  /api/v1/appointments
//@ts-check     Public
exports.getAppointments = async (req, res, next) => {
  let query;
  //General users can see only their appointments
  if (req.user.role !== "admin") {
    query = Appointment.find({ user: req.user.id }).populate({
      path: "dentist",
      select: "name yearsOfExperience areaOfExpertise",
    });
  } else {
    //If you are an admin you can see all
    if (req.params.dentistID) {
      console.log(req.params.dentistID);
      query = Appointment.find({
        dentist: req.params.dentistID,
      }).populate({
        path: "dentist",
        select: "name yearsOfExperience areaOfExpertise",
      });
    } else {
      query = Appointment.find().populate({
        path: "dentist",
        select: "name yearsOfExperience areaOfExpertise",
      });
    }
  }
```



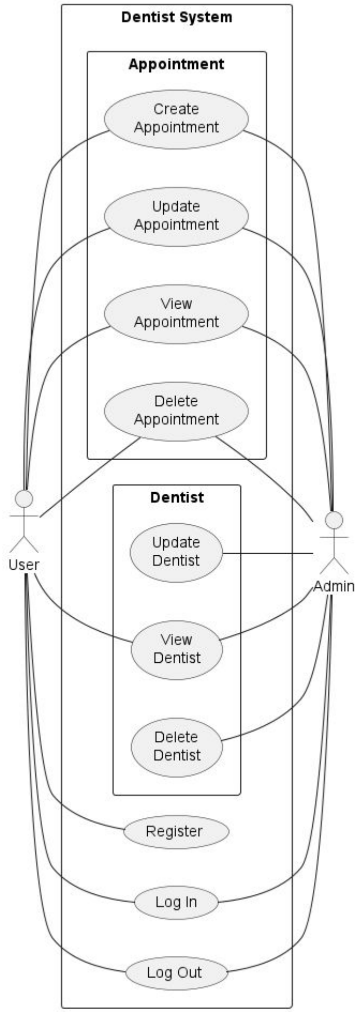
```javascript
//@dese     Get single appointments
//@route    Get  /api/v1/appointments/:id
//@ts-check     Public
exports.getAppointment = async (req, res, next) => {
  try {
    const appointment = await Appointment.findById(req.params.id).populate({
      path: "dentist",
      select: "name yearsOfExperience areaOfExpertise",
    });

    if (!appointment) {
      return res.status(404).json({
        success: false,
        message: `No appointment with the id of ${req.params.id}`,
      });
    }

    if (
      appointment.user.toString() !== req.user.id &&
      req.user.role !== "admin"
    ) {
      return res.status(401).json({
        success: false,
        message: `User ${req.user.id} is not authorized to view this appointment `,
      });
    }

    res.status(200).json({
      success: true,
      data: appointment,
    });
```

controllers/appointments.js

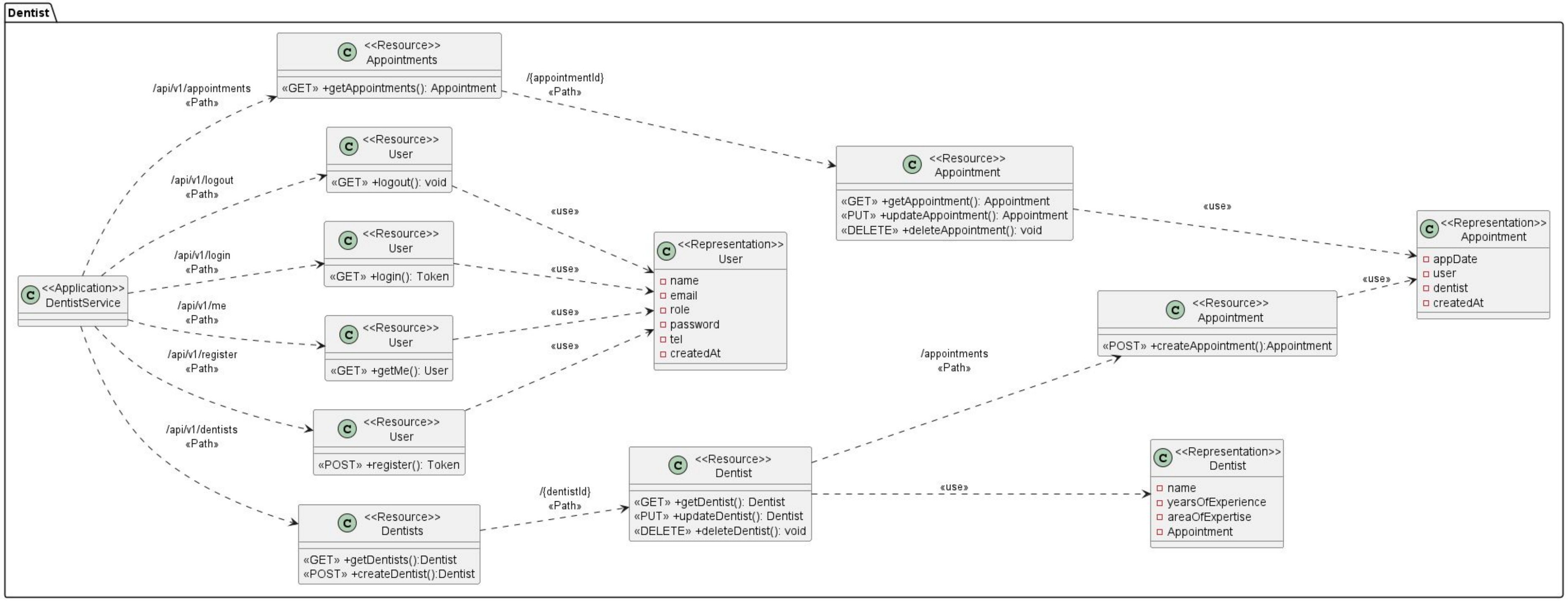


```
//@dese     add appointments
//@route    post  /api/v1/dentist/:dentistId/appointment
//@ts-check     Public
exports.addAppointment = async (req, res, next) => {
  try {
    req.body.dentist = req.params.dentistId;

    const dentist = await Dentist.findById(req.params.dentistId);

    if (!dentist) {
      return res.status(404).json({
        success: false,
        message: `No dentist with the id of ${req.params.dentistId}`,
      });
    }

    //add user Id to req.body
    req.body.user = req.user.id;
    //Check for existed appointment
    const existedAppointments = await Appointment.find({ user: req.user.id });

    //If the user is not an admin , htey can only create 1 appointment
    if (existedAppointments.length >= 1 && req.user.role !== "admin") {
      return res.status(400).json({
        success: false,
        message: `The user with id ${req.user.id} has already made 1 appointments`,
      });
    }
```

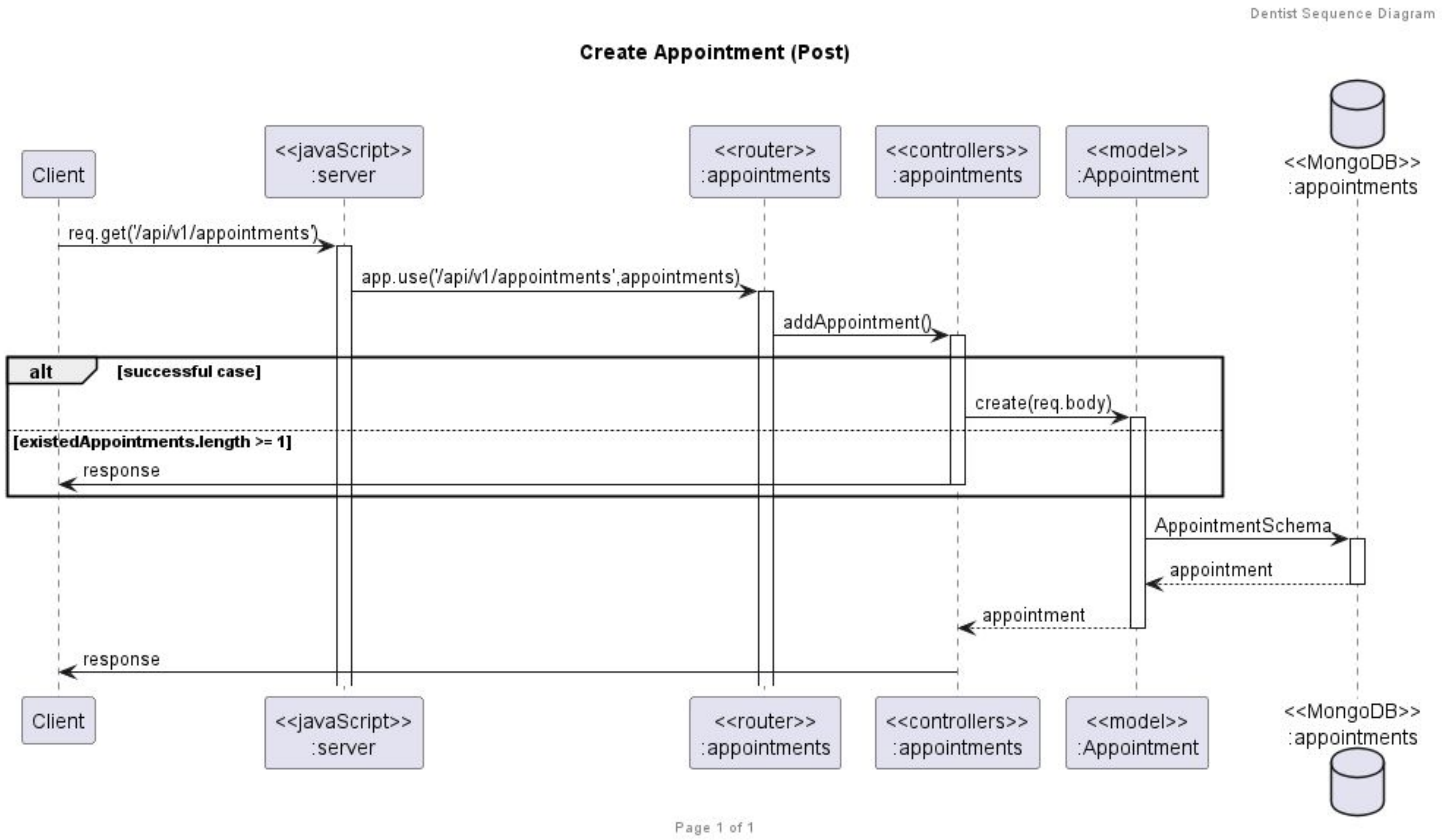
Use Case Diagram



Class Diagram

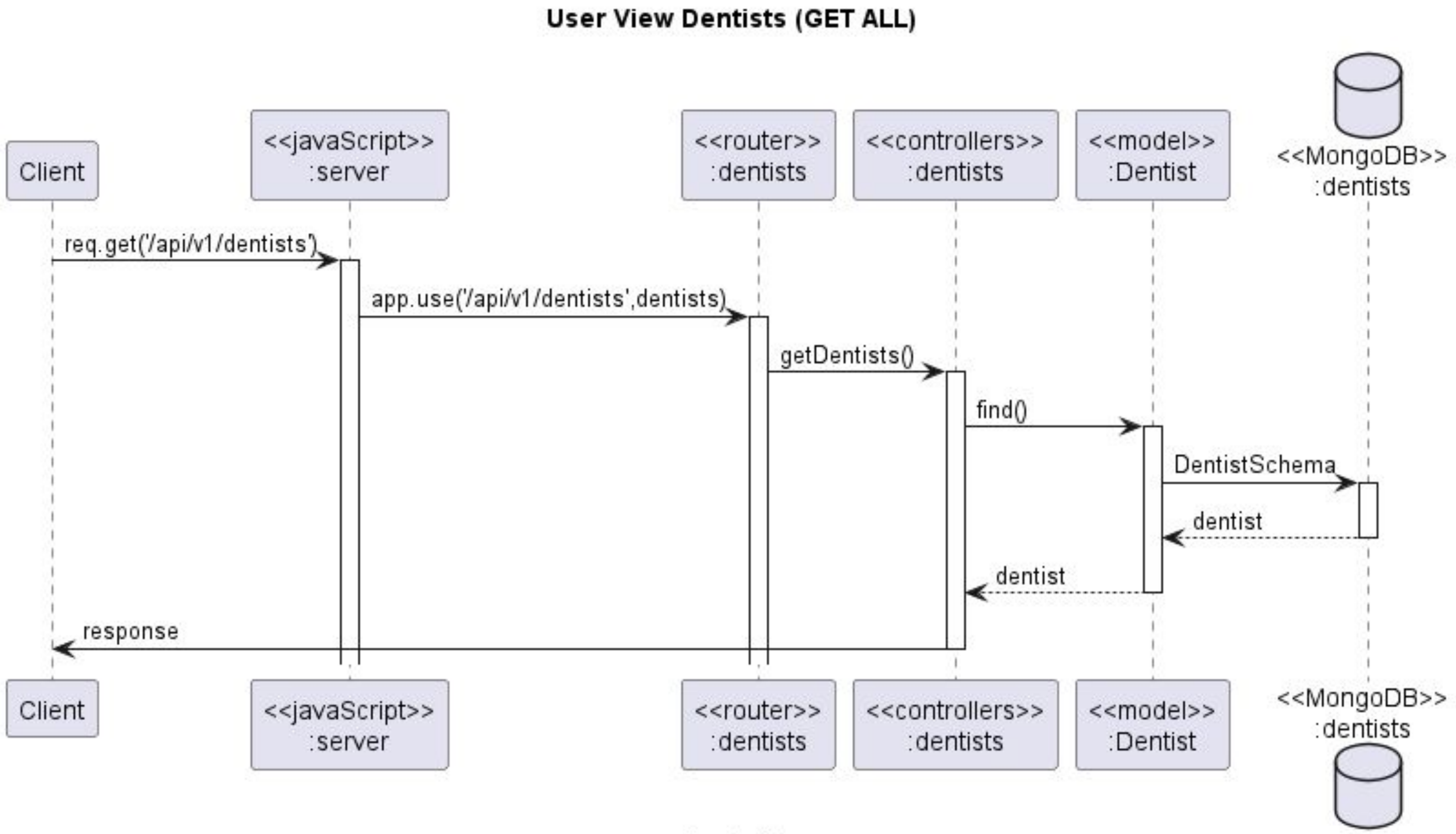


Create Appointment

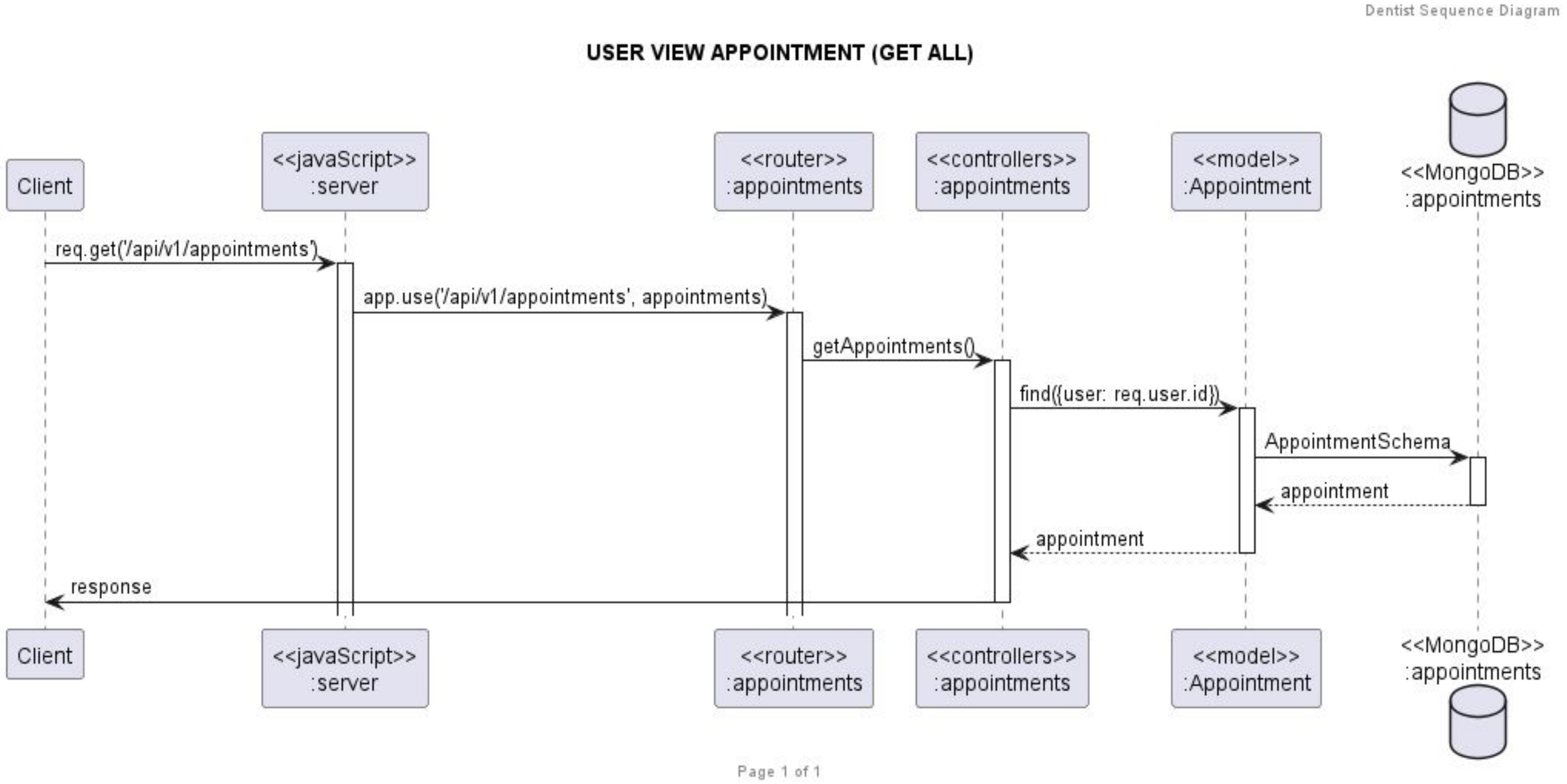


User View dentist

Dentist Sequence Diagram

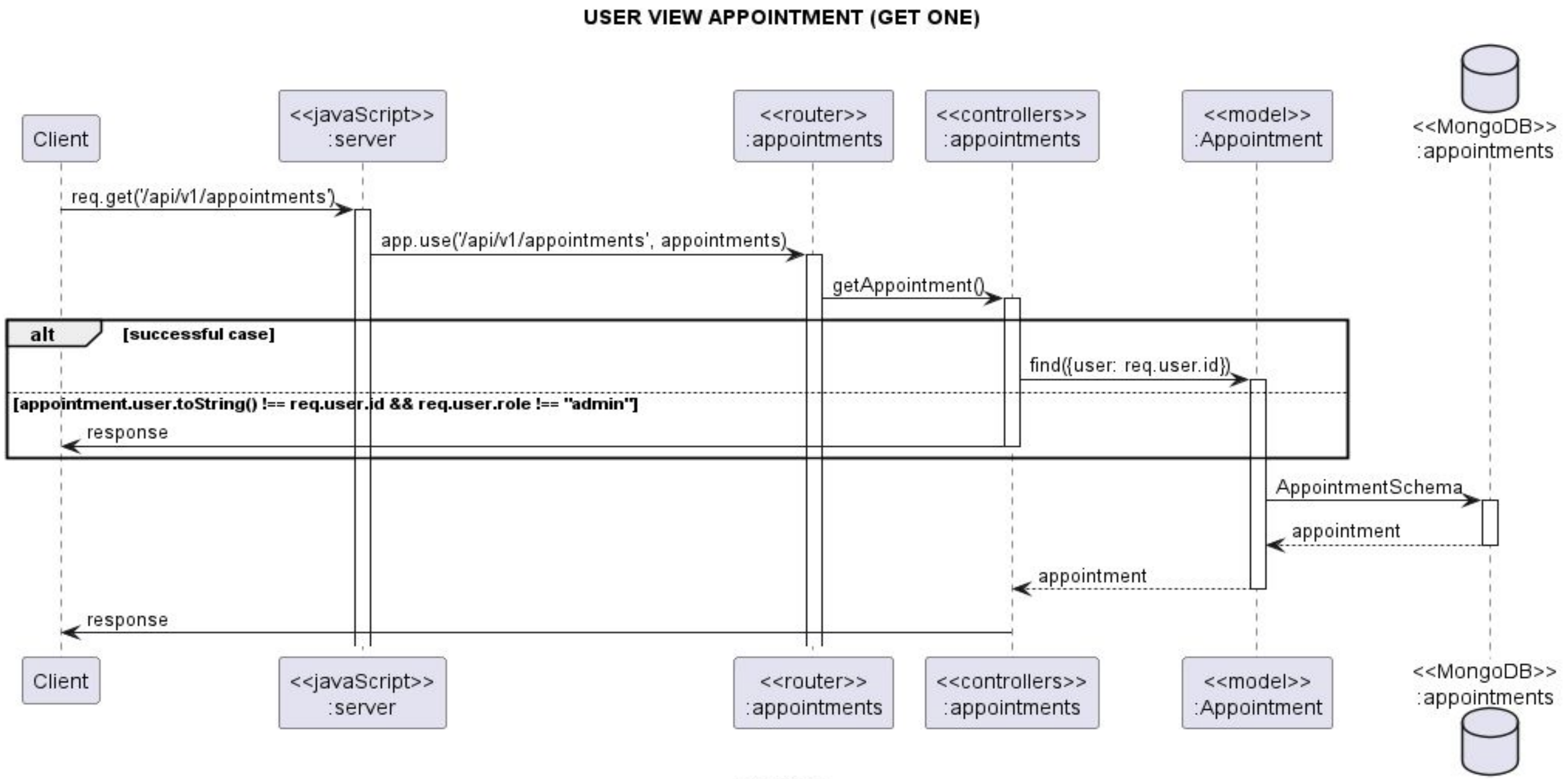


User view Appointment

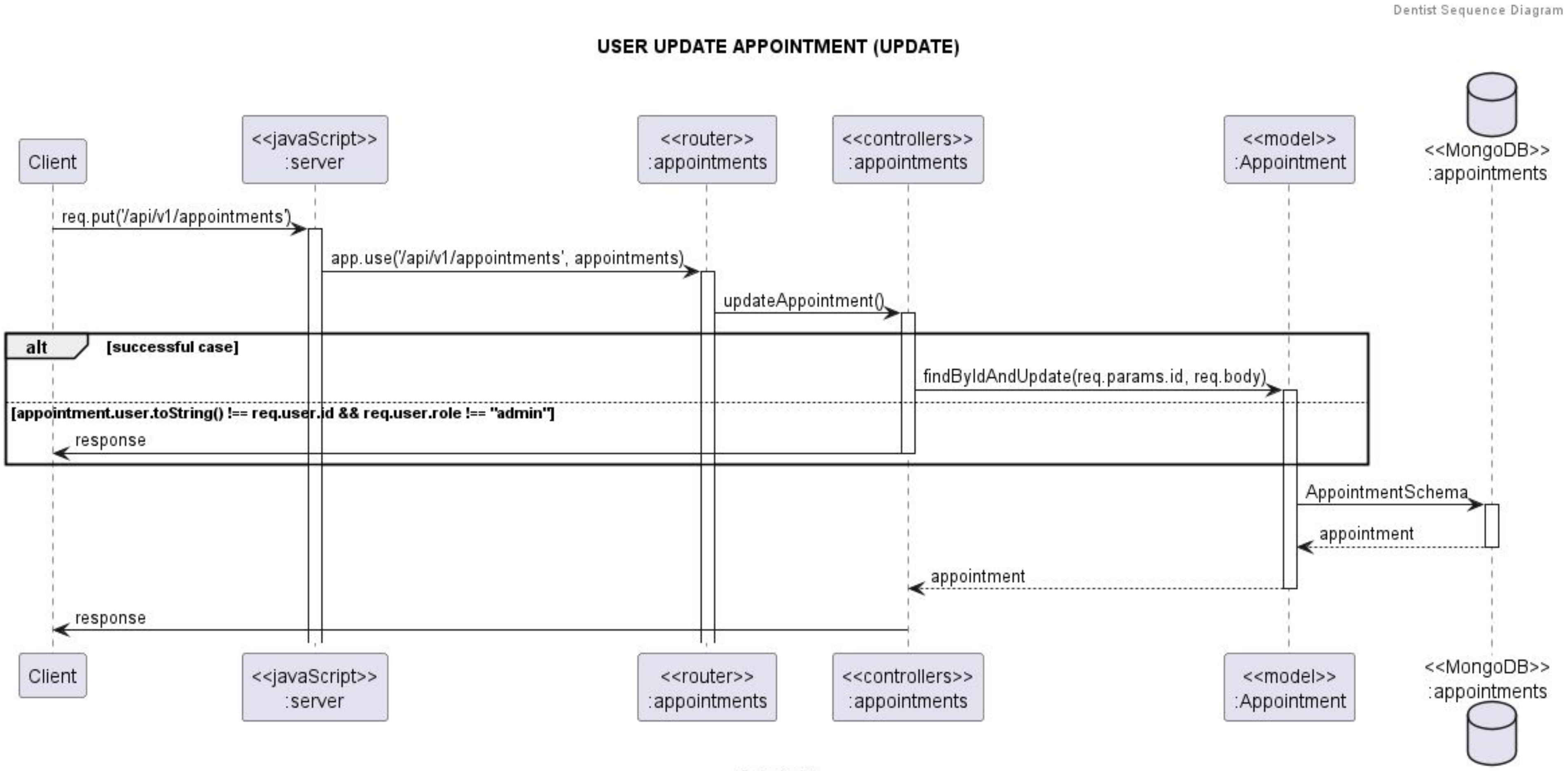

Dentist Sequence Diagram
USER VIEW APPOINTMENT (GET ALL)
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments
req.get('/api/v1/appointments')
app.use('/api/v1/appointments', appointments)
getAppointments()
find({user: req.user.id})
AppointmentSchema
appointment
appointment
response
Page 1 of 1

User view single Appointment

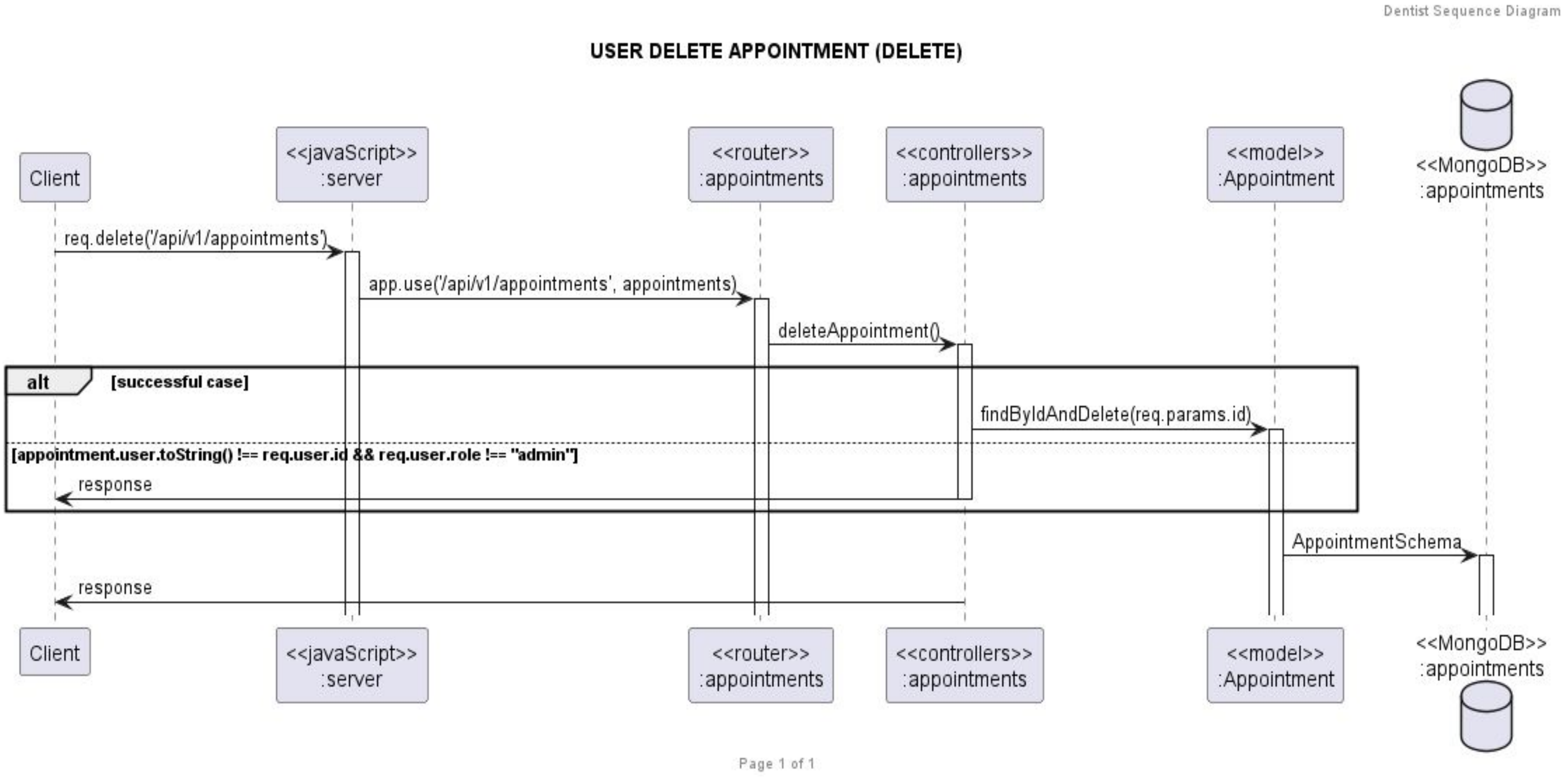
Dentist Sequence Diagram


USER VIEW APPOINTMENT (GET ONE)
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments
req.get('/api/v1/appointments')
app.use('/api/v1/appointments', appointments)
getAppointment()
alt
[successful case]
find({user: req.user.id})
[appointment.user.toString() !== req.user.id && req.user.role !== "admin"]
response
AppointmentSchema
appointment
appointment
response

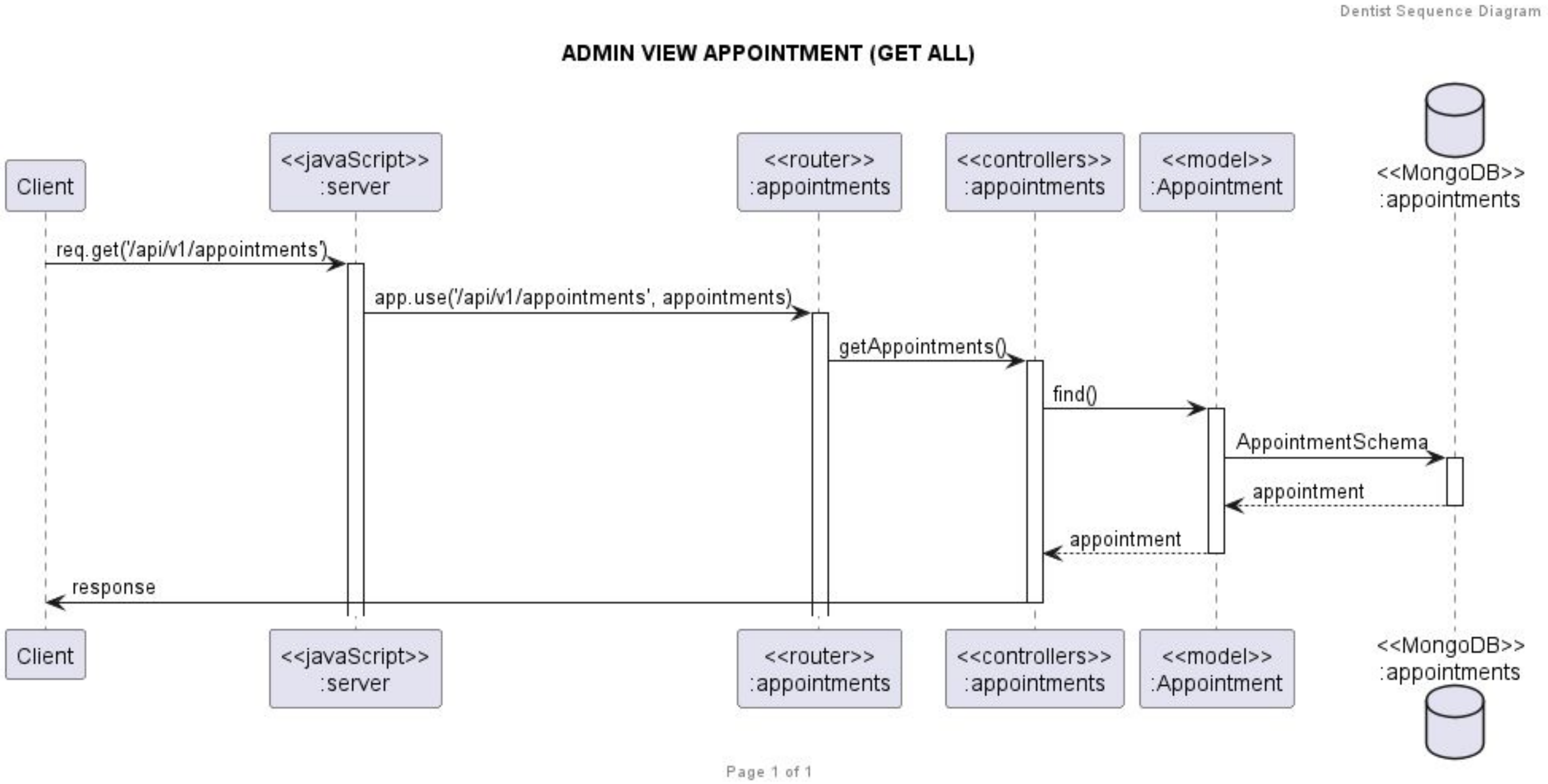
User update Appointment


Dentist Sequence Diagram
USER UPDATE APPOINTMENT (UPDATE)
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments
req.put('/api/v1/appointments')
app.use('/api/v1/appointments', appointments)
updateAppointment()
alt
[successful case]
findByIdAndUpdate(req.params.id, req.body)
[appointment.user.toString() !== req.user.id && req.user.role !== "admin"]
response
AppointmentSchema
appointment
appointment
response
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments

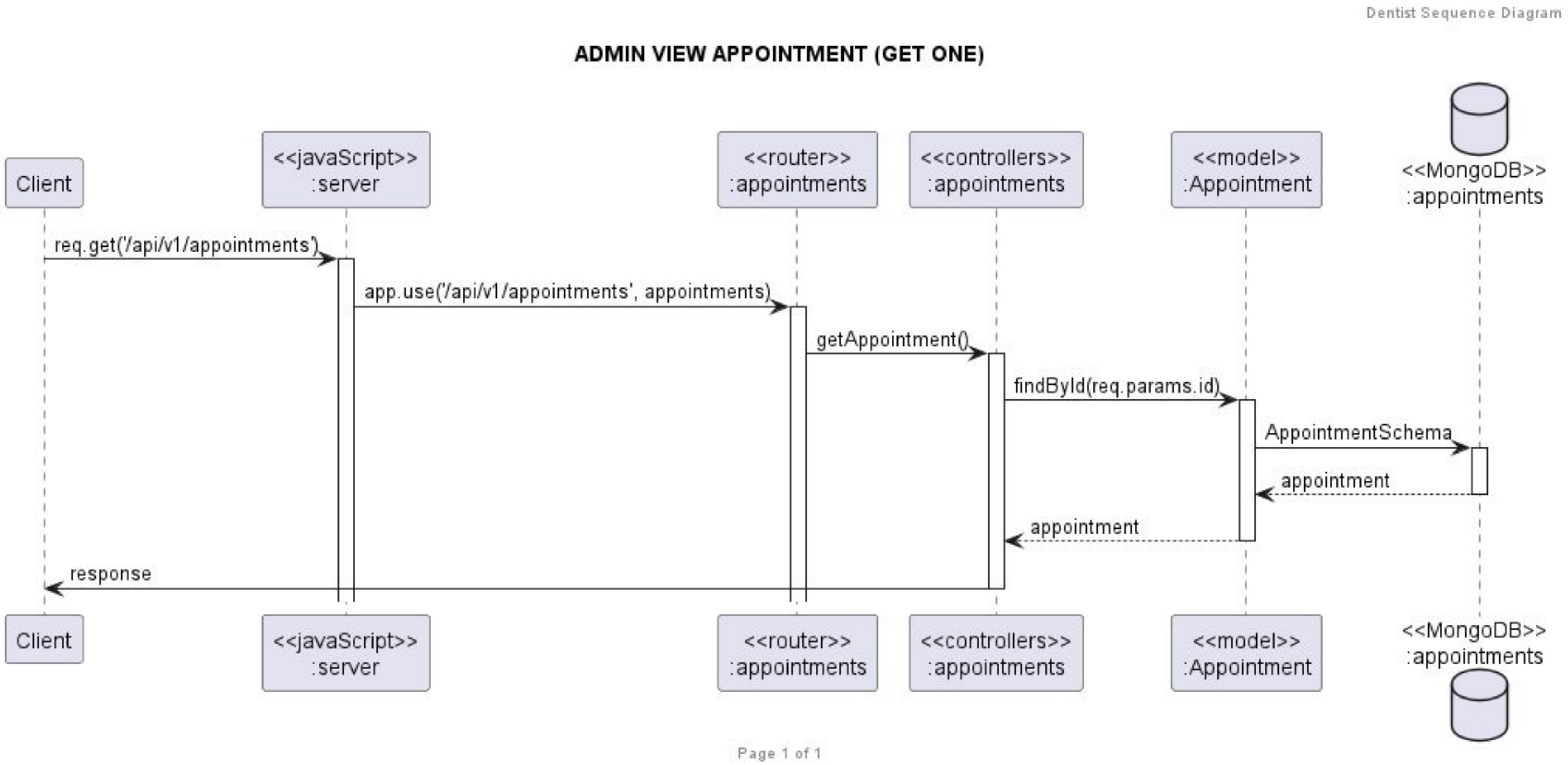
User delete Appointment


Dentist Sequence Diagram
USER DELETE APPOINTMENT (DELETE)
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments
req.delete('/api/v1/appointments')
app.use('/api/v1/appointments', appointments)
deleteAppointment()
alt
[successful case]
findByIdAndDelete(req.params.id)
[appointment.user.toString() !== req.user.id && req.user.role !== "admin"]
response
AppointmentSchema
response
Page 1 of 1

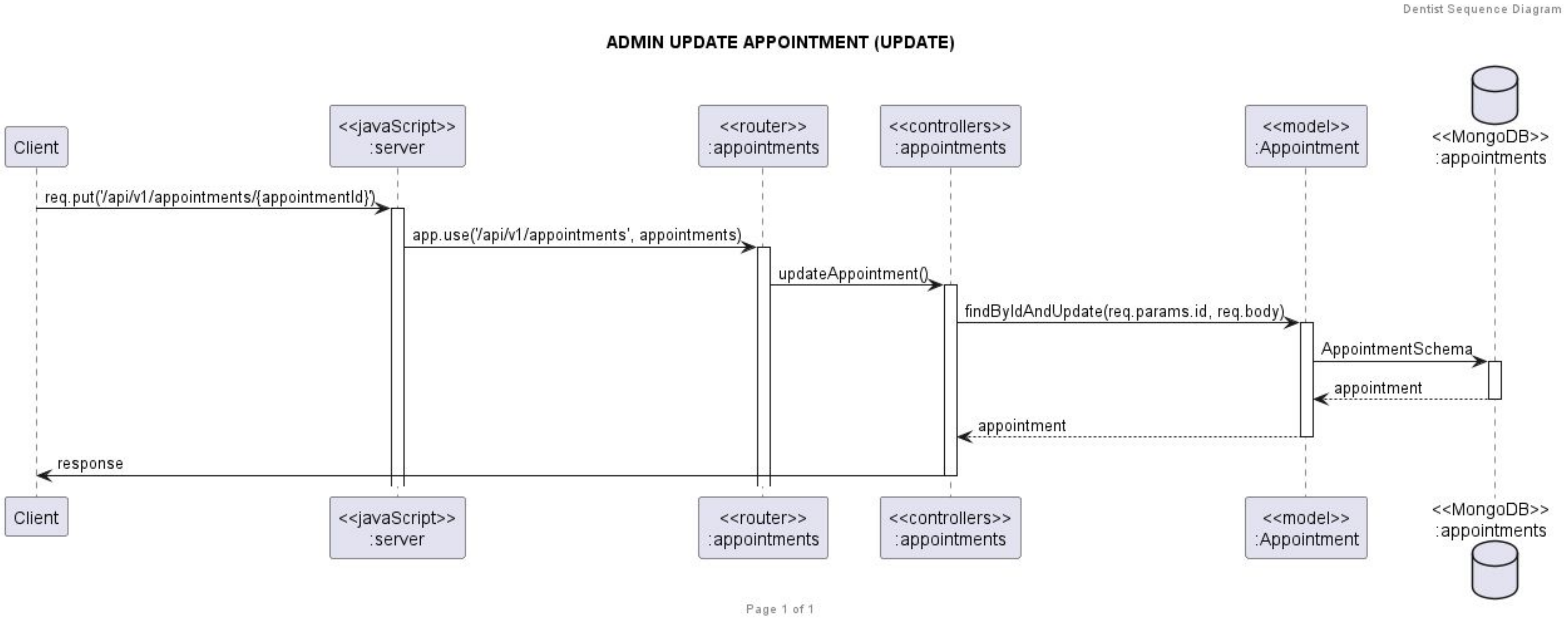
Admin view Appointment



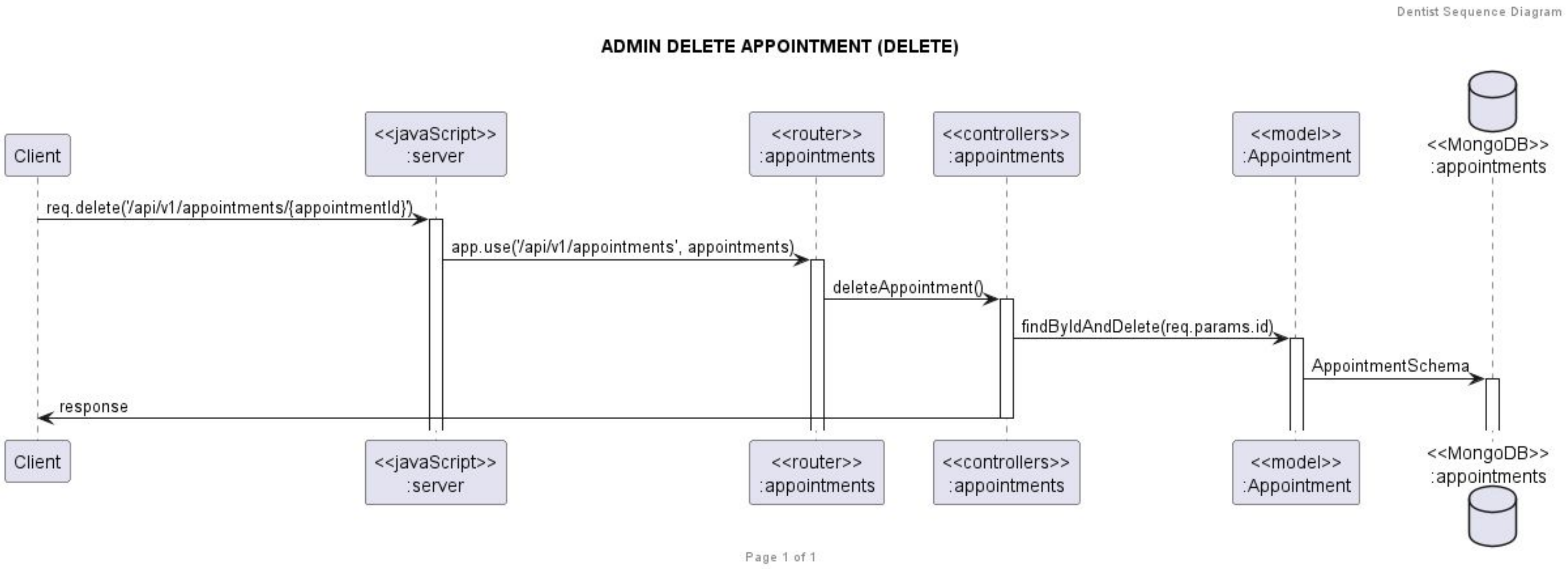
Admin view single Appointment



Admin update Appointment


Dentist Sequence Diagram
ADMIN UPDATE APPOINTMENT (UPDATE)
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments
req.put('/api/v1/appointments/{appointmentId}')
app.use('/api/v1/appointments', appointments)
updateAppointment()
findByIdAndUpdate(req.params.id, req.body)
AppointmentSchema
appointment
appointment
response
Page 1 of 1

Admin delete Appointment



Dentist Sequence Diagram
ADMIN DELETE APPOINTMENT (DELETE)
Client
<<javaScript>> :server
<<router>> :appointments
<<controllers>> :appointments
<<model>> :Appointment
<<MongoDB>> :appointments
req.delete('/api/v1/appointments/{appointmentId}')
app.use('/api/v1/appointments', appointments)
deleteAppointment()
findByIdAndDelete(req.params.id)
AppointmentSchema
response
Page 1 of 1

การแบ่งงาน

User - ปฤณสิษฐ์

Appointment - ธีรุตม์

Dentist - ภัทรดนัย

Use Case Diagram - ภัทรดนัย/ธีรุตม์/ปฤณสิษฐ์

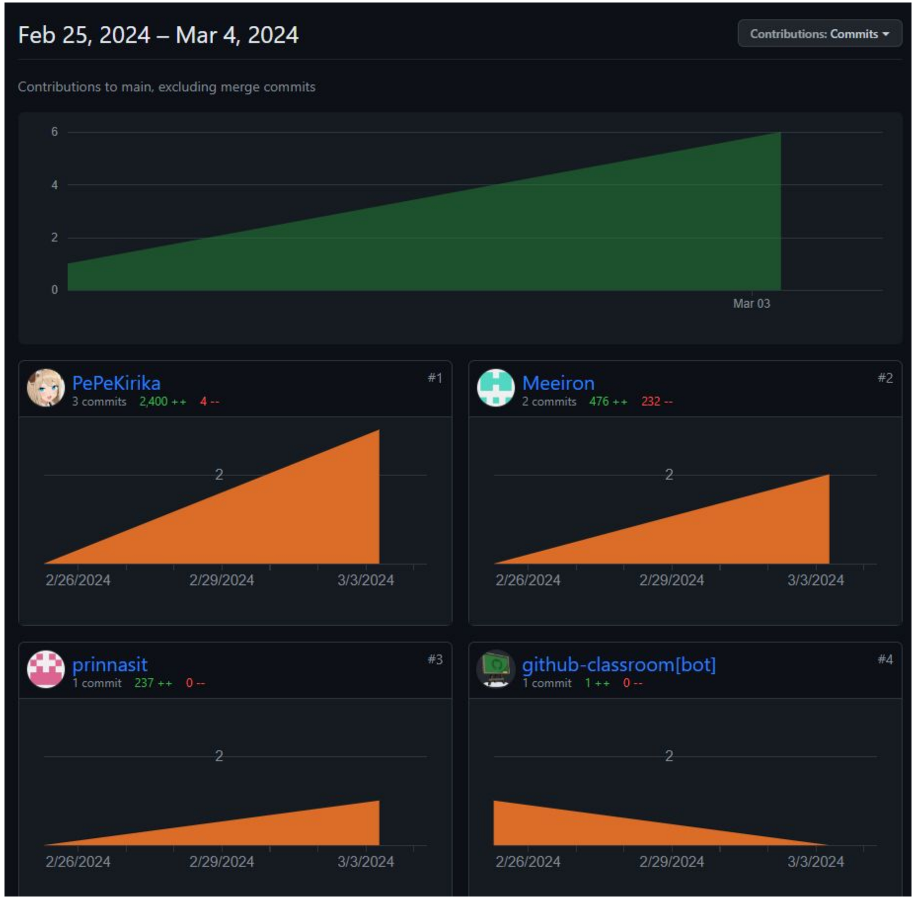
Class Diagram - ภัทรดนัย/ธีรุตม์/ปฤณสิษฐ์

Sequence Diagram - ภัทรดนัย/ธีรุตม์/ปฤณสิษฐ์

Presentation- ภัทรดนัย/ธีรุตม์/ปฤณสิษฐ์

Runner- ภัทรดนัย

การแบ่งงาน


Feb 25, 2024 – Mar 4, 2024

Contributions: Commits

Contributions to main, excluding merge commits

PePeKirika #1
3 commits 2,400 ++ 4 --

Meeiron #2
2 commits 476 ++ 232 --

prinnasit #3
1 commit 237 ++ 0 --

github-classroom[bot] #4
1 commit 1 ++ 0 --

https://github.com/2110503TACEDT66/presentation-day-1-this-is-my-kingdom-come/graphs/contributors