



MASSAGE SHOP

Software Development Practice

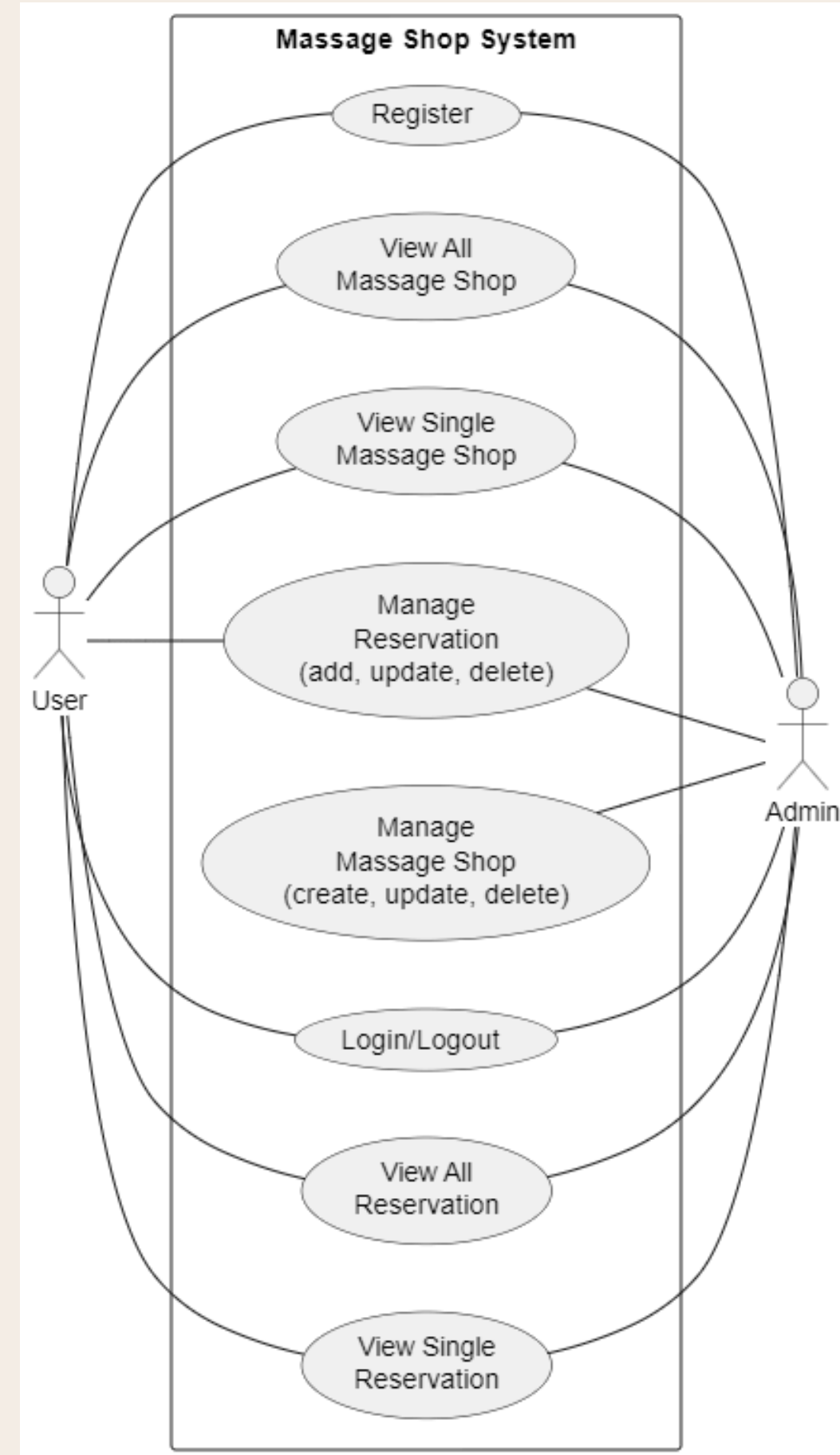


GIVEN PROBLEM

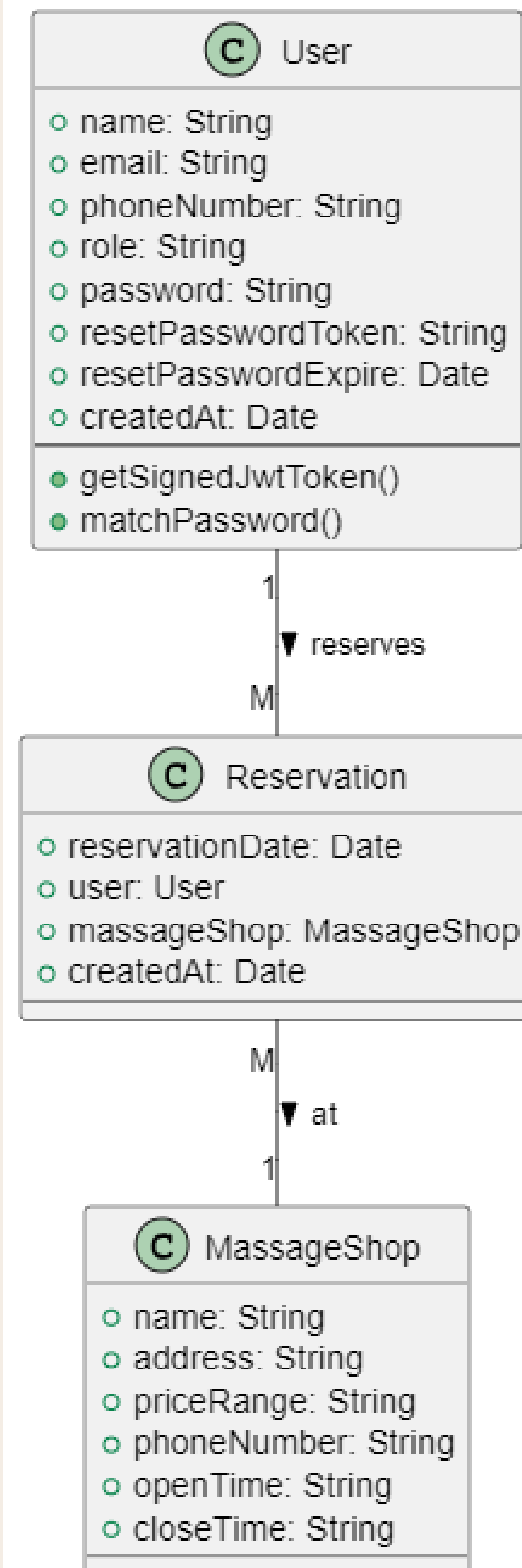
Massage Shop

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.

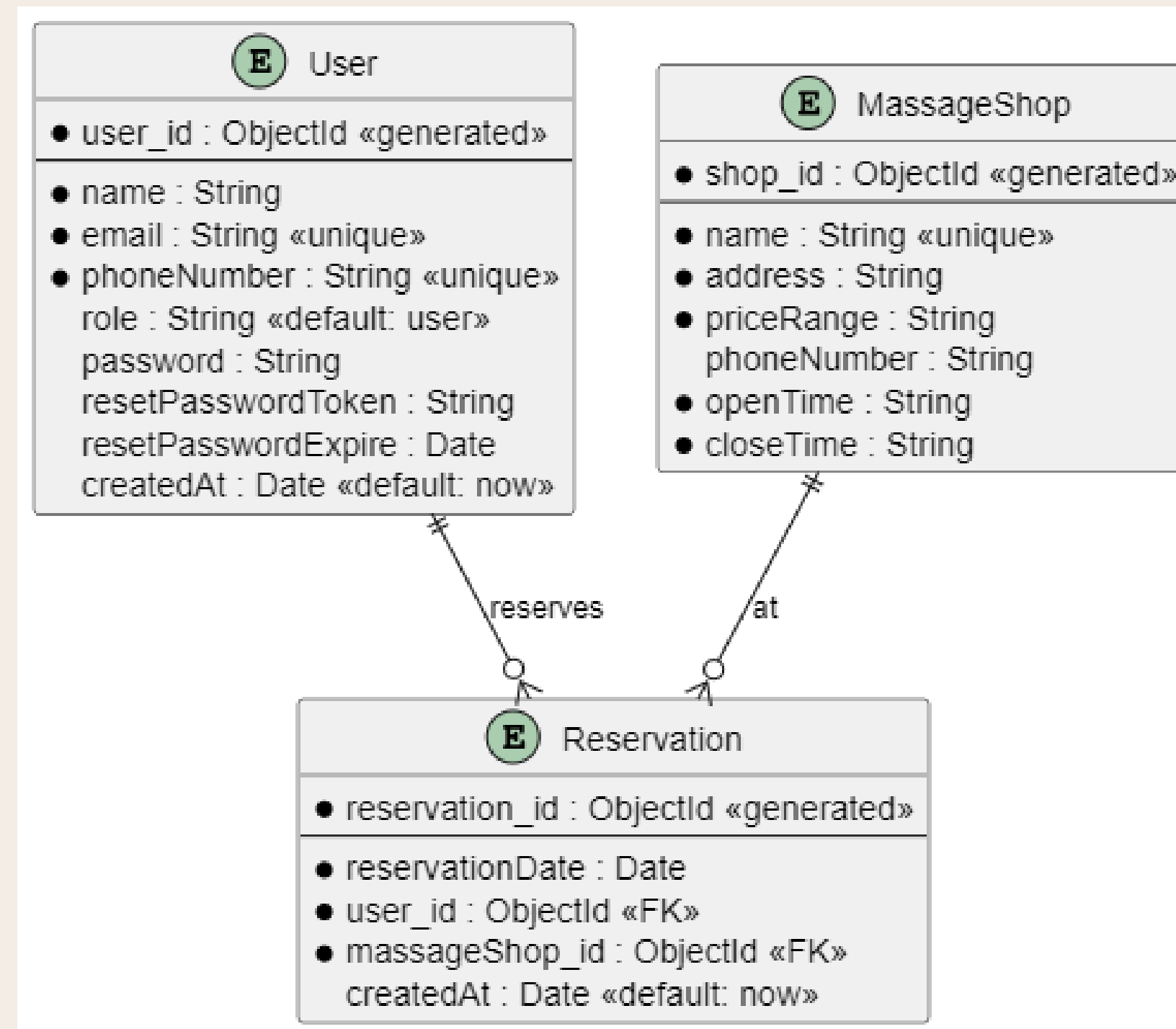
USE CASE DIAGRAM



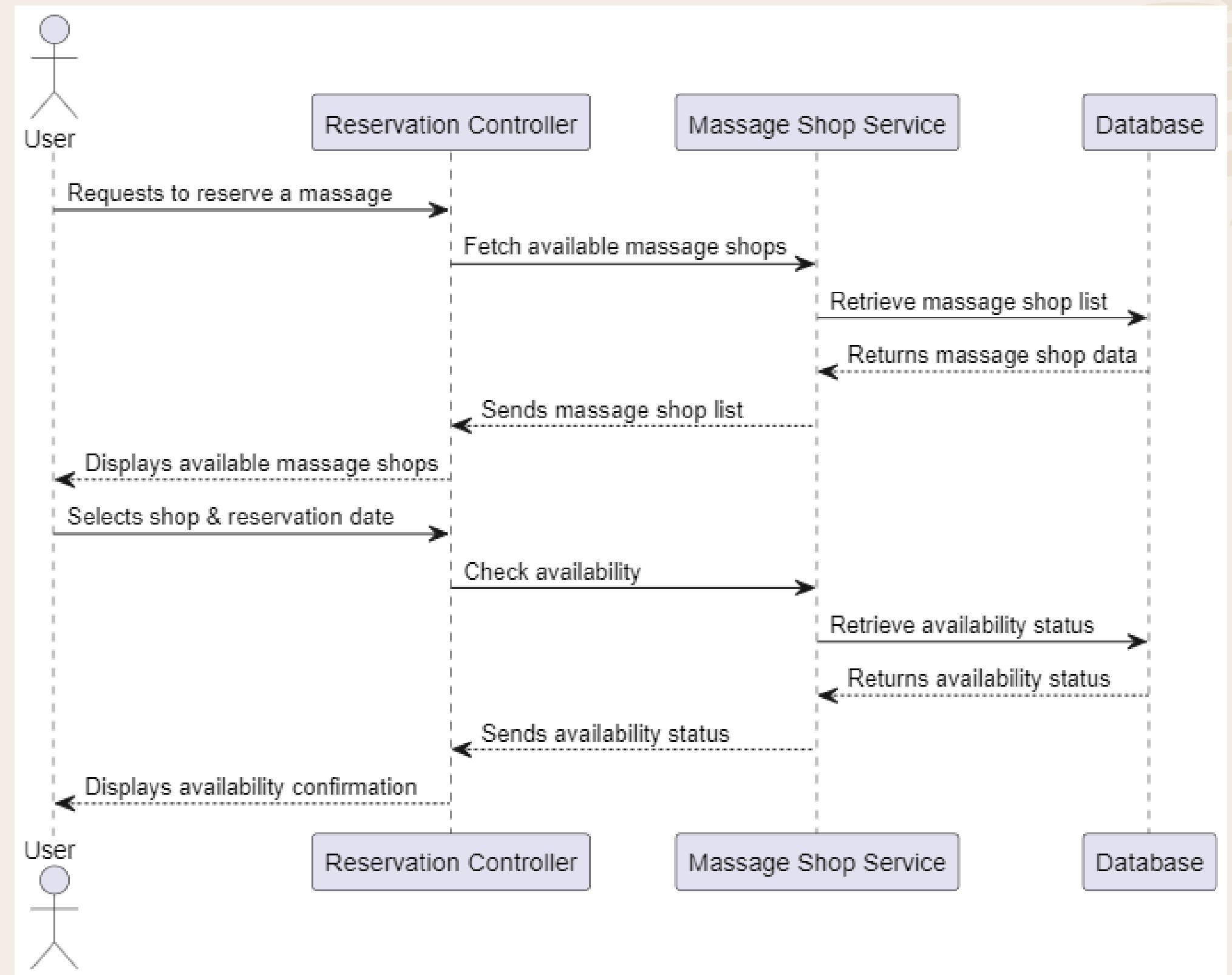
CLASS DIAGRAM



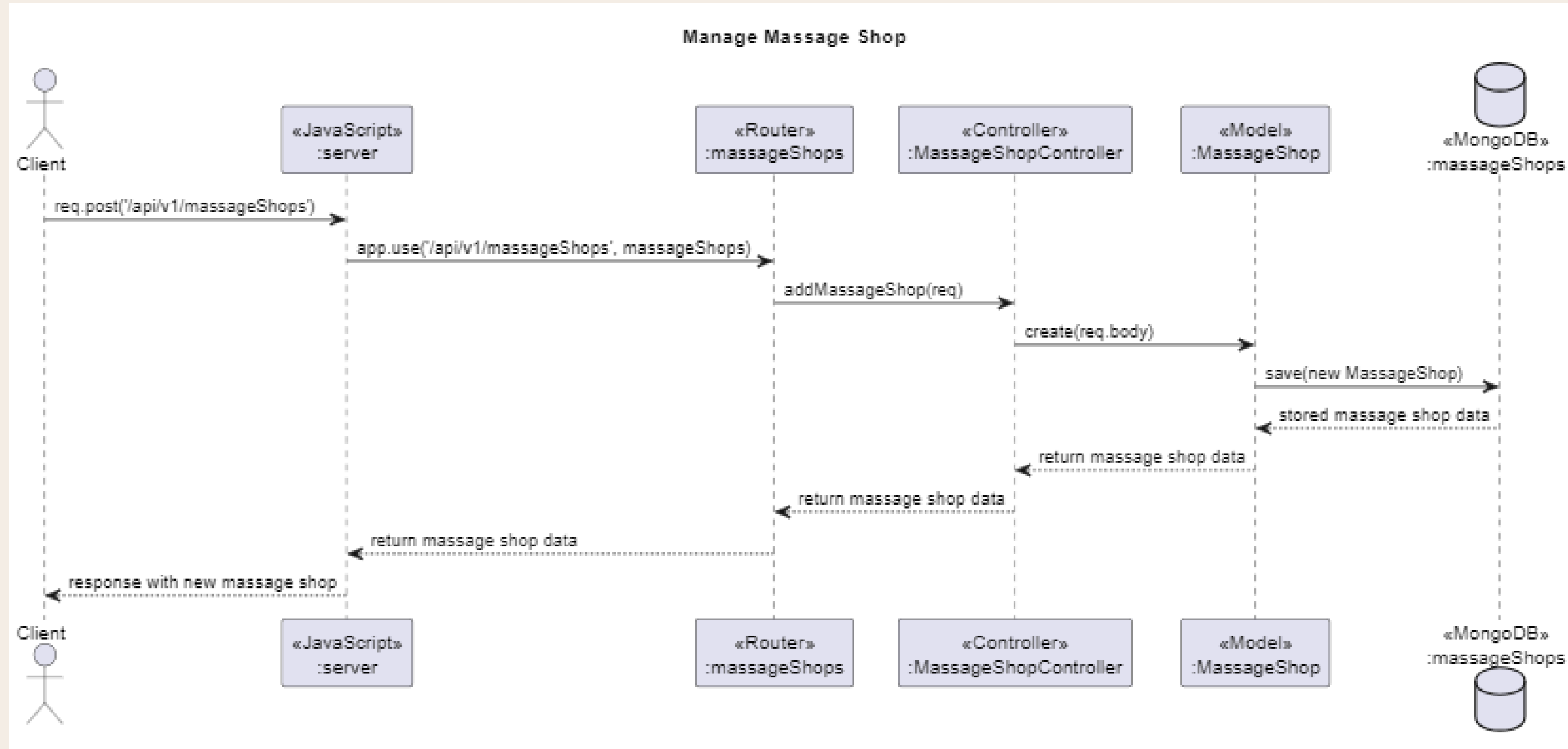
ER DIAGRAM



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
messageShop

```
const ReservationSchema = new mongoose.Schema({  
  reservationDate: {  
    type: Date,  
    required: true  
  },  
  user: {  
    type: mongoose.Schema.ObjectId,  
    ref: 'User',  
    required: true  
  },  
  messageShop: {  
    type: mongoose.Schema.ObjectId,  
    ref: 'MessageShop',  
    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 messageShops

```
const express = require('express');
const {
  getMessageShops,
  getMessageShop,
  createMessageShop,
  updateMessageShop,
  deleteMessageShop
} = require('../controllers/messageShops');

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

const router = express.Router();

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

router.use('/:messageShopId/reservations/', reservationRouter);
router.route('/').get(getMessageShops).post(protect, authorize('admin'),
createMessageShop);
router.route('/:id').get(getMessageShop).put(protect, authorize('admin'),
updateMessageShop).delete(protect, authorize('admin'), deleteMessageShop);

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 -> messageShops
and appointments -> reservations
in server.js

```
const auth = require('./routes/auth');  
const reservations = require('./routes/reservations');  
const messageShops = require('./routes/messageShops');  
  
app.use('/api/v1/auth', auth);  
app.use('/api/v1/reservations', reservations);  
app.use('/api/v1/messageShops', messageShops);
```

add logout in
controllers/auth.js

add logout in
routes/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: {}  
  });  
};
```

```
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



2110503-2564 / project-may-i-scan

Type / to search

<> Code

Issues

Pull requests

Actions

Projects

Security

Insights

Activity

All branches

All activity

All users

All time

Showing most recent first

add test case for message shop and additional requirements

...

vryw

pushed 1 commit to `main` • a3e6416...1b2d6f0 • 3 minutes ago

add reviews and create diagram folder

...

vryw

pushed 1 commit to `main` • c721a65...a3e6416 • 2 days ago

Message Shop code

...

vryw

pushed 1 commit to `main` • ea154af...c721a65 • 3 days ago

add deadline

...

github-classroom[bot]

created `main` • ea154af • 10 days ago

[Share feedback about this page](#)

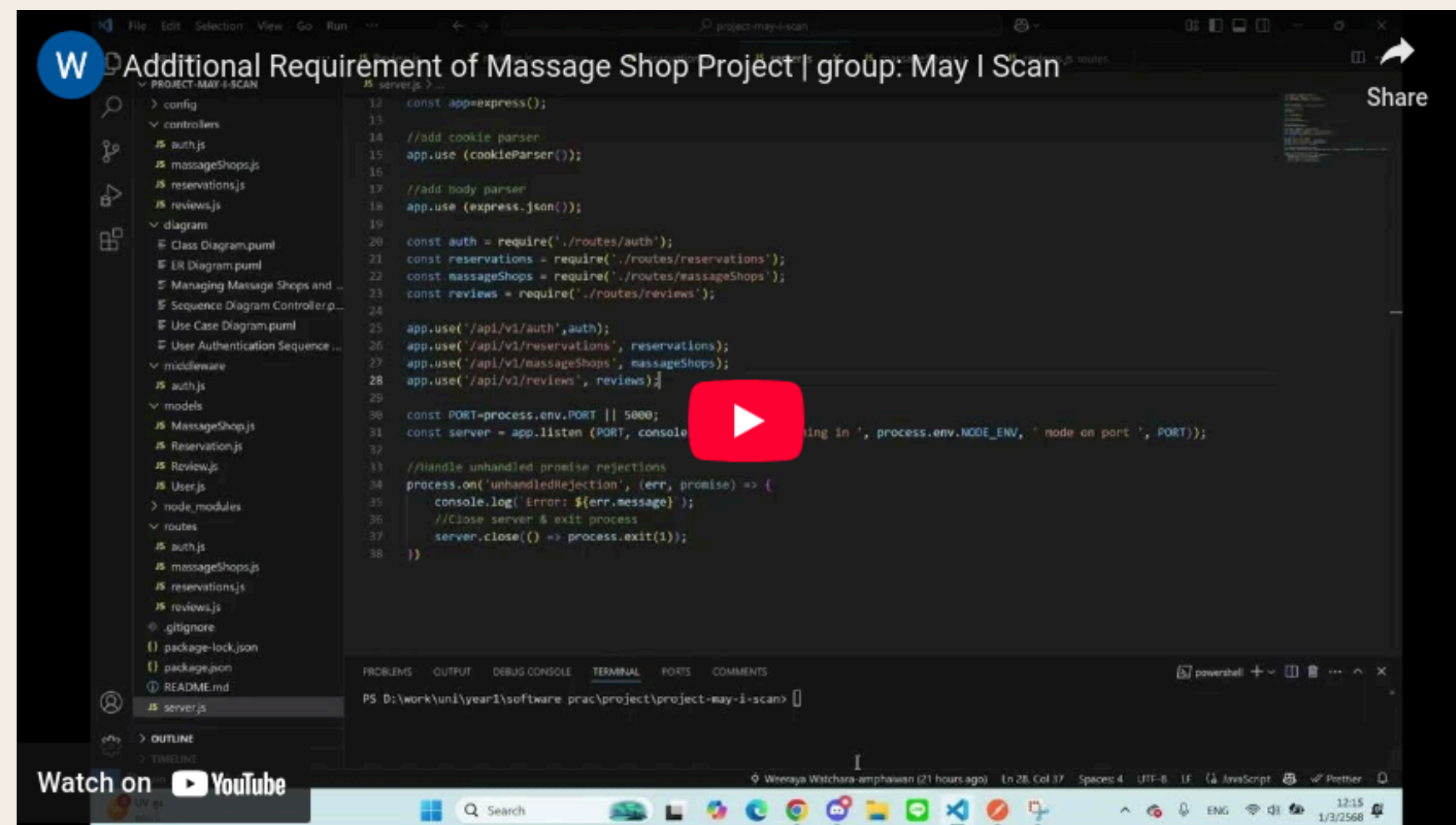
ADDITIONAL REQUIREMENT

- 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=ZsIONpImXJU>



MEMBERS

- ไตรวิช อินตาโสภี 6733078321
- วีรยา วัชรอำไพวัฒน์ 6733245821
- ไวกิน จิตติเรืองวิชัย 6733248721

