ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

StrikePlagiarism.com

Дата звіту 6/6/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-6_Чернова_А_М_скорочений
Автор
Чернова Анастасія Максимівна
Науковий керівник / Експерт
Євген Кардаш
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.95% 0.95% КП 1

0.89% 0.89% КЦ

25	7904	63485
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	8	11
Інтервали	A→	0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	a	2

Подібності за списком джерел

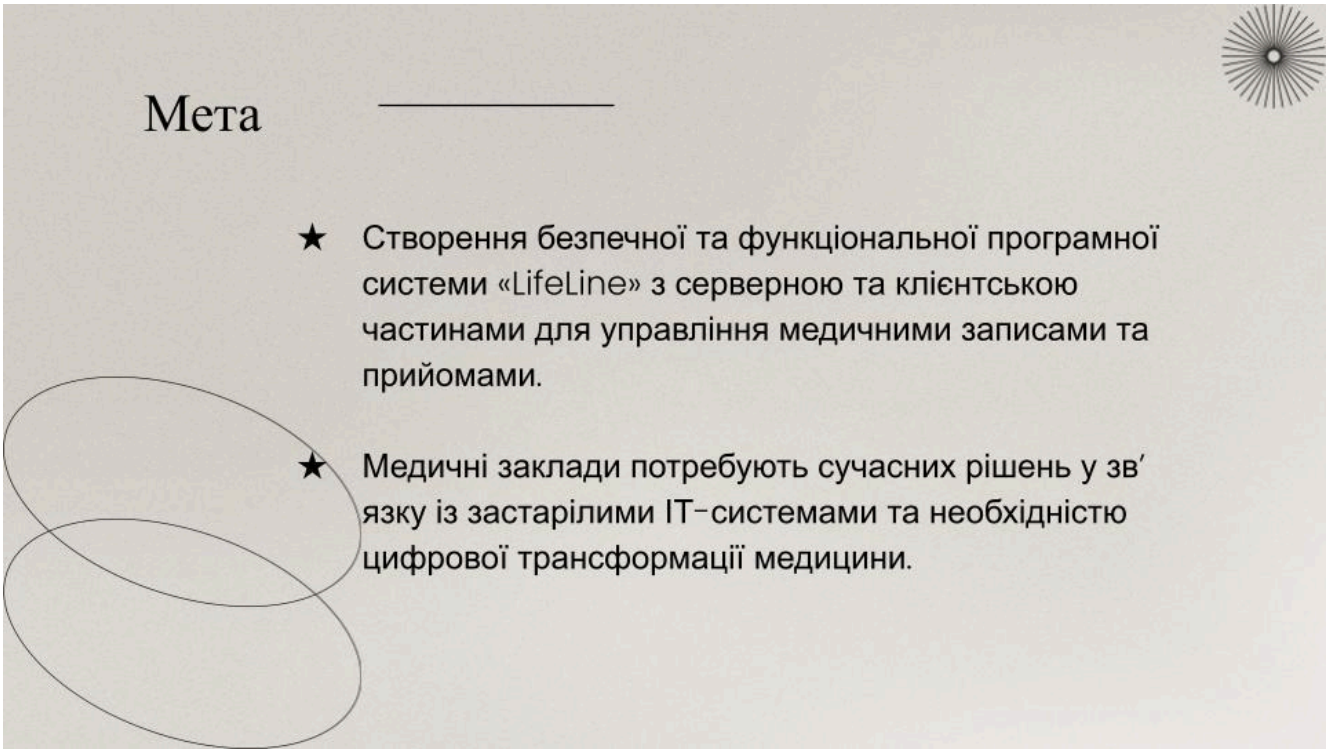
Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://www.basicexamples.com/example/node-js/crypto-createcipheriv	12 0.15 %
2	Кваліфікаційна робота Степаненко [redacted] 3/10/2025 Interregional Academy of Personnel Management (Інститут комп'ютерно-інформаційних технологій та дизайну)	9 0.11 %

ДОДАТОК Б

Слайди презентації



Аналіз проблеми

Застарілі системи

Низька ефективність

Ненадійність та небезпека

Витрати часу персоналу

Низька зручність для пацієнтів

Аналіз існуючих рішень

Існуючі медичні інформаційні системи не повністю відповідають потребам українських закладів: мають складний інтерфейс, обмежену адаптивність, проблеми з безпекою.

Конкурент	Переваги	Недоліки
Helsi	Інтеграція з eHealth, електронна картка, е-рецепти	Складний інтерфейс, нестабільна робота, обмежена адаптація
Health24	Хмарна архітектура, мобільні додатки, підтримка eHealth	Відсутність офлайн-режиму, слабка техпідтримка, фіксований функціонал

Постановка задачі

- ★ Розробити медичну інформаційну систему, що складається з серверної та клієнтської частин.
- ★ Серверна частина повинна забезпечити обробку запитів, зберігання даних пацієнтів, авторизацію, доступ за ролями та захист інформації.
- ★ Клієнтська частина має надавати зручний інтерфейс для пацієнтів, лікарів і адміністраторів.
- ★ Передбачена взаємодія через REST API та підтримка основних функцій: реєстрація, запис на прийом, робота з медкартками, генерація PDF, управління графіками.
- ★ Система повинна бути безпечною, зрозумілою у використанні та готовою до реального впровадження.

Вибір технологій розробки для Back-end частини

01 JS

02 node

03

04 JWT

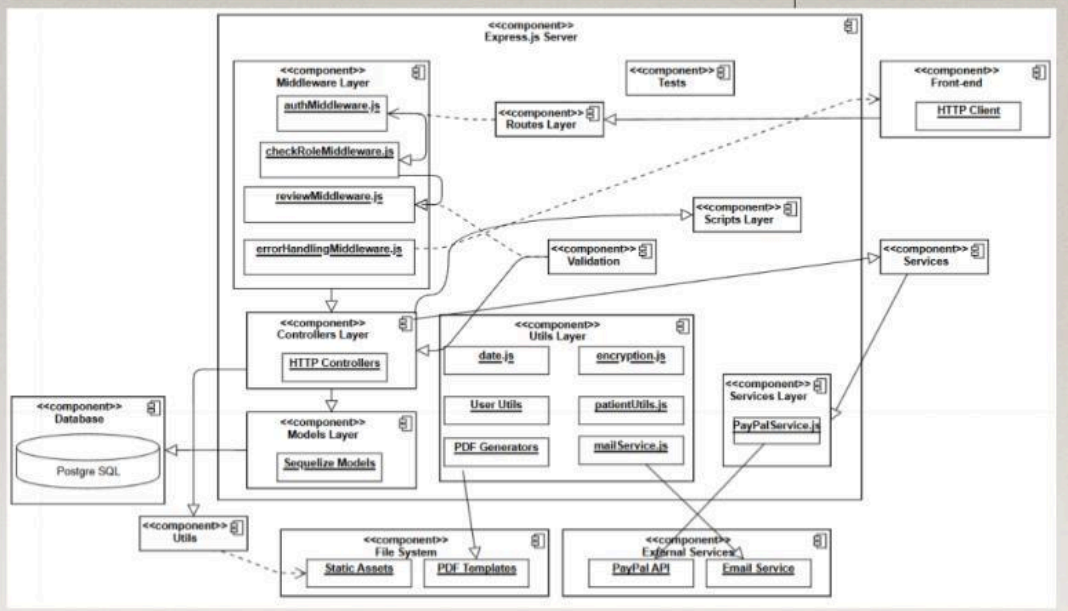
05 Sequelize

06 BCrypt HASH

07 PDFKit

08 Express JS

Архітектура створеного програмного забезпечення



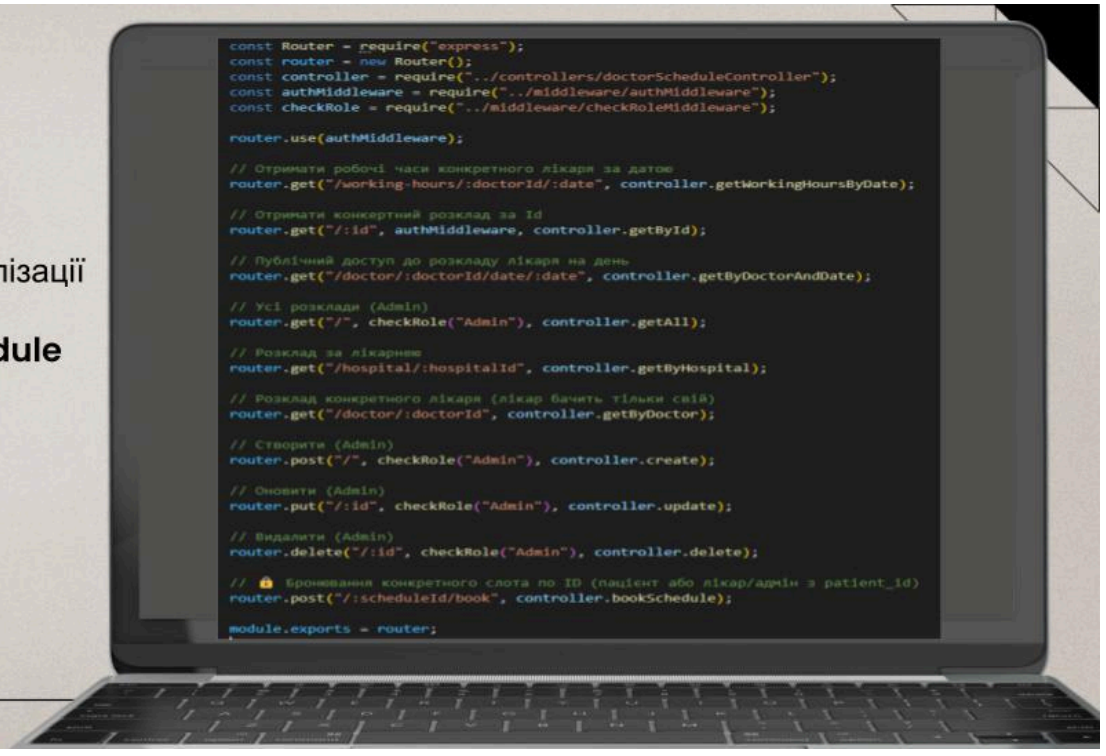
Головна функціональність

- ★ Управління електронними медичними картками
- ★ Запис на прийоми та обробка прийомів
- ★ Виписка рецептів та направлень
- ★ Розподіл доступу за ролями (Admin, Doctor, Patient)
- ★ Розклад лікарів, медичних і лабораторних послуг
- ★ Статистика та звітність
- ★ Захист даних: шифрування, авторизація, резервне копіювання
- ★ Аналітика активності користувачів
- ★ Модуль лабораторних послуг
- ★ Управління фінансовими звітами
- ★ Система відгуків та оцінок
- ★ Модуль оплати та інтеграція з платіжною системою
- ★ Панель адміністратора для управління працівниками та пацієнтами

Нефункціональні вимоги

- ★ Продуктивність
- ★ Масштабованість
- ★ Безперебійна робота системи
- ★ Безпека
- ★ Простий, інтуїтивний інтерфейс; документація API
- ★ Мультимовність: українська мова за замовчуванням, можливість додати інші
- ★ Сумісність з різними клієнтськими платформами
- ★ Безпечні оновлення без втрати даних і зупинки системи

Приклад реалізації роутеру **DoctorSchedule**



```
const Router = require("express");
const router = new Router();
const controller = require("../controllers/doctorScheduleController");
const authMiddleware = require("../middleware/authMiddleware");
const checkRole = require("../middleware/checkRoleMiddleware");

router.use(authMiddleware);

// Отримати робочі часи конкретного лікаря за датою
router.get("/working-hours/:doctorId/:date", controller.getWorkingHoursByDate);

// Отримати конкретний розклад за Id
router.get("/:id", authMiddleware, controller.getById);

// Публічний доступ до розкладу лікаря на день
router.get("/doctor/:doctorId/date/:date", controller.getByDoctorAndDate);

// Усі розклади (Admin)
router.get("/", checkRole("Admin"), controller.getAll);

// Розклад за лікарнею
router.get("/hospital/:hospitalId", controller.getByHospital);

// Розклад конкретного лікаря (лікар бачить тільки свій)
router.get("/doctor/:doctorId", controller.getByDoctor);

// Створити (Admin)
router.post("/", checkRole("Admin"), controller.create);

// Оновити (Admin)
router.put("/:id", checkRole("Admin"), controller.update);

// Видалити (Admin)
router.delete("/:id", checkRole("Admin"), controller.delete);

// 🔒 Бронювання конкретного слота по ID (пацієнт або лікар/адмін з patient_id)
router.post("/:scheduleId/book", controller.bookSchedule);

module.exports = router;
```

Приклад реалізації бронювання часу прийому в контролері **DoctorSchedule**

```
async bookSchedule(req, res, next) {
  try {
    const { scheduleId } = req.params;

    const schedule = await DoctorSchedule.findByPk(scheduleId);
    if (!schedule) return next(ApiError.notFound("Час розкладу не знайдено"));

    if (schedule.is_booked) {
      return next(ApiError.badRequest("Цей час вже заброньований"));
    }

    let patient;
    if (req.user.role === "Patient") {
      patient = await Patient.findOne({ where: { user_id: req.user.id } });
      if (!patient) return next(ApiError.forbidden("Пацієнта не знайдено"));
    } else if (req.body.patient_id) {
      patient = await Patient.findByPk(req.body.patient_id);
      if (!patient) return next(ApiError.badRequest("Невірний patient_id"));
    } else {
      return next(ApiError.badRequest("Необхідно вказати пацієнта"));
    }

    const appointment = await Appointment.create({
      patient_id: patient.id,
      doctor_id: schedule.doctor_id,
      doctor_schedule_id: schedule.id,
      appointment_date: schedule.appointment_date,
      status: "Scheduled",
    });

    await schedule.update({ is_booked: true });

    return res.json({
      message: "Час успішно заброньовано",
      appointment,
    });
  } catch (e) {
    console.error("bookSchedule error:", e);
    return next(ApiError.internal("Не вдалося забронювати час"));
  }
```

Приклад формування **PDF** результату процедури.

МІНІСТЕРСТВО ОХОРОНИ ЗДОРОВ'Я УКРАЇНИ

МЕДИЧНА ПРОЦЕДУРА

Лікарня: Медичний центр «Надія», м. Харків, проспект Науки 12

Пацієнт: Шевченко Анастасія Петрівна
Дата народження: 23.11.1990
Лікар: Іванова Анастасія Сергіївна
Спеціалізація: кардіолог

Дата надання послуги: 24.05.2025

Результати процедури

Procedure was successful. No complications.

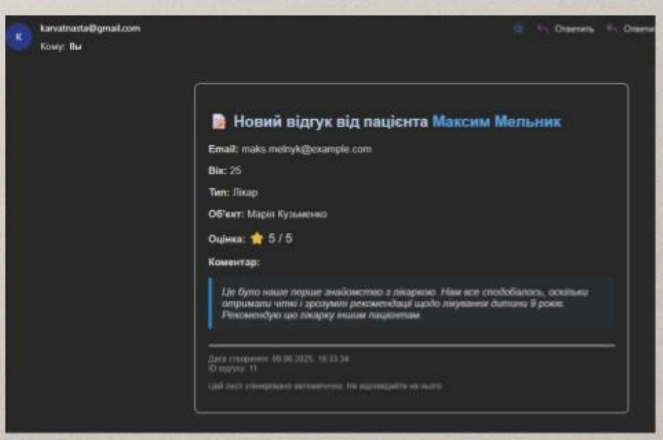
Примітки

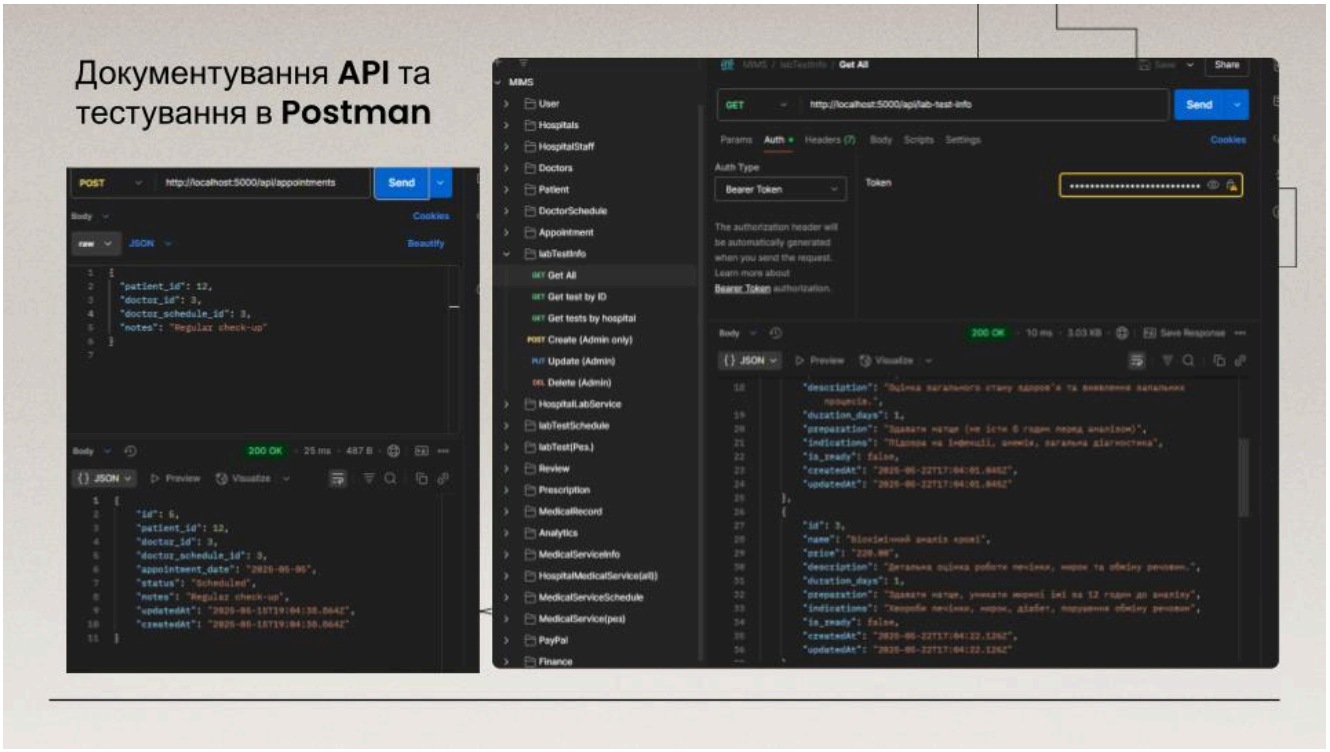
Patient tolerated the procedure well.

Дата видачі: 09.06.2025

Підпис лікаря

Приклад отриманого на пошті лікарні відгуку від пацієнта





Підсумки

Реалістичність і корисність:
Реалізована система відповідає реальним потребам українських медичних закладів, забезпечуючи безпечне зберігання та обробку даних, зручність для лікарів, пацієнтів і адміністрації.

Можливості використання:
Система може бути впроваджена в державних і приватних клініках, амбулаторіях, лабораторіях, адаптується до різних масштабів установ.

Подальший розвиток:
Планується додавання мобільного додатку, чат-бота для запису, інтеграції з eHealth, розширення аналітичних можливостей та підтримка багатомовності.

ДОДАТОК В

Фрагменти коду серверної частини

В.1 - Код для роутеру : doctorScheduleRoutes.js

```
1. const Router = require("express");
2. const router = new Router();
3. const controller =
   require("../controllers/doctorScheduleController");
4. const authMiddleware = require("../middleware/authMiddleware");
5. const checkRole = require("../middleware/checkRoleMiddleware");
6.
7. router.use(authMiddleware);
8.
9. // Отримати робочі часи конкретного лікаря за датою
10.   router.get("/working-hours/:doctorId/:date",
   controller.getWorkingHoursByDate);
11.
12.   // Отримати конкертний розклад за Id
13.   router.get("/:id", authMiddleware, controller.getById);
14.
15.   // Публічний доступ до розкладу лікаря на день
16.   router.get("/doctor/:doctorId/date/:date",
   controller.getByDoctorAndDate);
17.
18.   // Усі розклади (Admin)
19.   router.get("/", checkRole("Admin"), controller.getAll);
20.
21.   // Розклад за лікарнею
22.   router.get("/hospital/:hospitalId", controller.getByHospital);
23.
24.   // Розклад конкретного лікаря (лікар бачить тільки свій)
25.   router.get("/doctor/:doctorId", controller.getByDoctor);
26.
27.   // Створити (Admin)
28.   router.post("/", checkRole("Admin"), controller.create);
29.
30.   // Оновити (Admin)
31.   router.put("/:id", checkRole("Admin"), controller.update);
32.
33.   // Видалити (Admin)
34.   router.delete("/:id", checkRole("Admin"), controller.delete);
35.
36.   // Бронювання конкретного слота по ID (пацієнт або лікар/адмін з
   patient_id)
37.   router.post("/:scheduleId/book", controller.bookSchedule);
38.
39.   module.exports = router;
```

В.2 - Частина коду для контроллеру: doctorScheduleController.js

```
1.   const {
2.     DoctorSchedule,
3.     Appointment,
4.     Doctor,
5.     Patient,
6.     Hospital
7.   } = require("../models/models");
8.   const ApiError = require("../error/ApiError");
9.   const { Op } = require("sequelize");
10.  const moment = require("moment");
11.
12.  class DoctorScheduleController {
13.
14.    // Розклад лікаря на день (публічний)
15.    async getByDoctorAndDate(req, res, next) {
16.      try {
17.        const { doctorId, date } = req.params;
18.        if (!doctorId || !date)
19.          return next(ApiError.badRequest("Потрібні doctorId i
date"));
20.
21.        const schedules = await DoctorSchedule.findAll({
22.          where: {
23.            doctor_id: doctorId,
24.            appointment_date: date,
25.          },
26.        });
27.
28.        return res.json(schedules);
29.      } catch (e) {
30.        console.error("getByDoctorAndDate error:", e);
31.        return next(
32.          ApiError.internal("Не вдалося отримати розклад на вказану
дату")
33.        );
34.      }
35.    }
36.
37.    // Створення (Admin)
38.    async create(req, res, next) {
39.      try {
40.        if (req.user.role !== "Admin") {
41.          return next(
42.            ApiError.forbidden("Тільки адміністратор може створювати
розклад")
43.          );
44.        }
45.
```

```
46.         const { doctor_id, start_date, end_date, schedule_template } =
req.body;
47.
48.         if (!doctor_id || !start_date || !end_date ||
!schedule_template) {
49.           return next(
50.             ApiError.badRequest(
51.               "Необхідно вказати doctor_id, start_date, end_date та
schedule_template"
52.             )
53.           );
54.         }
55.
56.         const schedulesToCreate = [];
57.
58.         let current = moment(start_date);
59.         const end = moment(end_date);
60.
61.         while (current.isSameOrBefore(end, "day")) {
62.           const dayOfWeek = current.format("dddd");
63.           const template = schedule_template[dayOfWeek];
64.
65.           if (template) {
66.             const { start_time, end_time } = template;
67.             const start = moment(
68.               `${current.format("YYYY-MM-DD")} ${start_time}`,
69.               "YYYY-MM-DD HH:mm"
70.             );
71.             const end = moment(
72.               `${current.format("YYYY-MM-DD")} ${end_time}`,
73.               "YYYY-MM-DD HH:mm"
74.             );
75.
76.             while (start.isBefore(end)) {
77.               const slotEnd = moment(start).add(30, "minutes");
78.
79.               if (slotEnd.isAfter(end)) break;
80.
81.               schedulesToCreate.push({
82.                 doctor_id,
83.                 appointment_date: current.format("YYYY-MM-DD"),
84.                 start_time: start.format("HH:mm"),
85.                 end_time: slotEnd.format("HH:mm"),
86.                 is_booked: false,
87.               });
88.
89.               start.add(30, "minutes");
90.             }
91.           }
92.
```

```
93.           current.add(1, "day");
94.         }
95.
96.         const created = await
DoctorSchedule.bulkCreate(schedulesToCreate);
97.
98.         return res.json({
99.           message: `Успішно створено ${created.length} розкладів`,
100.          created,
101.        });
102.      } catch (e) {
103.        console.error("create (week) error:", e);
104.        return next(ApiError.internal("Не вдалося створити розклад на
тиждень"));
105.      }
106.    }
107.
108.    // Бронювання слота
109.    async bookSchedule(req, res, next) {
110.      try {
111.        const { scheduleId } = req.params;
112.
113.        const schedule = await DoctorSchedule.findByPk(scheduleId);
114.        if (!schedule) return next(ApiError.notFound("Час розкладу не
знайдено"));
115.
116.        if (schedule.is_booked) {
117.          return next(ApiError.badRequest("Цей час вже
забронюваний"));
118.        }
119.
120.        let patient;
121.        if (req.user.role === "Patient") {
122.          patient = await Patient.findOne({ where: { user_id:
req.user.id } });
123.          if (!patient) return next(ApiError.forbidden("Пацієнта не
знайдено"));
124.        } else if (req.body.patient_id) {
125.          patient = await Patient.findByPk(req.body.patient_id);
126.          if (!patient) return next(ApiError.badRequest("Невірний
patient_id"));
127.        } else {
128.          return next(ApiError.badRequest("Необхідно вказати
пацієнта"));
129.        }
130.
131.        const appointment = await Appointment.create({
132.          patient_id: patient.id,
133.          doctor_id: schedule.doctor_id,
134.          doctor_schedule_id: schedule.id,
```

```
135.         appointment_date: schedule.appointment_date,
136.         status: "Scheduled",
137.       });
138.
139.       await schedule.update({ is_booked: true });
140.
141.       return res.json({
142.         message: "Час успішно заброньовано",
143.         appointment,
144.       });
145.     } catch (e) {
146.       console.error("bookSchedule error:", e);
147.       return next(ApiError.internal("Не вдалося забронювати час"));
148.     }
149.   }
150.   async getWorkingHoursByDate(req, res, next) {
151.     try {
152.       const { doctorId, date } = req.params;
153.
154.       if (!doctorId || !date) {
155.         return next(ApiError.badRequest("Потрібні doctorId і
date"));
156.       }
157.
158.       const slots = await DoctorSchedule.findAll({
159.         where: {
160.           doctor_id: doctorId,
161.           appointment_date: date,
162.         },
163.         order: [["start_time", "ASC"]],
164.       });
165.
166.       if (slots.length === 0) {
167.         return res.json({
168.           doctor_id: doctorId,
169.           date,
170.           message: "На цей день немає розкладу",
171.         });
172.       }
173.
174.       return res.json({
175.         doctor_id: doctorId,
176.         date,
177.         start_time: slots[0].start_time,
178.         end_time: slots[slots.length - 1].end_time,
179.       });
180.     } catch (e) {
181.       console.error("getWorkingHoursByDate error:", e);
182.       return next(ApiError.internal("Не вдалося отримати час прийому
лікаря"));
183.     }
184.   }
185. }
186.
187. module.exports = new DoctorScheduleController();
```

ДОДАТОК Г

Початок таблиці Г.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
GET	/api/hospital-medical-services/doctor/:doctorId	Отримати послуги лікаря
GET	/api/hospital-medical-services/:id	Отримати медичну послугу за ID
GET	/api/hospital-medical-services	Отримати всі медичні послуги
POST	/api/hospital-medical-services	Створити нову медичну послугу (лише Admin)
PUT	/api/hospital-medical-services/:id	Оновити медичну послугу(лише Admin)
DELETE	/api/hospital-medical-services/:id	Видалити медичну послугу (лише Admin)
GET	/api/hospitals/unique-names	Отримати унікальні назви закладів
GET	/api/hospitals	Отримати список усіх лікарень
GET	/api/hospitals/:id	Отримати інформацію про заклад за ID
POST	/api/hospitals	Створити новий медичний заклад (Admin)
PUT	/api/hospitals/:id	Оновити дані закладу (Admin)
DELETE	/api/hospitals/:id	Видалити заклад (Admin)
GET	/api/hospital-staff/doctors	Отримати всіх лікарів персоналу (доступно для Admin, Doctor)
GET	/api/hospital-staff/medical-staff	Отримати весь медичний персонал (доступно для Admin, Doctor)
GET	/api/hospital-staff/:id	Отримати працівника за ID (доступно для Admin, Doctor)

Продовження таблиці Г.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
GET	/api/hospital-staff	Отримати всіх працівників (доступно для Admin, Doctor)
POST	/api/hospital-staff	Створити працівника (Admin)
PUT	/api/hospital-staff/:id	Оновити працівника (Admin)
DELETE	/api/hospital-staff/:id	Видалити працівника (Admin)
GET	/api/hospital-staff/by-user/:userId	Отримати працівника за userId
GET	/api/hospital-staff/non-doctors/:hospitalId	Отримати працівника медичний персонал за hospitalId
GET	/api/lab-test-info	Отримати всі довідкові записи про аналізи
GET	/api/lab-test-info/:id	Отримати довідкову інформацію про аналіз за ID
GET	/api/lab-test-info/hospital/:hospitalId	Отримати аналізи доступні в лікарні
POST	/api/lab-test-info	Створити довідкову інформацію про аналіз (Admin)
PUT	/api/lab-test-info/:id	Оновити довідкову інформацію (Admin)
DELETE	/api/lab-test-info/:id	Видалити запис про аналіз (Admin)
GET	/api/lab-tests	Отримати всі результати аналізів
GET	/api/lab-tests/:id	Отримати аналіз за ID
GET	/api/lab-tests/patient/:patientId	Отримати аналізи пацієнта

Продовження таблиці Г.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
GET	/api/lab-tests/doctor/:doctorId	Отримати аналізи, призначені лікарем
GET	/api/lab-tests/status/filter	Фільтрувати аналізи за статусом
GET	/api/lab-tests/:id/pdf	Завантажити PDF аналізу
GET	/api/lab-tests/by-hospital/:hospitalId	Отримати аналізи по лікарні (Admin/Doctor)
PATCH	/api/lab-tests/mark-ready/:id	Позначити аналіз як готовий (Admin/Doctor)
POST	/api/lab-tests	Створити запис про аналіз
PUT	/api/lab-tests/:id	Оновити запис аналізу
DELETE	/api/lab-tests/:id	Видалити запис аналізу
POST	/api/lab-test-schedule/pay-and-book	Оплатити та записатися на аналіз
GET	/api/lab-test-schedule/by-hospital/:hospitalId	Отримати графіки аналізів по лікарні
GET	/api/lab-test-schedule/lab/:labServiceId/date/:date	Отримати розклад аналізів за послугою та датою
GET	/api/lab-test-schedule/working-hours/lab/:hospital_lab_service_id/:date	Отримати час роботи за послугою
GET	/api/lab-test-schedule	Отримати всі графіки аналізів
GET	/api/lab-test-schedule/:id	Отримати графік аналізу за ID

Продовження таблиці Г.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
POST	/api/lab-test-schedule	Створити розклад аналізів (Admin/Doctor)
PUT	/api/lab-test-schedule/:id	Оновити розклад аналізів (Admin/Doctor)
DELETE	/api/lab-test-schedule/:id	Видалити розклад аналізів (Admin/Doctor)
GET	/api/medical-records/:recordId /patient/:patientId/prescriptions	Отримати рецепти за медичною карткою пацієнта
GET	/api/medical-records/patient/:patientId	Отримати всі записи пацієнта
GET	/api/medical-records	Отримати всі медичні записи
GET	/api/medical-records/:id	Отримати медичний запис за ID
POST	/api/medical-records	Створити новий медичний запис
PUT	/api/medical-records/:id	Оновити медичний запис
DELETE	/api/medical-records/:id	Видалити медичний запис
GET	/api/medical-service-info	Отримати всі медичні послуги
GET	/api/medical-service-info/:id	Отримати медичну послугу за ID
GET	/api/medical-service-info/hospital/:hospitalId	Отримати послуги за лікарнею
POST	/api/medical-service-info	Створити медичну послугу (Admin)

Продовження таблиці Г.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
PUT	/api/medical-service-info/:id	Оновити медичну послугу (Admin)
DELETE	/api/medical-service-info/:id	Видалити медичну послугу (Admin)
GET	/api/medical-services/by-doctor/:doctorId	Отримати послуги лікаря
GET	/api/medical-services/hospital/:hospitalId	Отримати послуги за лікарнею
PATCH	/api/medical-services/mark-ready/:id	Позначити послугу як завершену
GET	/api/medical-services/:id/pdf	Завантажити PDF результат
GET	/api/medical-services/patient/:patientId	Отримати послуги пацієнта
GET	/api/medical-services/:id	Отримати медичну послугу за ID
GET	/api/medical-services	Отримати всі медичні послуги
POST	/api/medical-services	Створити медичну послугу
PUT	/api/medical-services/:id	Оновити медичну послугу
DELETE	/api/medical-services/:id	Видалити медичну послугу
POST	/api/medical-service-schedules/book	Записатись та оплатити медичну процедуру
GET	/api/medical-service-schedules/service/:medicalServiceId/date/:date	Отримати розклад послуги за датою

Продовження таблиці Г.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
GET	/api/medical-service-schedules/by-hospital/:hospitalId	Отримати розклад послуг за лікарнею
GET	/api/medical-service-schedules/working-hours/medical/:hospital_medical_service_id/:date	Отримати години роботи для послуги
GET	/api/medical-service-schedules	Отримати всі розклади процедур
GET	/api/medical-service-schedules/:id	Отримати розклад процедури за ID
POST	/api/medical-service-schedules	Створити розклад процедури
PUT	/api/medical-service-schedules/:id	Оновити розклад процедури
DELETE	/api/medical-service-schedules/:id	Видалити розклад процедури
GET	/api/patient/by-hospital/:hospitalId	Отримати пацієнтів лікарні (Doctor, Admin)
PUT	/api/patient/:id/photo	Оновити фото пацієнта (Patient)
GET	/api/patient/by-user/:userId	Отримати пацієнта за user_id
GET	/api/patient/by-doctor/:doctorId	Отримати пацієнтів лікаря (Doctor, Admin)
GET	/api/patient	Отримати всіх пацієнтів (Admin, Doctor)
GET	/api/patient/:id	Отримати одного пацієнта (Patient - свій запис)
POST	/api/patient	Створити пацієнта (Doctor, Admin)

Кінець таблиці Г.1 - Специфікація REST (таблиця виконана самостійно)

HTTP метод	Кінцева точка	Опис
PUT	/api/patient/:id	Оновити пацієнта (Doctor, Admin, Patient)
DELETE	/api/patient/:id	Видалити пацієнта (Doctor, Admin)
POST	/api/paypal/create-order	Створити замовлення (авторизовані користувачі)
POST	/api/paypal/capture-order	Підтвердити замовлення (авторизовані користувачі)
GET	/api/prescription/:id/pdf	Завантажити PDF рецепта
GET	/api/prescription/patient/:patientId/:id	Отримати рецепт пацієнта за ID
GET	/api/prescription/patient/:patientId	Отримати всі рецепти пацієнта
GET	/api/prescription	Отримати всі рецепти
GET	/api/prescription/:id	Отримати рецепт за ID
POST	/api/prescription	Створити новий рецепт
PUT	/api/prescription/:id	Оновити рецепт
DELETE	/api/prescription/:id	Видалити рецепт
GET	/api/review/target/:target_type/:target_id	Отримати відгуки за об'єктом(лікар або лікарня)
GET	/api/review	Отримати всі відгуки
GET	/api/review/:id	Отримати відгук за ID

ДОДАТОК Д

Рисунок Д.1 – ER-модель даних