Dentist Booking

By แล้วแต่

Waranthorn Chansawang 6432154921

Nattapong Anansomsin 6431317721

https://github.com/nattapong232/DentistBooking

Fuctional 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 log out.
3. After login, the system shall allow the registered user to book only ONE session by specifying the date and the preferred dentist. The dentist list is also provided to the user. A dentist information includes the dentist's name, years of experience, and area of expertise.
4. The system shall allow the registered user to view his booking.
5. The system shall allow the registered user to edit his booking.
6. The system shall allow the registered user to delete his booking.
7. The system shall allow the admin to view any bookings.
8. 8. The system shall allow the admin to edit any bookings.
9. 9. The system shall allow the admin to delete any bookings.

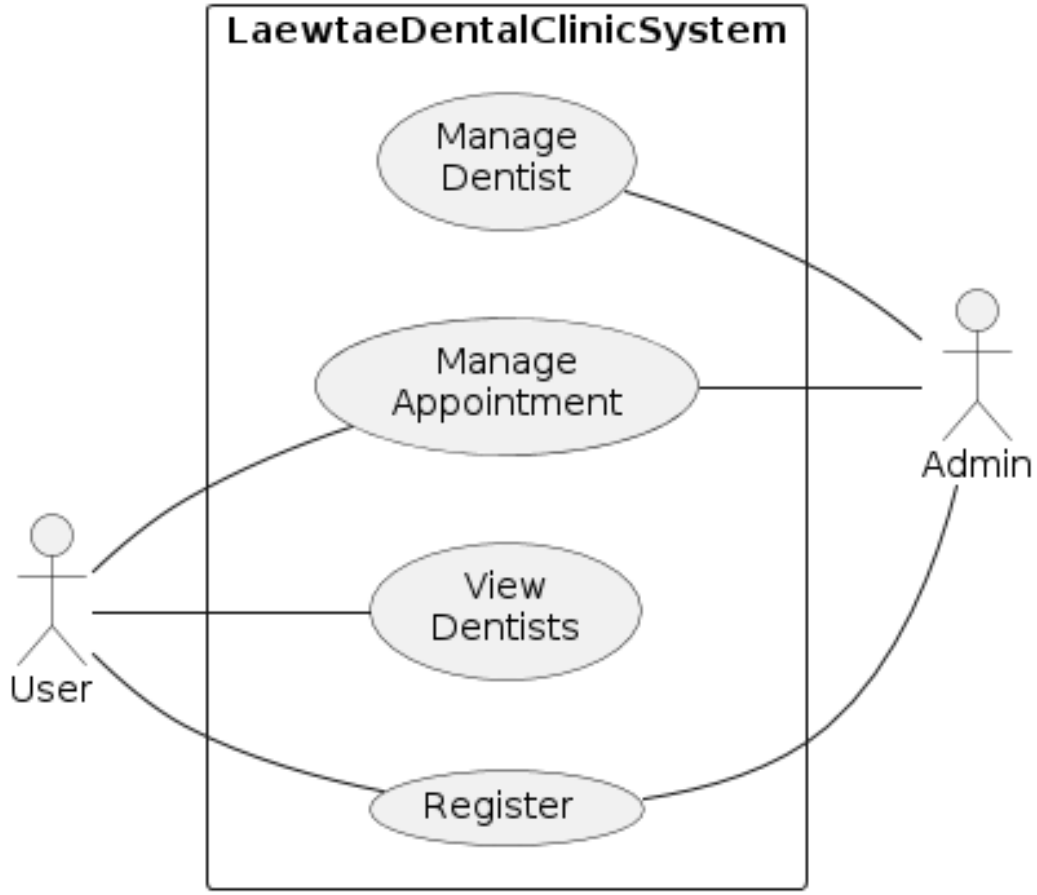
Non-Functional Requirements

- 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.

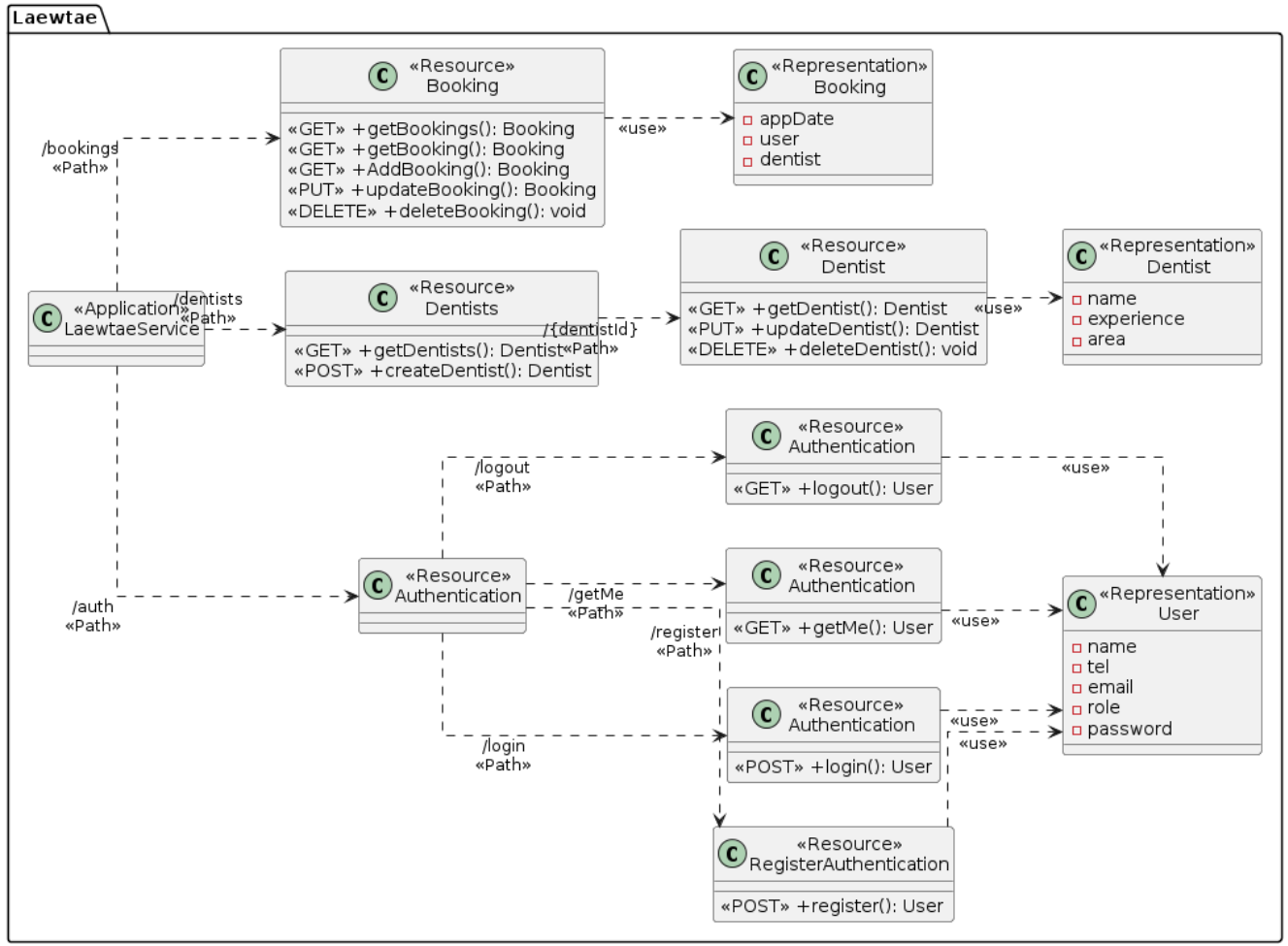
Constraints

- The system shall be a web API.
- The frontend part of the application is not required.
- The development team shall develop the backend system as REST APIs.
- The database system can be either MongoDB Atlas or MySQL.

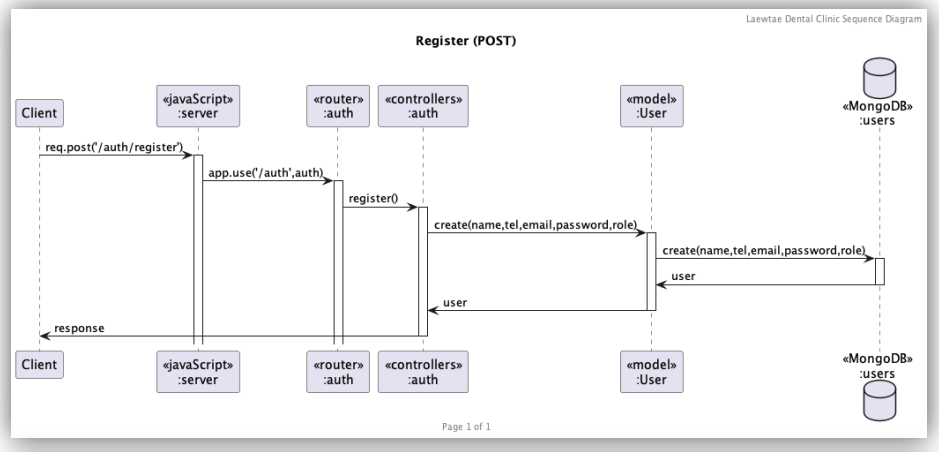
Use Case Diagram

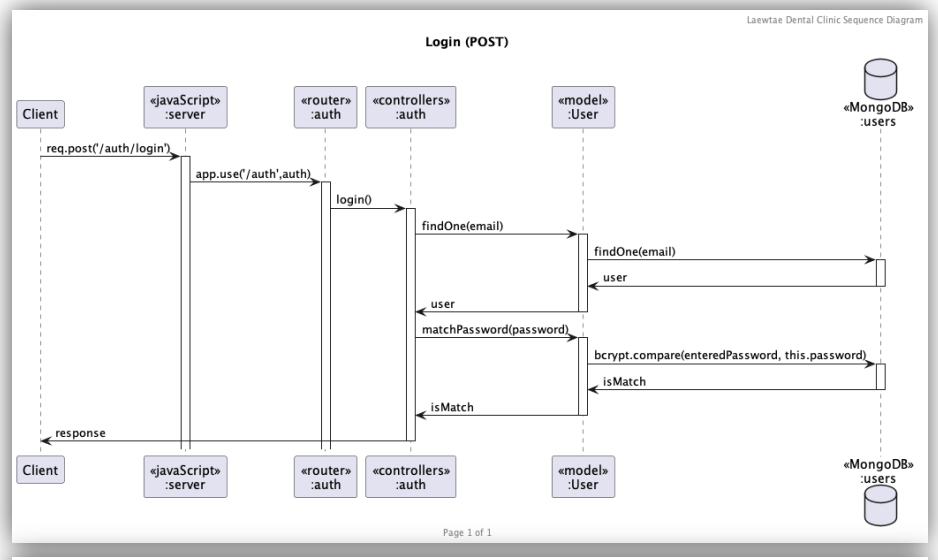


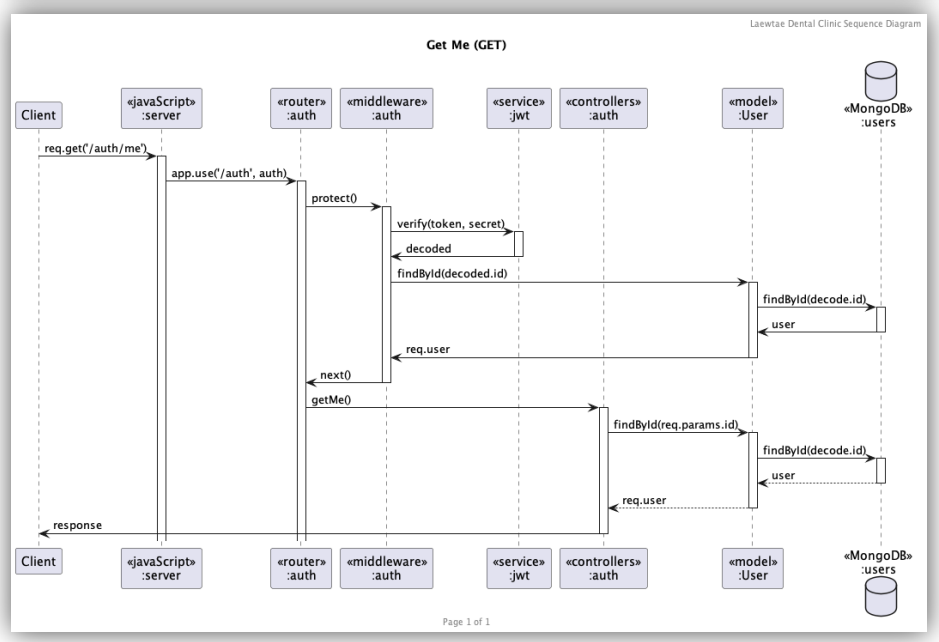
Class Diagram

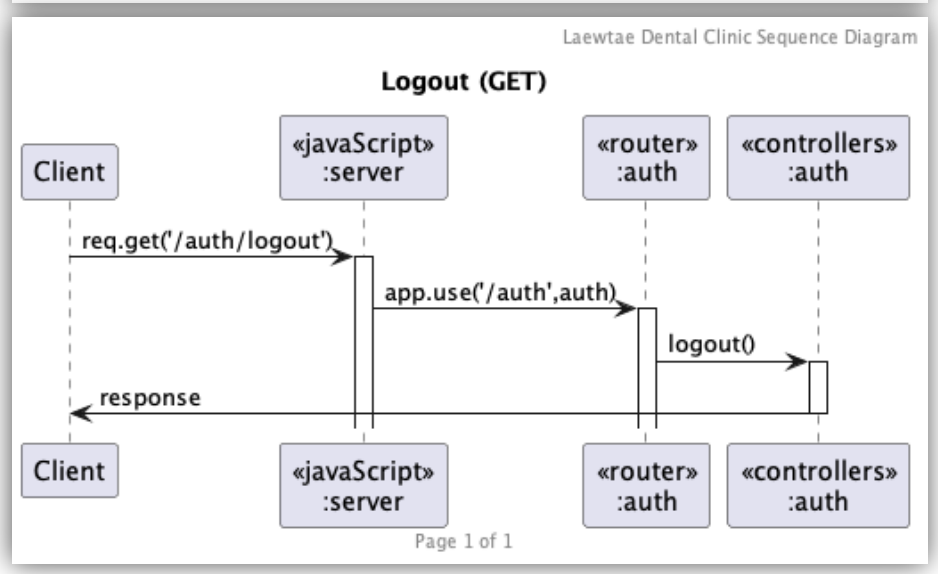


Sequence Diagram - Authentication

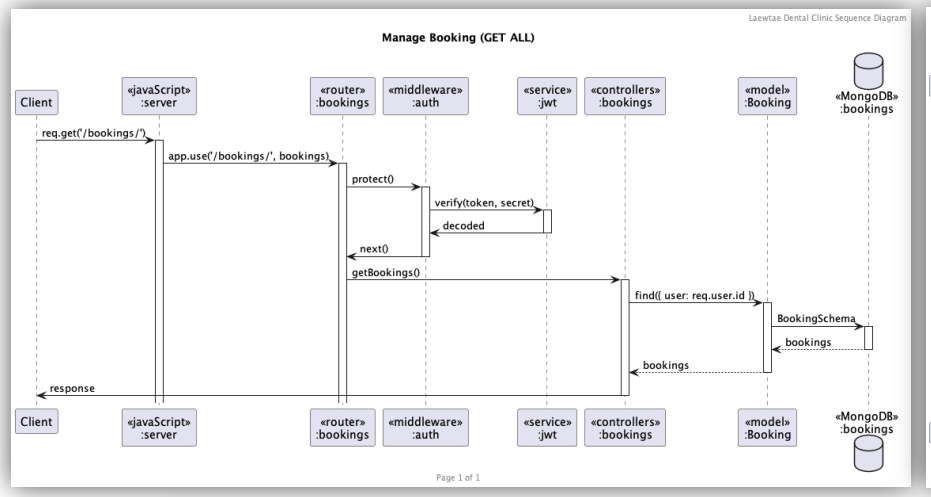


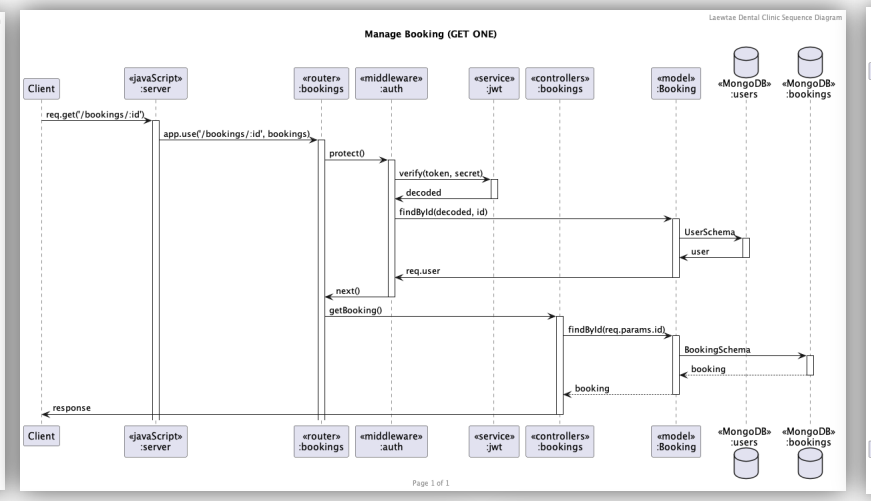


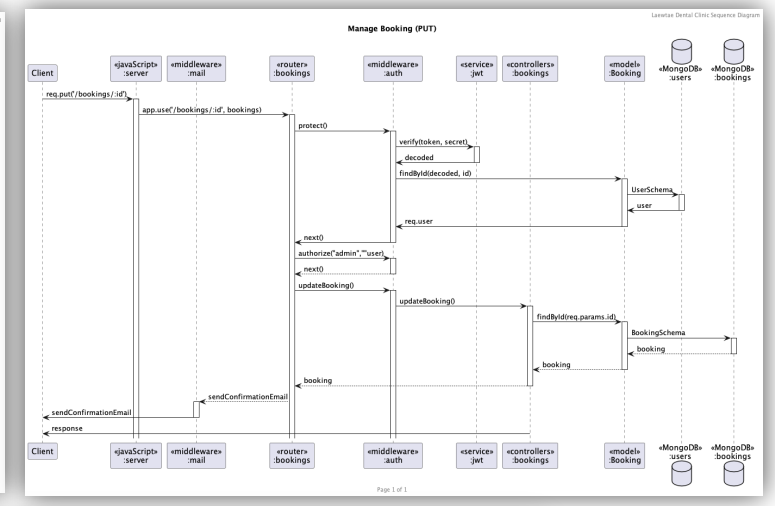


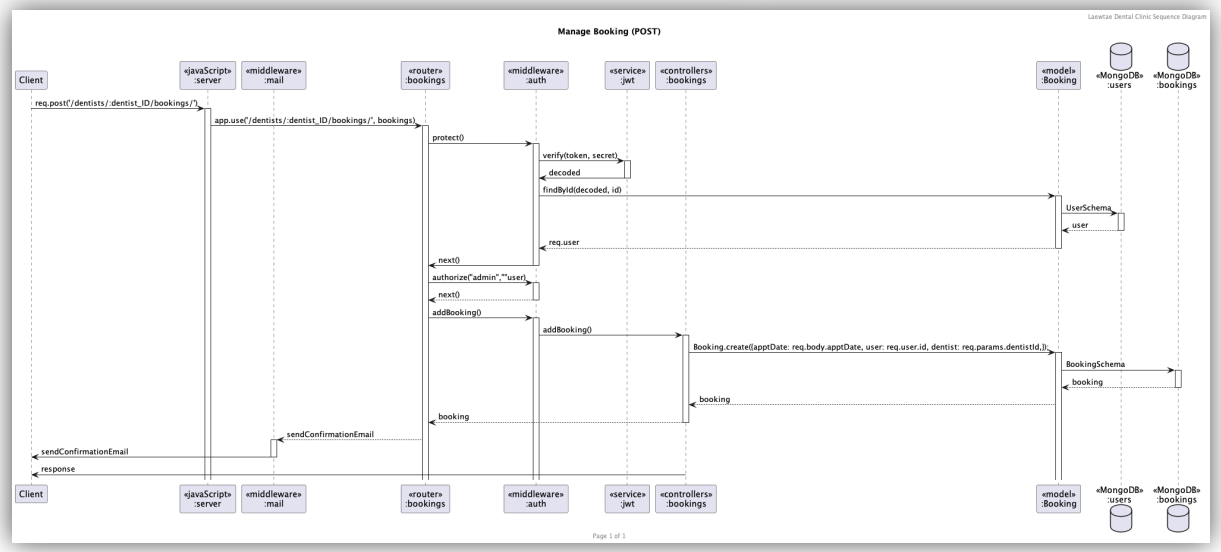


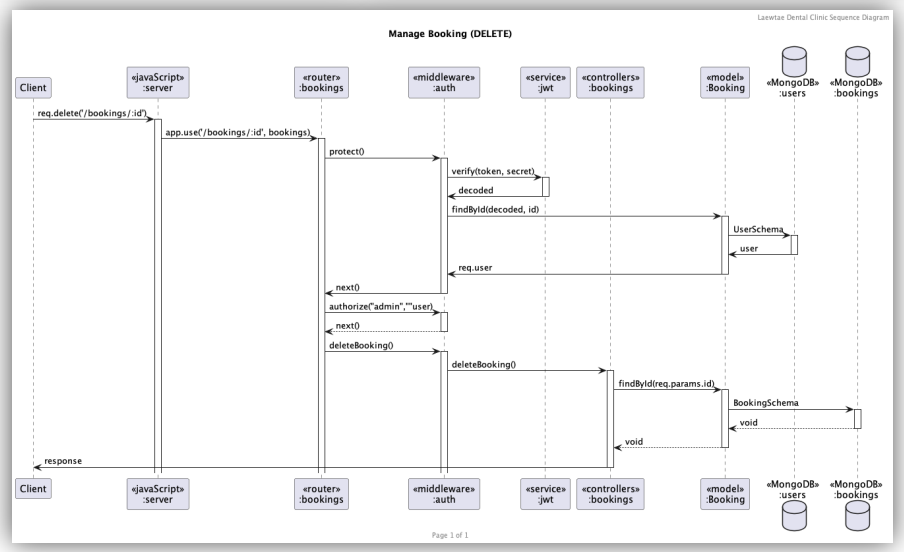
Sequence Diagram - Booking



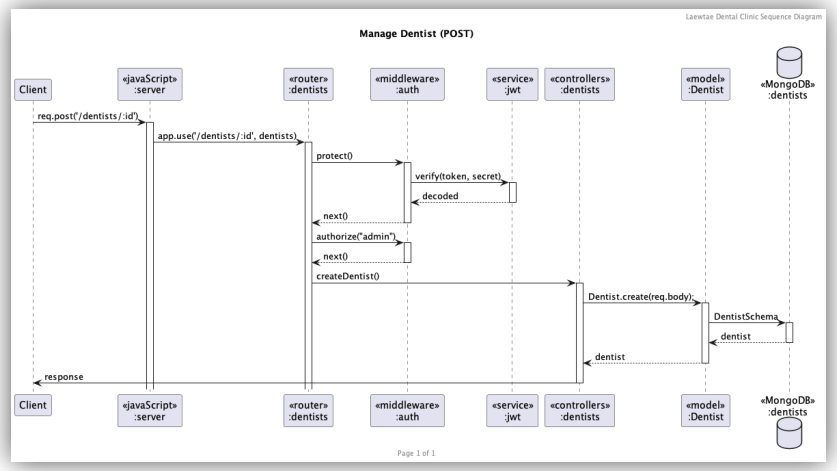


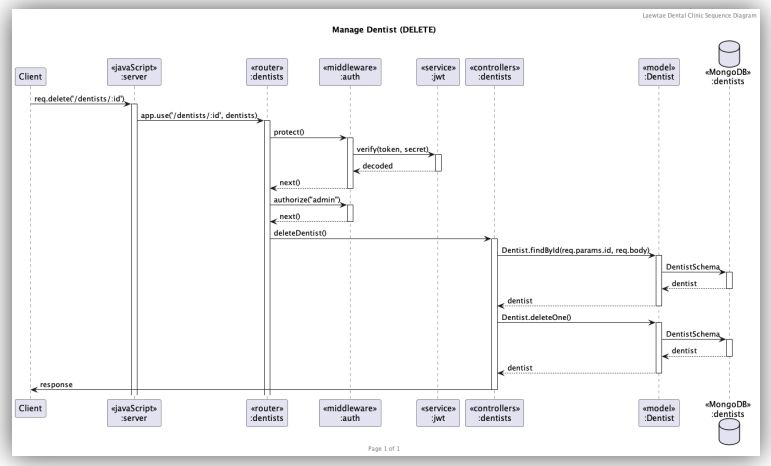


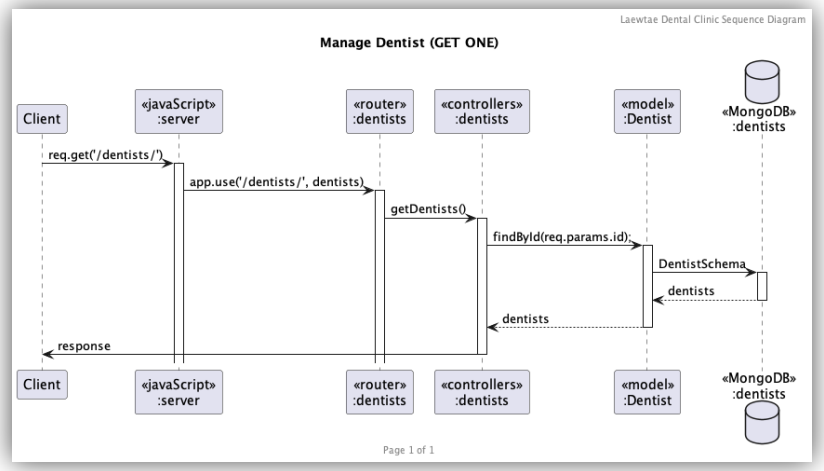


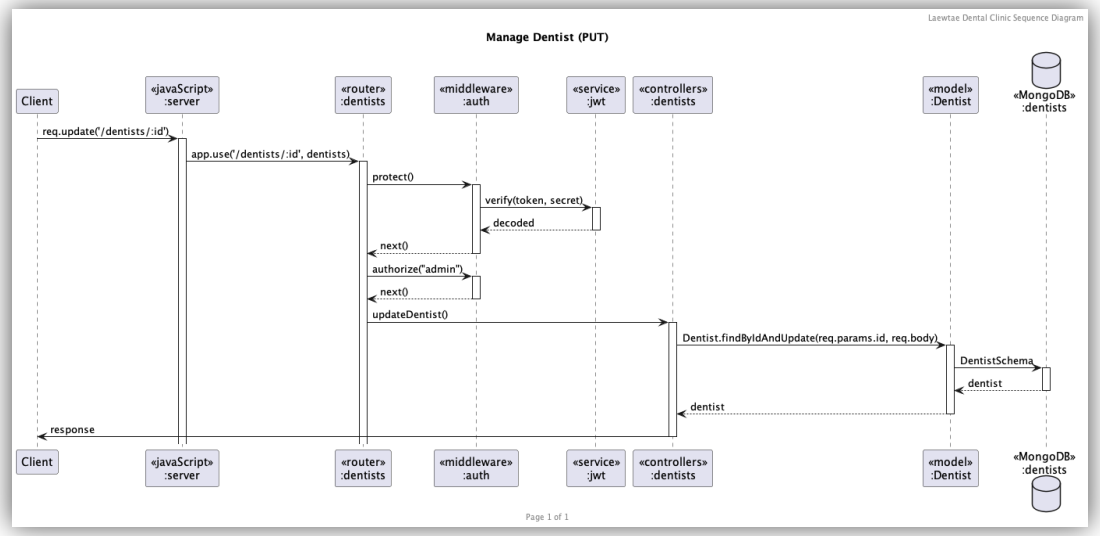


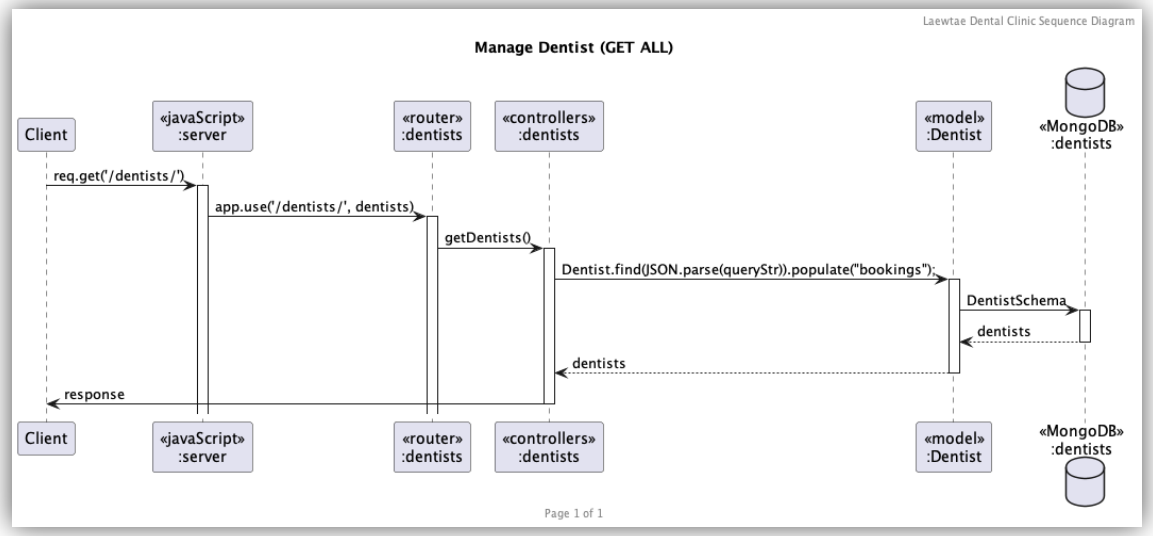
Sequence Diagram - Dentist











Additional Requirement

- The system will send confirmation emails upon successful reservation.

Code ที่แก้ไป

- Change from Hospital -> Dentist
- Change from Appointment -> Booking

config.env

models

Dentist.js

User.js

controllers

booking.js

auth.js

dentist.js

middleware

mail.js

route

booking.js

models/Dentist.js

- Modify from model/Hospital.js to model/Dentist.js

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

const DentistSchema = 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"],
    },
    experience: {
      type: Number,
      required: [true, "Please add year of experience"],
      min: 0,
    },
    area: {
      type: String,
      required: [true, "Please add an area of expertise"],
    },
  },
  {
    toJSON: { virtuals: true },
    toObject: { virtuals: true },
  }
);
```

```
//Reverse populate
DentistSchema.virtual("bookings", {
  ref: "Booking",
  localField: "_id",
  foreignField: "dentist",
  justOne: false,
});

//Cascade delete bookings when a dentist is deleted
DentistSchema.pre(
  "deleteOne",
  { document: true, query: false },
  async function (next) {
    console.log(`Bookings with dentist ${this._id} being removed`);
    await this.model("Booking").deleteMany({ dentist: this._id });
    next();
  }
);

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

models/User.js

- Add tel for user

```
const UserSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, "Please add a name"],
  },
  tel: {
    type: String,
    required: [true, "Please add a telephone number"],
    maxlength: 10,
    match: [
      /^[\+]?[(]?[0-9]{3}[)]?[-\s\.]?[0-9]{3}[-\s\.]?[0-9]
{4,6}$/, "Please add a valid telephone number",
    ],
  },
```

controllers/booking.js

- Modify from controllers/appointments.js to controllers/booking.js
- Modify addBooking for the requirement “After login, the system shall allow the registered user to book only ONE session by specifying the date and the preferred dentist. The dentist list is also provided to the user. A dentist information includes the dentist’s name, years of experience, and area of expertise.”

```
exports.addBooking = async (req, res, next) => {
  try {
    // Check if the user already has an active booking
    const existingBooking = await Booking.findOne({
      user: req.user.id,
    });

    // If an booking exists, prevent creating a new one
    if (existingBooking) {
      return res.status(400).json({
        success: false,
        message: "User already has an active booking",
      });
    }

    // Proceed to create a new booking if no existing one is found
    const booking = await Booking.create({
      apptDate: req.body.apptDate,
      user: req.user.id,
      dentist: req.params.dentistId,
    });

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

    console.log("Create booking successfully");

    res.status(201).json({
      success: true,
      data: booking,
    });

    next();
  } catch (err) {
    console.log(err.stack);
    return res.status(500).json({
      success: false,
      message: "Cannot create booking",
    });
  }
};
```

controllers/auth.js

- Add tel for user

```
exports.register = async (req, res, next) => {
  try {
    const { name, tel, email, password, role } =
req.body;
    const user = await User.create({
      name,
      tel,
      email,
      password,
      role,
    });
```

controllers/dentists.js

- Change Hosptital(s) to Dentist(s)
- Change hosptital(s) to dentist(s)

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

//@desc     Get all dentists
//@route    GET /api/v1/dentists
//@access   Public
exports.getDentists = async (req, res, next) => {
  let query;

  const reqQuery = { ...req.query };
  // console.log(reqQuery);

  const removeFields = ["select", "sort", "page",
"limit"];
  removeFields.forEach((param) => delete reqQuery[param]);
  // console.log(reqQuery);

  let queryStr = JSON.stringify(reqQuery);

  queryStr = queryStr.replace(
    /\b(gt|gte|lt|lte|in)\b/g,
    (match) => `$${match}`
  );
```

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

    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 });
  }
  // res
  //    .status(200)
  //    .json({ success: true, msg: `Show dentist ${req.params.id}`
};;
```

controllers/dentists.js

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

```
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 });
  }
};
```

```
exports.deleteDentist = async (req, res, next) => {
  try {
    const dentist = await Dentist.findById(req.params.id, req.body,
{
      new: true,
      runValidators: true,
    });

    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 });
  }
};
```

config.env

- Connect to new database
- Add username & password for email to send

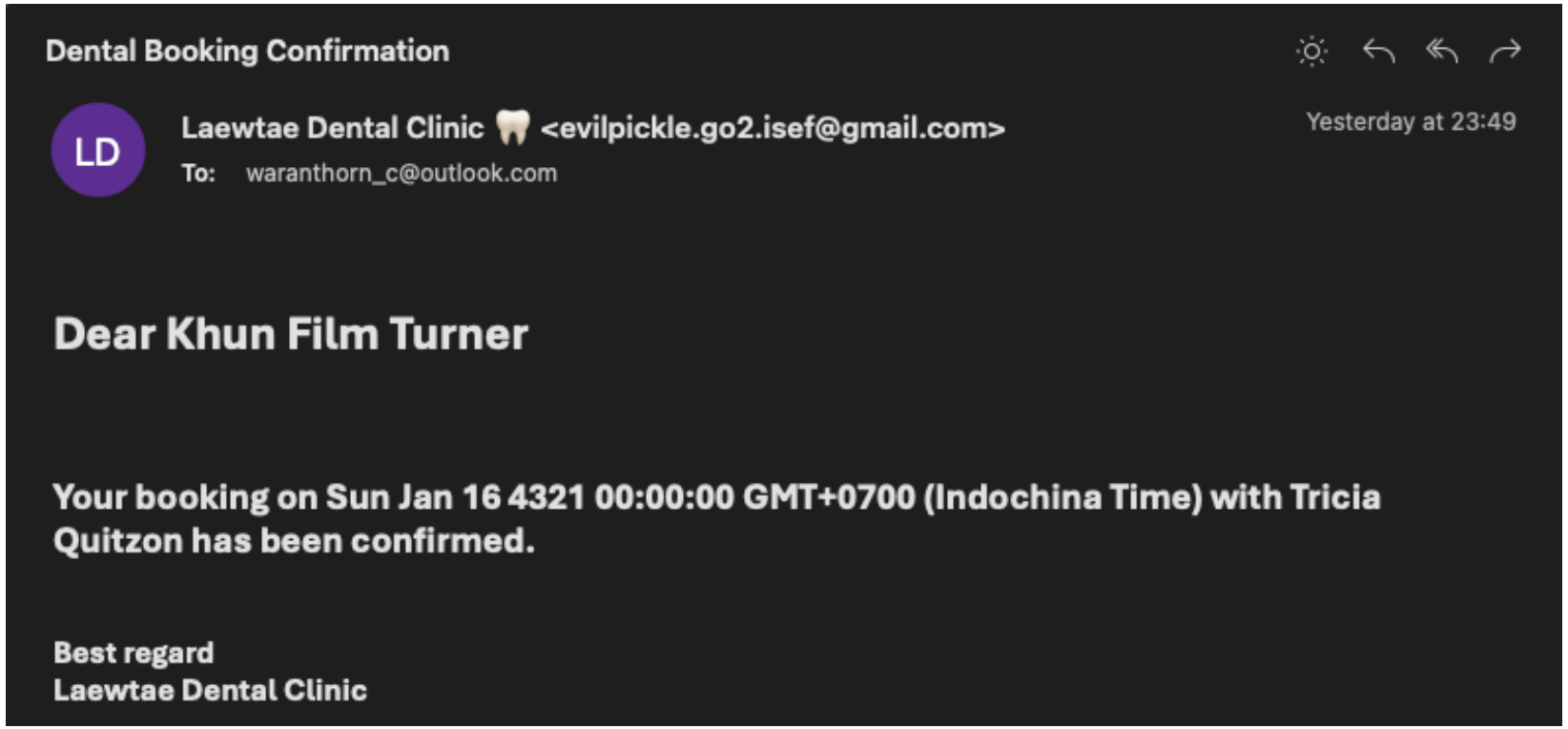
```
PORT = 5001
NODE_ENV = development
# MONGO_URI = mongodb+srv://Nattapong:Shelves04654@vacqcluster.h2fniqp.mongodb.net/VacQ?
retryWrites=true&w=majority
MONGO_URI = mongodb+srv://6431317721:DentistBooking@dentistcluster.wta4sas.mongodb.net/DentistBooking?
retryWrites=true&w=majority&appName=DentistCluster
JWT_SECRET = asdfjkl;;lkjfdsa
JWT_EXPIRE = 30d

JWT_COOKIE_EXPIRE = 30

EMAIL_USERNAME = 'evilpickle.go2.isef@gmail.com'
EMAIL_PASSWORD = 'afzq lljx selt tiso'
```

Additional Requirement

- The system will send confirmation emails upon successful reservation.



middleware/mail.js

- Using Gmail and Nodemailer


```
const nodemailer = require("nodemailer");

exports.sendConfirmationEmail = async (req, res, next) =>
{ const transporter = nodemailer.createTransport({
    service: "gmail",
    host: "smtp.gmail.com",
    port: 587, // SMTP port for TLS/STARTTLS
    secure: false, // Use TLS
    auth: {
      user: process.env.EMAIL_USERNAME,
      pass: process.env.EMAIL_PASSWORD,
    },
  });
```

```
const mailOptions = {
    from: '"Laewtae Dental Clinic 🦷" <evilpickle.go2.isef@gmail.com>',
    to: req.user.email,
    subject: "Dental Booking Confirmation",
    html: `<h2>Dear Khun ${req.user.name}</h2>
        <br>
        <h3>Your booking on ${req.booking.apptDate} with ${req.dentist.name} has been confirmed.</h3>
        <br>
        <b>Best regard</b>
        <br>
        <b>Laewtae Dental Clinic</b>`,
  };
  try {
    await transporter.sendMail(mailOptions);
    console.log("Sent");
  } catch (error) {
    console.log(error);
  }
};
```

routes/booking.js

- Add sendConfirmationEmail to router of addBooking and updateBooking

```
const { sendConfirmationEmail } = require("../middleware/mail");

router
  .route("/")
  .get(protect, getBookings)
  .post(
    protect,
    authorize("admin", "user"),
    addBooking,
    sendConfirmationEmail
  );
router
  .route("/:id")
  .get(protect, getBooking)
  .put(
    protect,
    authorize("admin", "user"),
    updateBooking,
    sendConfirmationEmail
  )
```

การแบ่งงานในทีม

Waranthorn Chansawang 6432154921

- แก้ function ให้ใช้งานได้ตรงตาม requirement
- เพิ่ม Additional Requirement
- ทำ Document (Changed Code, Class Diagram, Sequence Diagram, Use Case Diagram)
- ทำ VDO

Nattapong Anansomsin 6431317721

- แก้ไฟล์และตัวแปรจาก Hospital ให้เป็น Dentist และ Appointment เป็น Booking
- จัดการไฟล์ Additional Requirement ให้เป็น Module
- สร้างและต่อ MongoDB
- ทำ Runner
- ทำ Document (Sequence Diagram)

VDO Link

https://youtu.be/sCq5MmsQzfU

THANK YOU