may i scan

MASSAGE SHOP

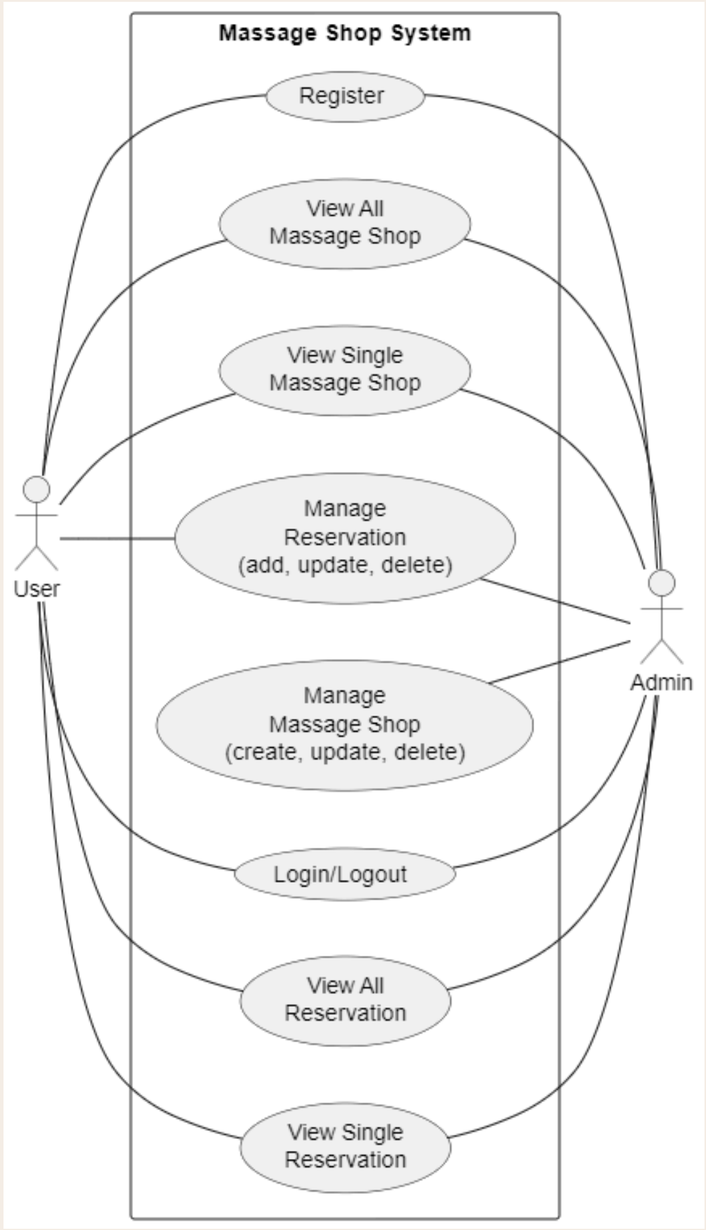
Software Development Practice

GIVEN PROBLEM

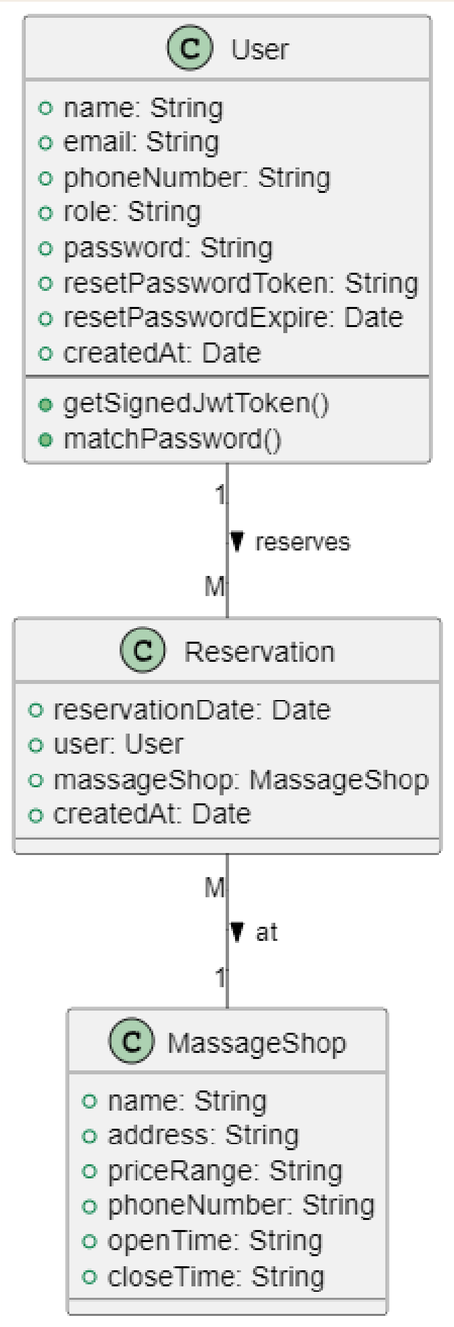
Massage Shop

1. The system shall allow a user to register by specifying the name, telephone number, email, and password.
2. After registration, the user becomes a registered user, and the system shall allow the user to log in to use the system by specifying the email and password. The system shall allow a registered user to log out.
3. After login, the system shall allow the registered user to reserve up to 3 queues by specifying the date and the preferred massage shop. The massage shop list is also provided to the user. A massage shop information includes the name, address, telephone number, and open-close time.
4. The system shall allow the registered user to view his/her massage reservation.
5. The system shall allow the registered user to edit his/her massage reservation.
6. The system shall allow the registered user to delete his/her massage reservation.
7. The system shall allow the admin to view any massage reservation.
8. The system shall allow the admin to edit any massage reservation.
9. The system shall allow the admin to delete any massage reservation.

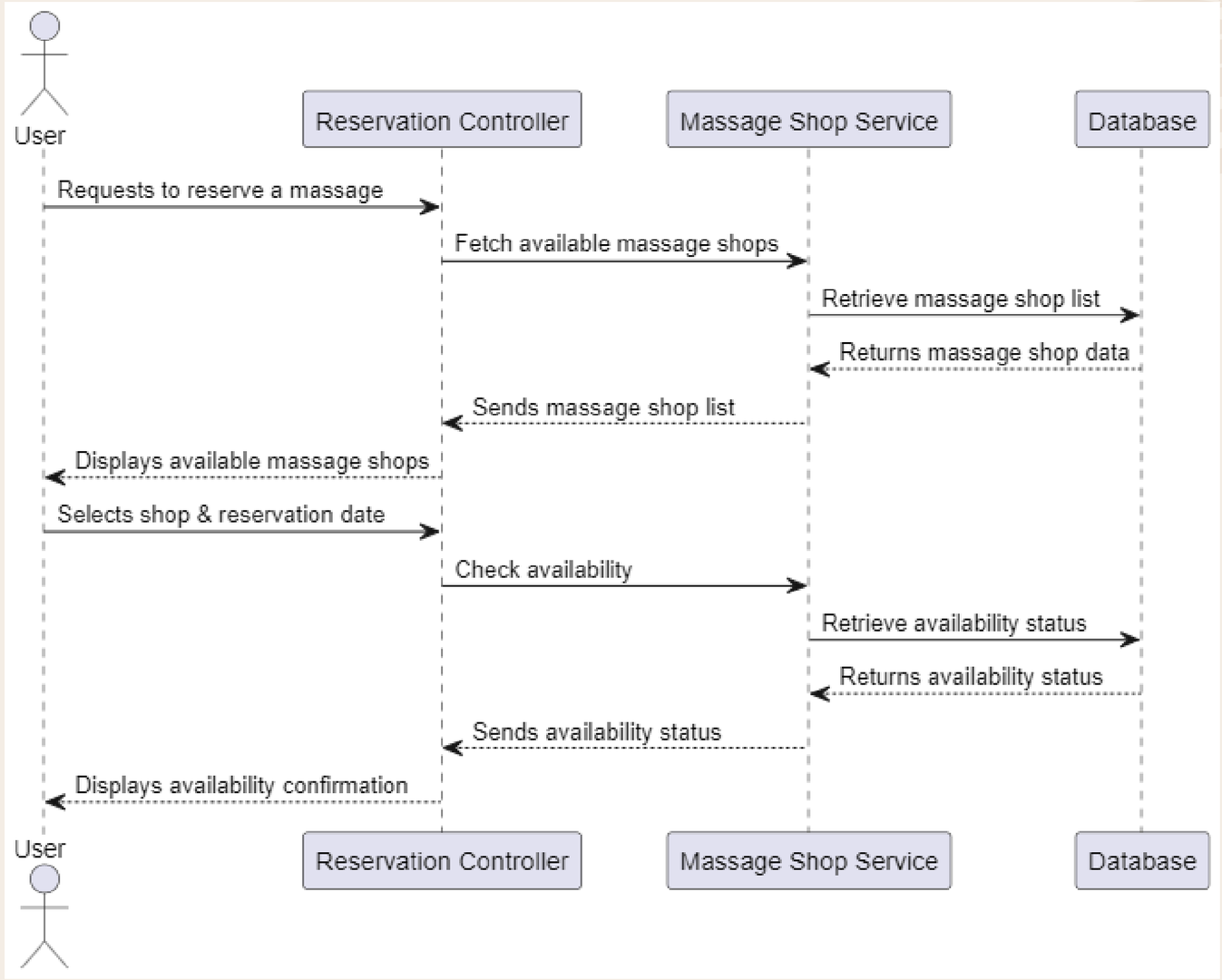
USE CASE DIAGRAM



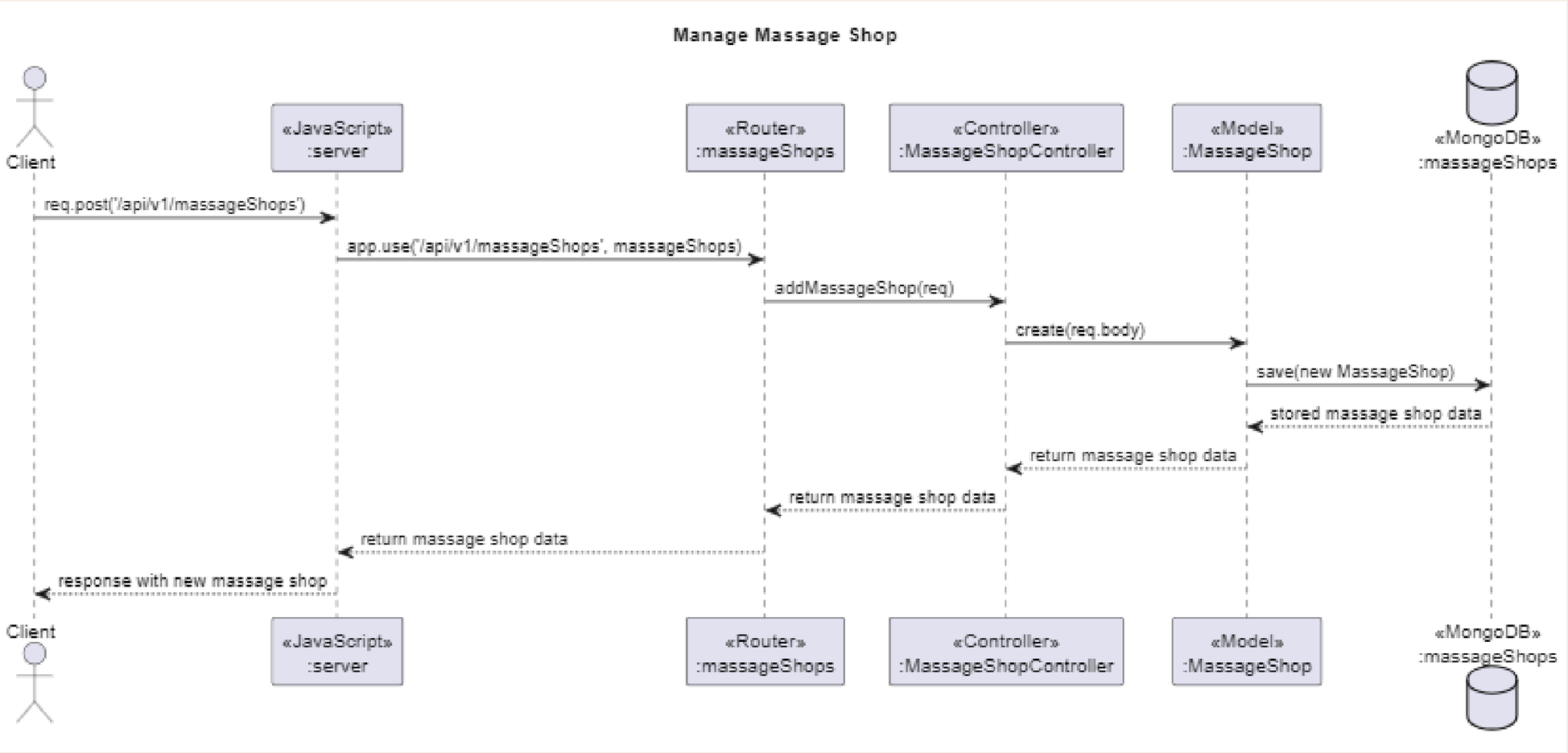
CLASS DIAGRAM



SEQUENCE DIAGRAM RESERVATION CONTROLLER



SEQUENCE DIAGRAM CONTROLLER



add phoneNumber in models/User

```
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@"]+
            'Please add a valid email'
        ]
    },
    phoneNumber: {
        type: String,
        required: [true, 'Please add a phone number'],
        unique: true
    },
    role: {
        type: String,
        enum: ['user', 'admin'],
        default: 'user'
    },
    password: {
        type: String,
        required: [true, 'Please add a password'],
        minlength: 6,
        select: false
    },
    resetPasswordToken: String,
    resetPasswordExpire: Date,
    createdAt: {
        type: Date,
        default: Date.now
    }
});
```

change schema of Hospital to MassageShop

```
const MassageShopSchema = new mongoose.Schema({

    name: {
        type: String,
        required: [true,'Please add a name'],
        unique: true,
        trim:true,
        maxlength:[50,'Name can not be more than 50 characters']
    },
    address: {
        type: String,
        required: [true, 'Please add an address']
    },
    priceRange: {
        type: String,
        required: [true, 'Please add a price range']
    },
    phoneNumber: {
        type: String
    },
    openTime: {
        type: String,
        required: [true, 'Please add opening time']
    },
    closeTime: {
        type: String,
        required: [true, 'Please add closing time']
    }
}, {
    toJSON: { virtuals: true },
    toObject: { virtuals: true }
});
```

edit schema of Appointment into Reservation

change ApptDate to reservationDate

change hospital to massageShop

```
const ReservationSchema = new mongoose.Schema({
    reservationDate: {
        type: Date,
        required: true
    },
    user: {
        type: mongoose.Schema.ObjectId,
        ref: 'User',
        required: true
    },
    massageShop: {
        type: mongoose.Schema.ObjectId,
        ref: 'MassageShop',
        required: true
    },
    createdAt: {
        type: Date,
        default: Date.now
    }
});
```

insert phoneNumber for Register user/admin

```
exports.register = async (req, res, next)=> {
    console.log(req.body);

    try {
        const {name, email, phoneNumber, password, role}=req.body;

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

change routes/hospitals to massageShops

```
const express = require('express');
const {
    getMassageShops,
    getMassageShop,
    createMassageShop,
    updateMassageShop,
    deleteMassageShop
} = require('../controllers/massageShops');

const reservationRouter = require('./reservations');

const router = express.Router();

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

router.use('/:massageShopId/reservations/', reservationRouter);
router.route('/').get(getMassageShops).post(protect, authorize('admin'),
createMassageShop);
router.route('/:id').get(getMassageShop).put(protect, authorize('admin'),
updateMassageShop).delete(protect, authorize('admin'), deleteMassageShop);

module.exports = router;
```

change routes/appointments to reservations

```
const express = require('express');
const {
    getReservations,
    getReservation,
    addReservation,
    updateReservation,
    deleteReservation
} = require('../controllers/reservations');

const router = express.Router({ mergeParams: true });

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

router.route('/')
    .get(protect, getReservations)
    .post(protect, authorize('admin', 'user'), addReservation);

router.route('/:id')
    .get(protect, getReservation)
    .put(protect, authorize('admin', 'user'), updateReservation)
    .delete(protect, authorize('admin', 'user'), deleteReservation);

module.exports = router;
```

change hospitals -> massageShops and appointments -> reservations in server.js

```
const auth = require('./routes/auth');
const reservations = require('./routes/reservations');
const massageShops = require('./routes/massageShops');

app.use('/api/v1/auth',auth);
app.use('/api/v1/reservations', reservations);
app.use('/api/v1/massageShops', massageShops);
```

add logout in controllers/auth.js

```
exports.logout = async (req, res, next) => {
    res.cookie('token', 'none', {
        expires: new Date(Date.now() + 10 * 1000),
        httpOnly: true
    });
    res.status(200).json({
        success: true,
        data: {}
    });
};
```

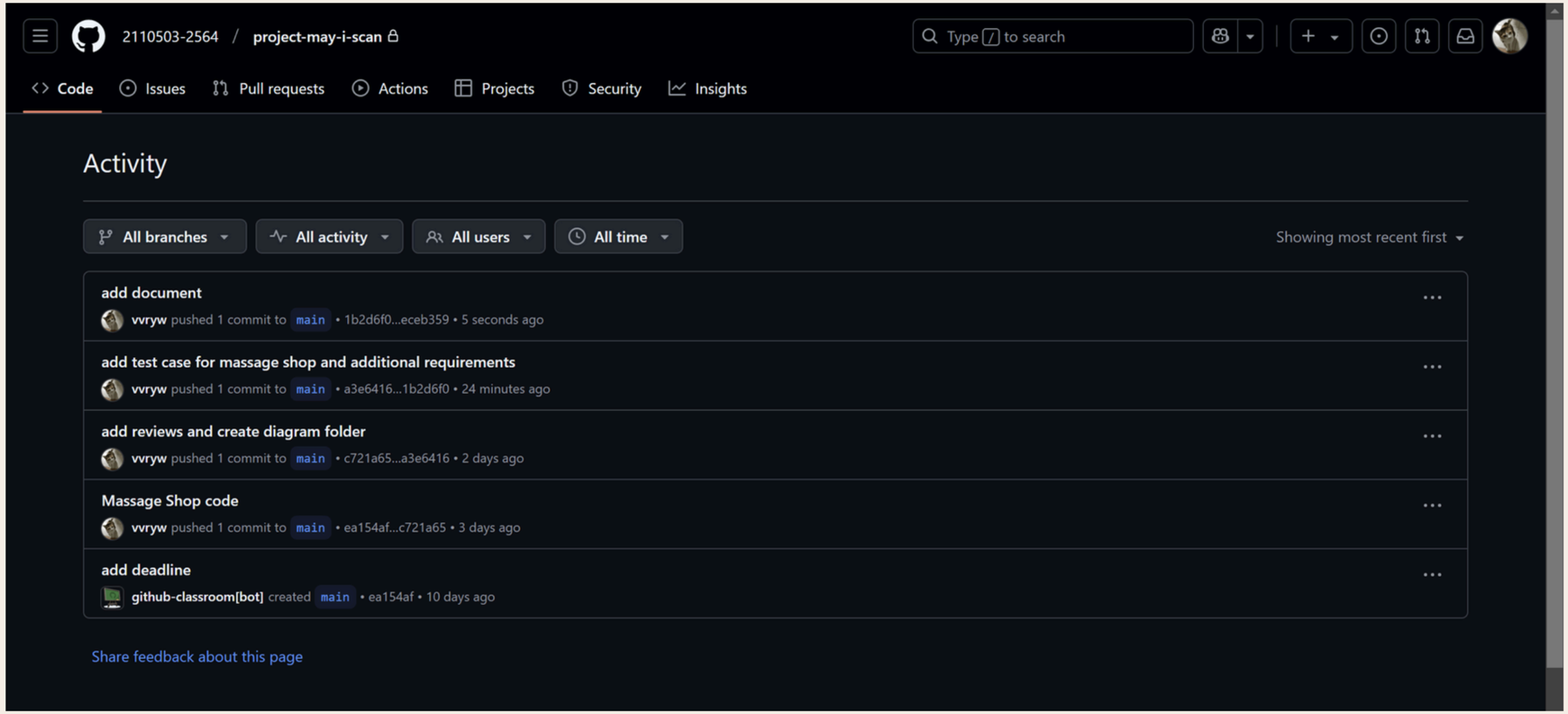
add logout in routes/auth.js

```
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;
```

LINK TO PROJECT GITHUB REPO

https://github.com/2110503-2564/project-may-i-scan

WORK DIVISION



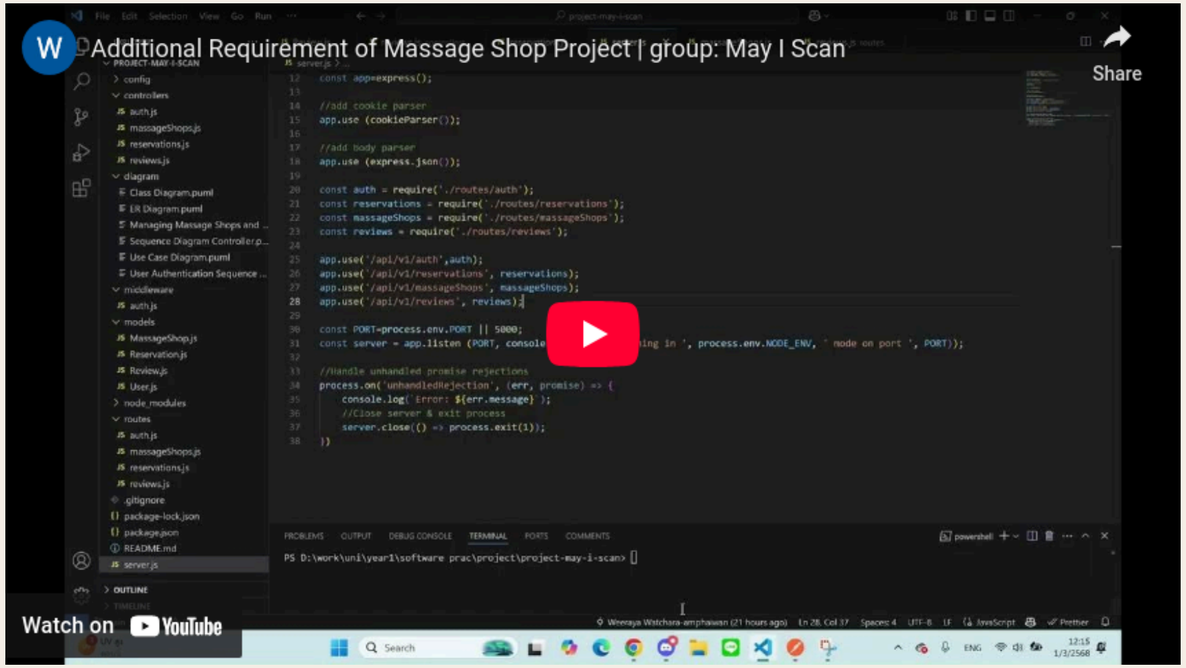
ADDITIONAL REQUIREMENT

- Create reviews for massage shop
- Create Models/Review.js
- Create controllers/reviews.js
    a. getReviews
    b. getReview
    c. addReview
    d. updateReview
    e. deleteReview
- Create routes/reviews.js
- Add route in server.js

ADDITIONAL REQUIREMENT

- link for additional requirement video

https://www.youtube.com/watch?v=ZsIONplmXJU



MEMBERS

- ไตรวิช อินตาโสภี 6733078321
- วีรยา วัชรอำไพวัณย์ 6733245821
- ไวทิน จิตติเรืองวิชัย 6733248721