FINAL PROJECT PRESENTATION

Massage Reservation

The Team Introduction

Team 1

Suppawich Tawornpichayachai
6231364021

Team 2

Thanin Sawetkititham
6230234821

Non-Functional Requirements

Security

- The system shall authenticate users using username-password.
- The system shall be able to keep user’s transactions confidential.

Performance

The system shall response to a request in 3 seconds.

Usability

The system shall be used and test via Postman.

Functional Requirements

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 logout.

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.

* เพิ่มข้อกำหนดว่า massage shop เปิดทุกวัน
โดยเปิด-ปิด ในวันเดียวกันระหว่าง 00:00 - 23:59 *

Functional Requirements

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.

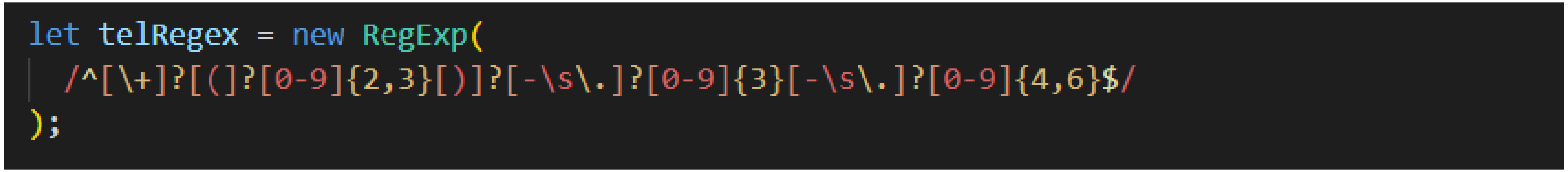
7 The system shall allow the admin to view any massage reservation.

8 The system shall allow the admin to edit any massage reservation.

9 The system shall allow the admin to delete any massage reservation.

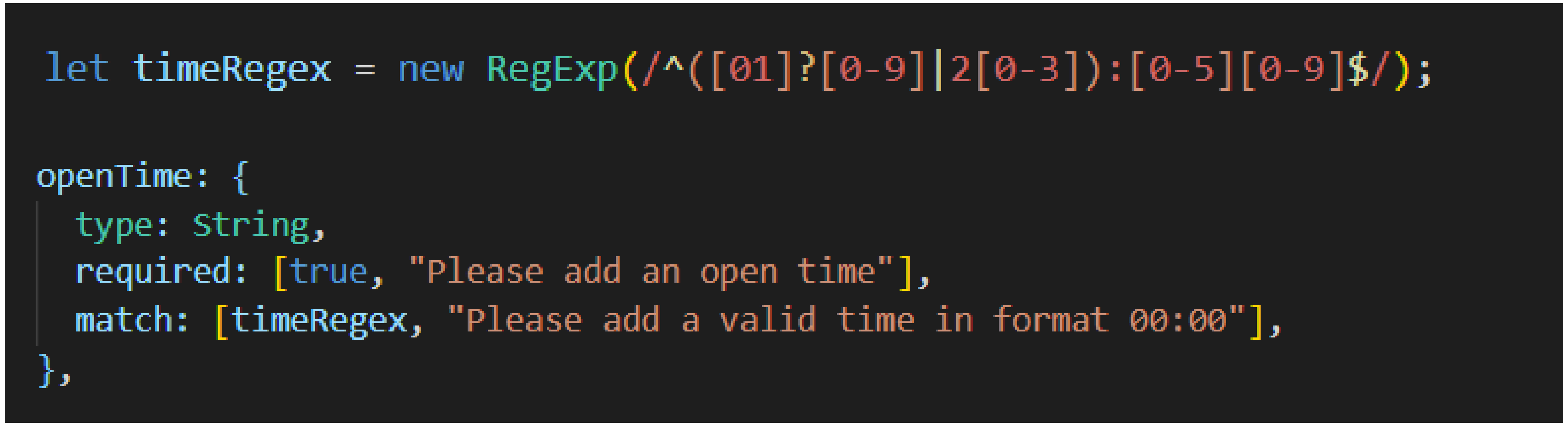
Code change

models/User



```
let telRegex = new RegExp(
  /^[\+]?[(]?[0-9]{2,3}[)]?[-\s\.]?[0-9]{3}[-\s\.]?[0-9]{4,6}$/
);
```

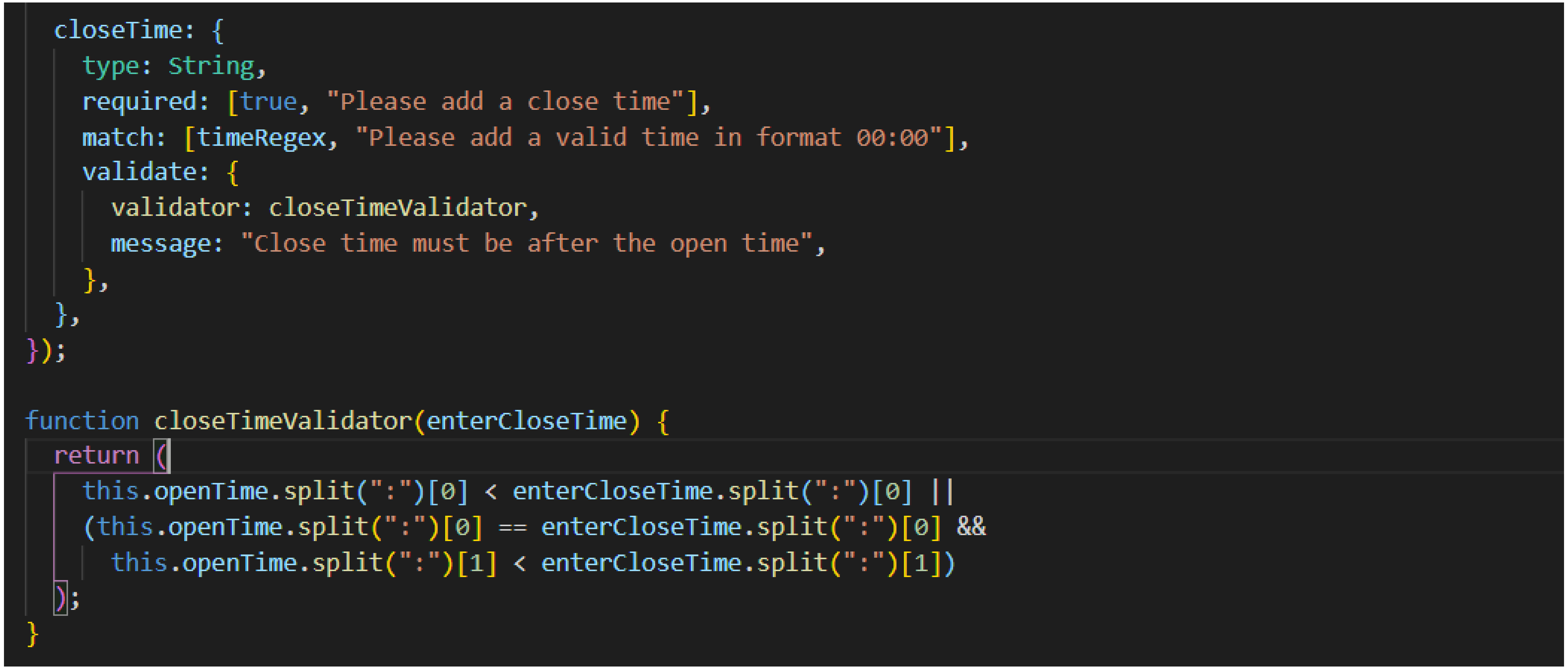
models/massageShop



```
 let timeRegex = new RegExp(/^([01]?[0-9]|2[0-3]):[0-5][0-9]$/);

openTime: {
  type: String,
  required: [true, "Please add an open time"],
  match: [timeRegex, "Please add a valid time in format 00:00"],
},
```

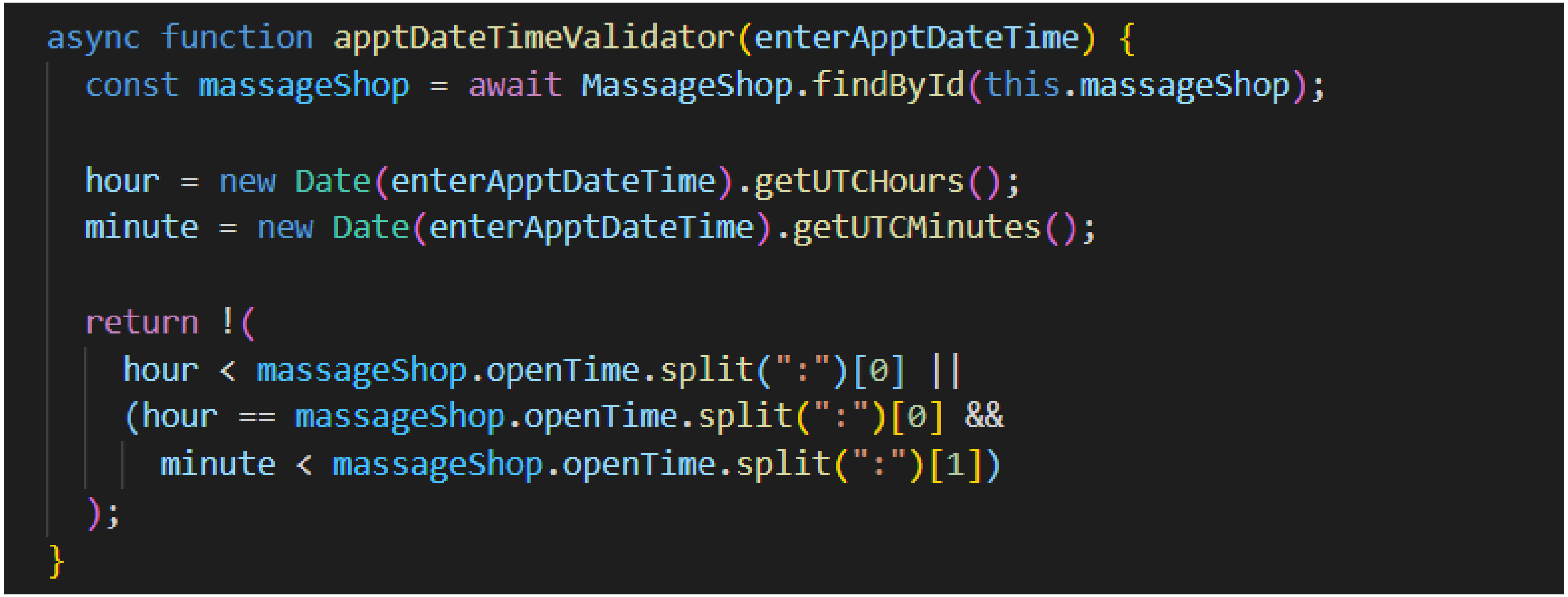
Code change

models/massageShop


```
  closeTime: {
    type: String,
    required: [true, "Please add a close time"],
    match: [timeRegex, "Please add a valid time in format 00:00"],
    validate: {
      validator: closeTimeValidator,
      message: "Close time must be after the open time",
    },
  },
});

function closeTimeValidator(enterCloseTime) {
  return (
    this.openTime.split(":")[0] < enterCloseTime.split(":")[0] ||
    (this.openTime.split(":")[0] == enterCloseTime.split(":")[0] &&
      this.openTime.split(":")[1] < enterCloseTime.split(":")[1])
  );
}
```

Code change

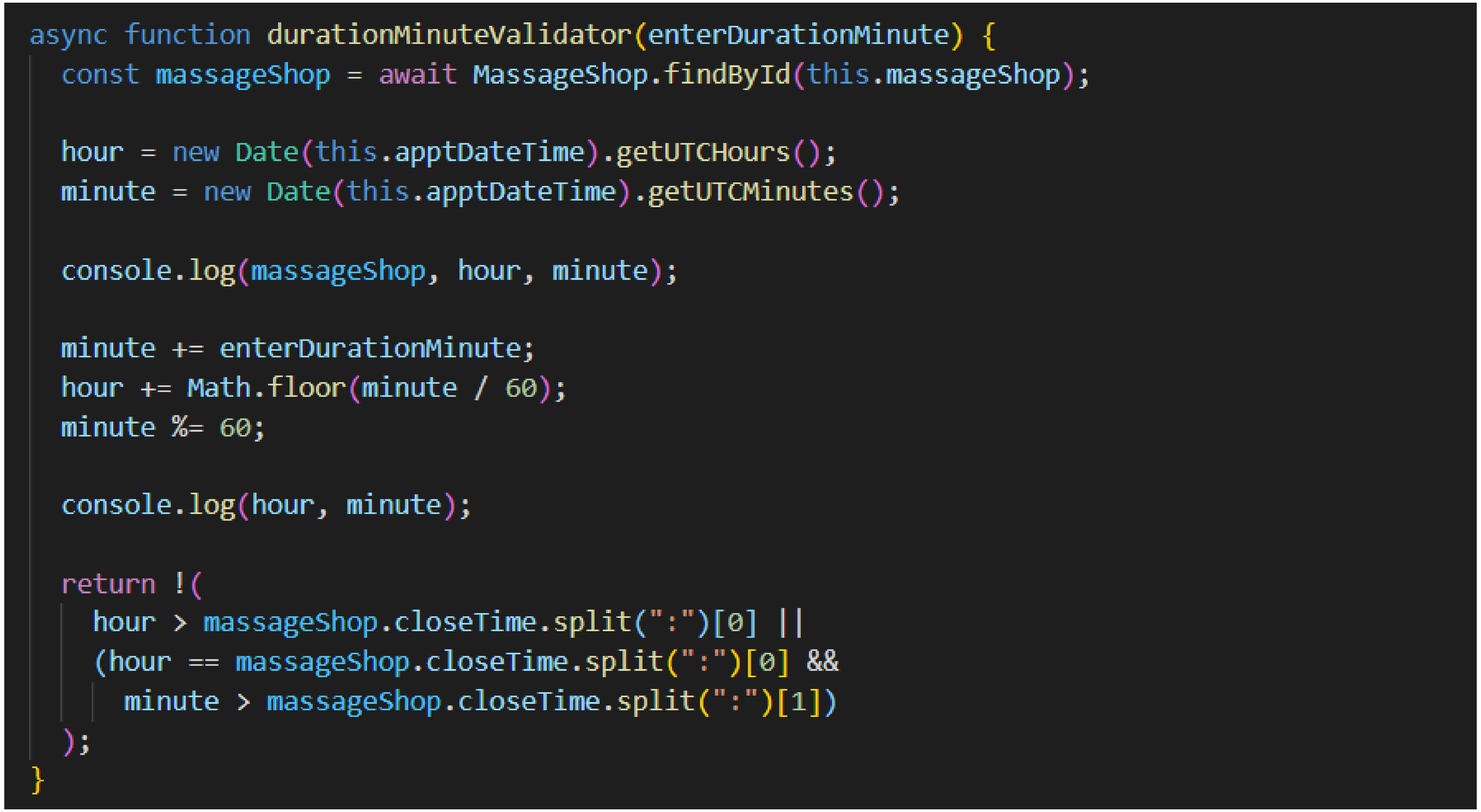
models/Appointment



```
async function apptDateTimeValidator(enterApptDateTime) {
  const massageShop = await MassageShop.findById(this.massageShop);

  hour = new Date(enterApptDateTime).getUTCHours();
  minute = new Date(enterApptDateTime).getUTCMinutes();

  return !(
    hour < massageShop.openTime.split(":")[0] ||
    (hour == massageShop.openTime.split(":")[0] &&
      minute < massageShop.openTime.split(":")[1])
  );
}
```

Code change

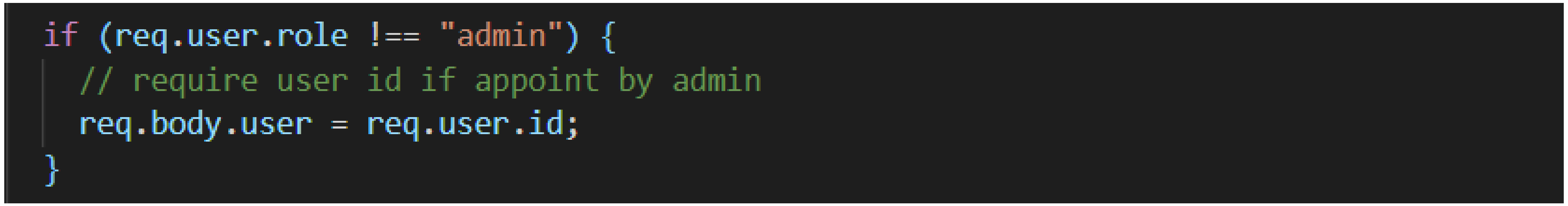
models/Appointment



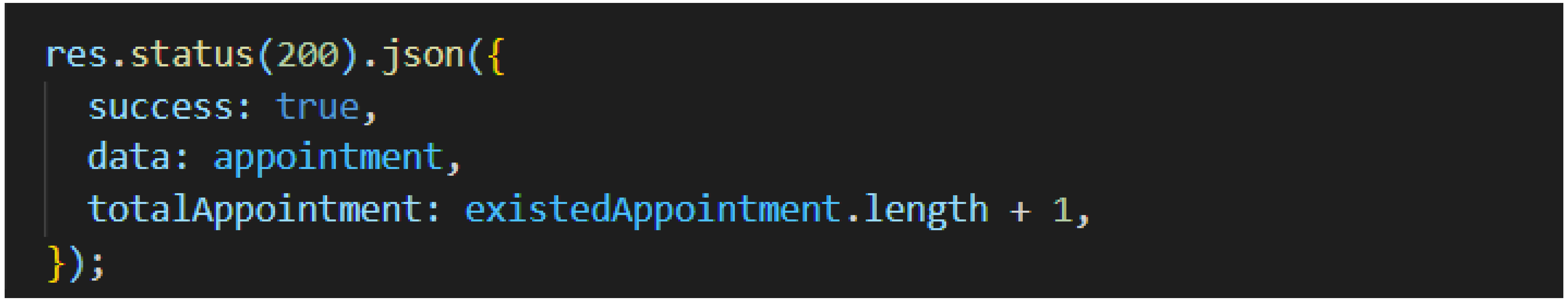
```
async function durationMinuteValidator(enterDurationMinute) {
  const massageShop = await MassageShop.findById(this.massageShop);

  hour = new Date(this.apptDateTime).getUTCHours();
  minute = new Date(this.apptDateTime).getUTCMinutes();

  console.log(massageShop, hour, minute);

  minute += enterDurationMinute;
  hour += Math.floor(minute / 60);
  minute %= 60;

  console.log(hour, minute);

  return !(
    hour > massageShop.closeTime.split(":")[0] ||
    (hour == massageShop.closeTime.split(":")[0] &&
      minute > massageShop.closeTime.split(":")[1])
  );
}
```

Code change

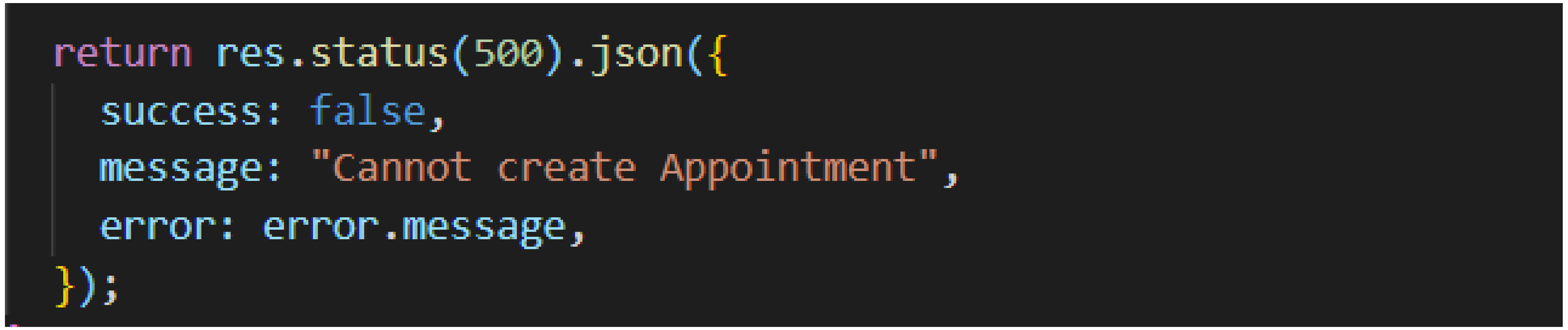
controllers/appointments (addAppointment)



```
if (req.user.role !== "admin") {
  // require user id if appoint by admin
  req.body.user = req.user.id;
}
```

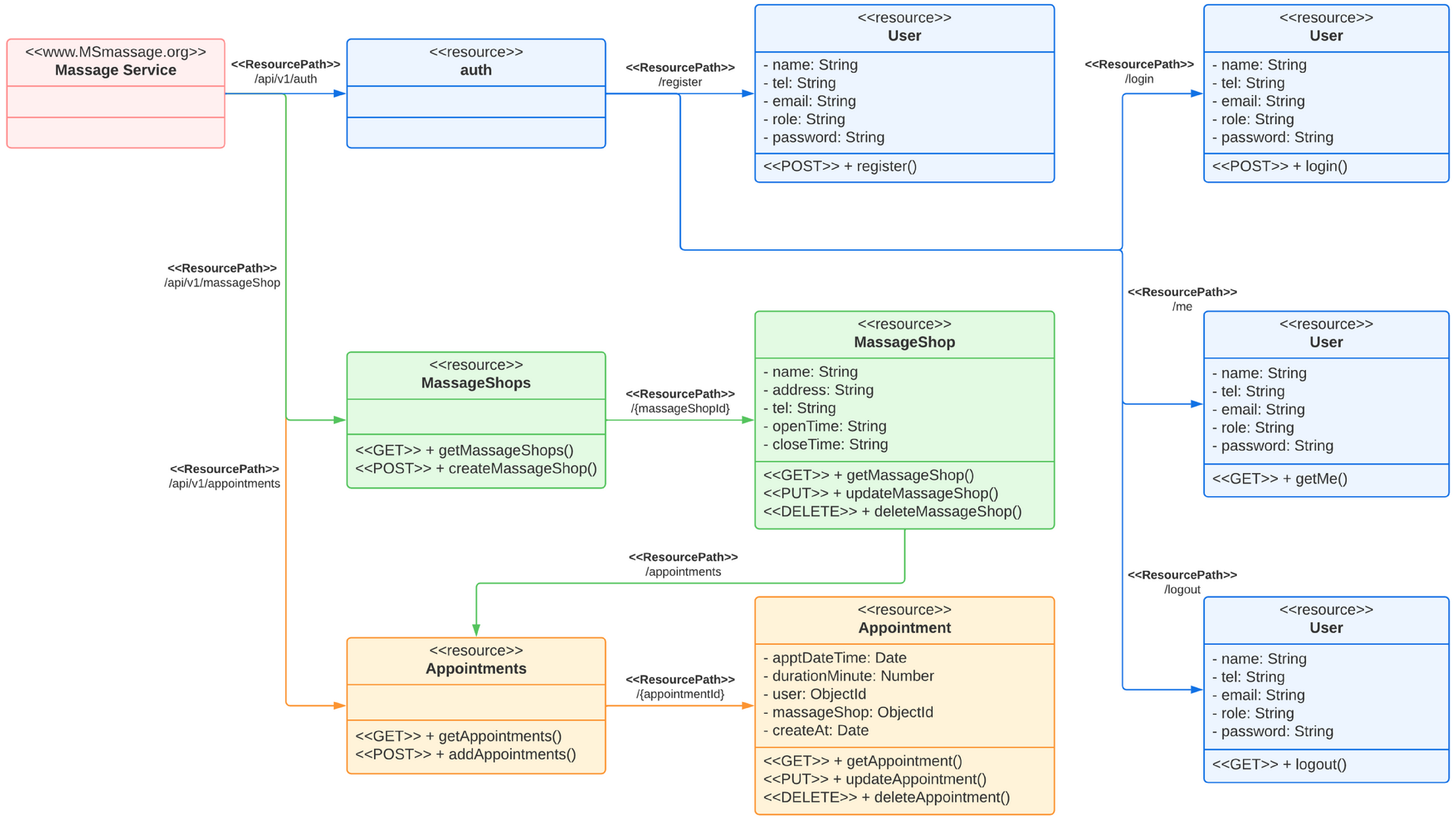


```
res.status(200).json({
  success: true,
  data: appointment,
  totalAppointment: existedAppointment.length + 1,
});
```

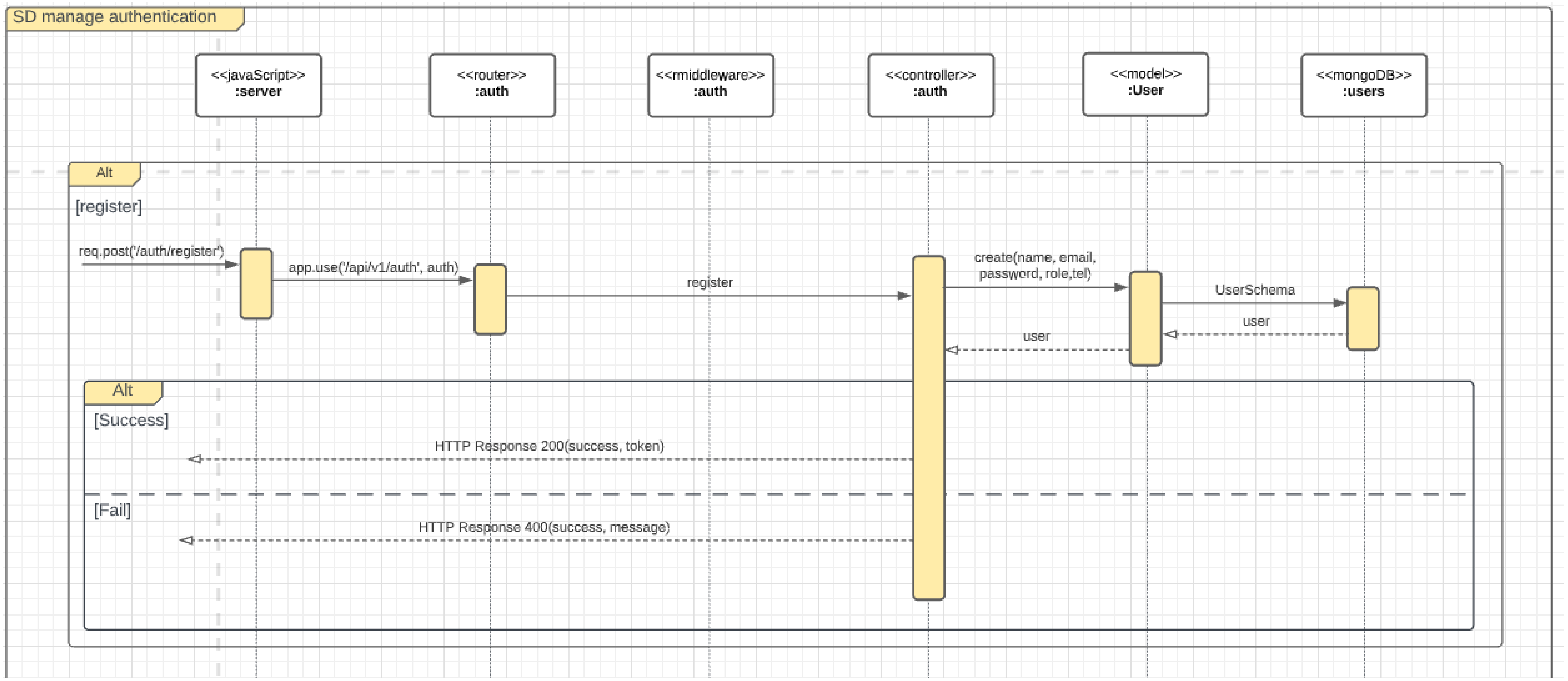


```
return res.status(500).json({
  success: false,
  message: "Cannot create Appointment",
  error: error.message,
});
```

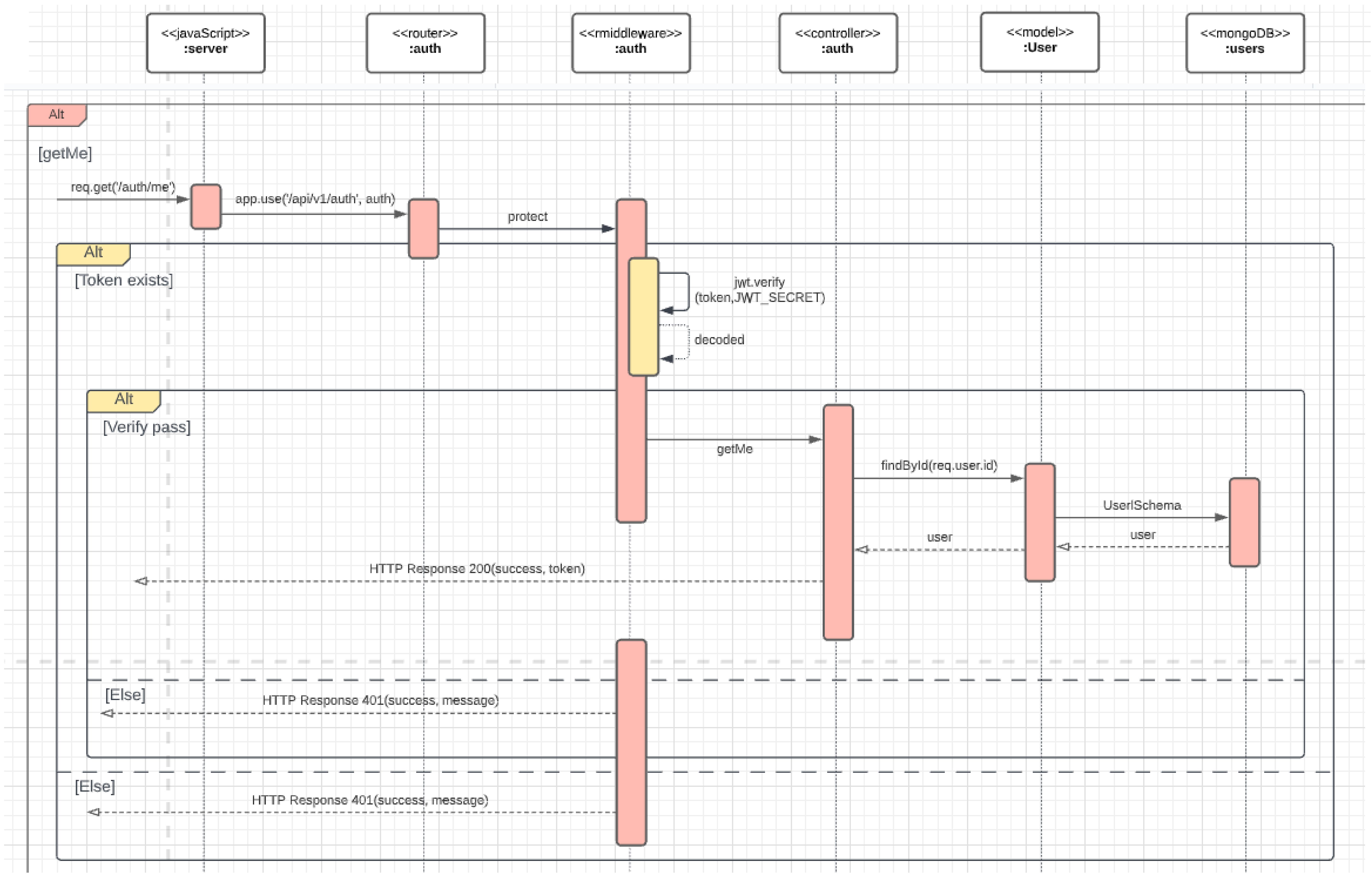
Class Diagram



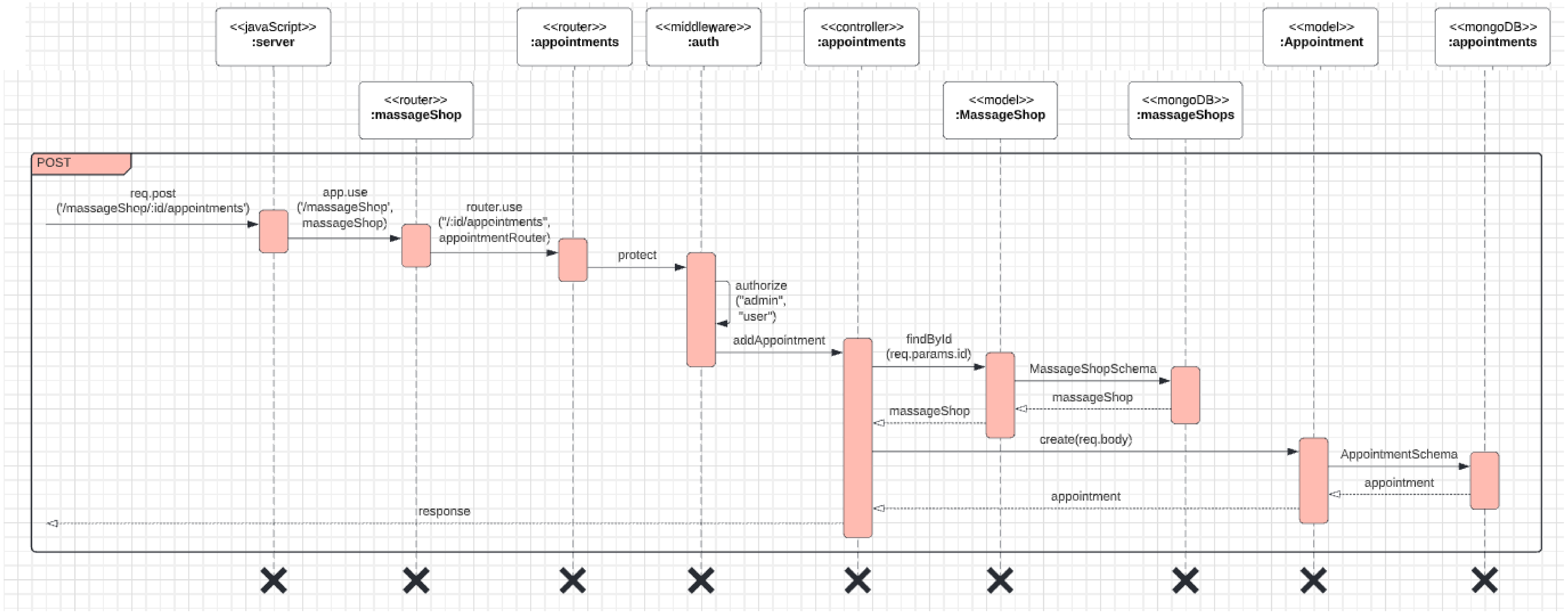
Sequence Diagram (Auth)



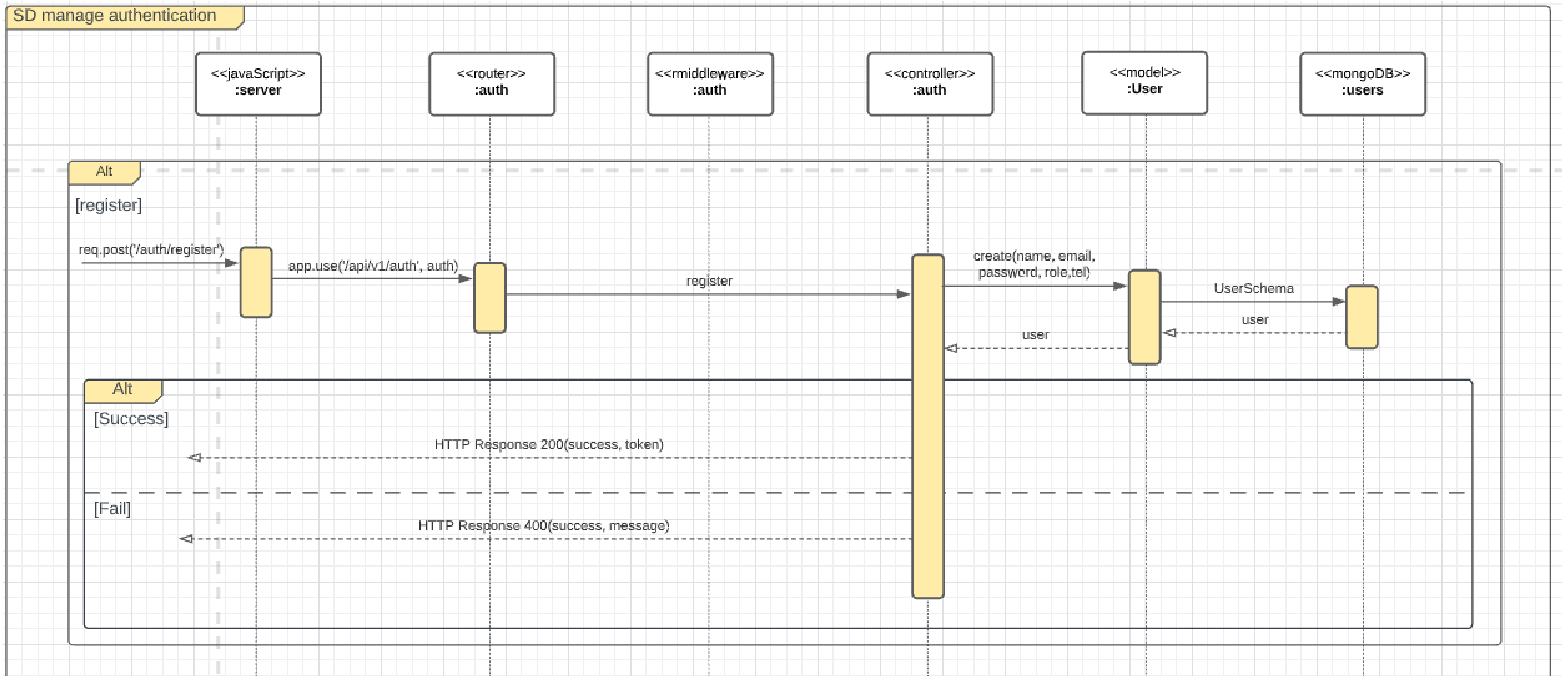
Sequence Diagram (Auth)



Sequence Diagram (Appt)



Sequence Diagram (Auth)



Contributions

Suppawich & Thanin

Thanin Sawetkititham	Suppawich Tawornpichayachai
models: User, Appointment	models: MassageShop
controllers: auth, appointments	controllers: massageShop
routes: auth, appointments	routes: massageShop
Sequence Diagram: appointments	Sequence Diagram: auth, massageShop Class Diagram
Project document	Project presentation