МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра «Інформаційних систем та мереж»



**Звіт**

**про проходження практики**

Студента 4 курсу групи КН-47  
напряму підготовки Комп’ютерні науки

Іванова В.Б.   
                                  (прізвище та ініціали)   
Керівник практики від університету:

к.т.н. асистент Худий А. М.   
   (посада, вчене звання, науковий ступінь, прізвище та ініціали)

### 

### Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Кількість балів: \_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_

### Кількість балів: \_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали) |
|  | | |

Львів - 2018

**ЗМІСТ**

[Вступ……………………………………………………](#_Toc482457643)……………………………3

[1. ХАРАКТЕРИСТИКА ПІДПРИЄМСТВА 4](#_Toc482457644)

[2. ХАРАКТЕРИСТИКА ОФІСУ Львів2 4](#_Toc482457645)

[3. ПРАКТИКА НА РОБОЧОМУ МІСЦІ 10](#_Toc482457646)

[4. ВИКОНАННЯ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ 1](#_Toc482457647)4

[ВИСНОВКИ 26](#_Toc482457653)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 2](#_Toc482457654)7

**ВСТУП**

Практика студентів є невід’ємною складовою освітньо-професійної програми підготовки студентів вищих навчальних закладів України. Вона становить важливу та обов’язкову ланку в підготовці висококваліфікованих спеціалістів до майбутньої діяльності за фахом. Спрямована на закріплення теоретичних знань, отриманих студентами за час навчання, та набуття, вдосконалення і розширення практичних навичок і умінь в роботі за обраною спеціальністю.

Виробнича практика була пройдена у ТзОВ «СофтСерв Індустрія».

Метою даної практики є:

* поглиблення та закріплення знань, які одержали студенти під час теоретичної підготовки;
* оволодіння сучасними формами та методами роботи з комплексом задач, розв’язуваних на підприємстві з використанням комп’ютерної техніки та інформаційних технологій;
* набуття необхідних навичок у здійсненні операцій технологічного процесу обробки інформації;
* виховання потреби систематичного оновлення своїх знань та їх творчого застосування у практичній діяльності.

Завданнями практики є вивчення організації і етапів розробки програмного продукту, набуття практичних навичок програмування, самостійного вирішення технічних задач на базі сучасних комп’ютеризованих систем, ознайомлення з сучасними технологічними процесами розробки, впровадження та налагодження програмного продукту, з сучасним апаратним та програмним забезпеченням, з організацією праці і економікою виробництва, набуття умінь організаторської роботи по спеціальності, узагальнення і поглиблення знань з вивчених дисциплін.

**1. ХАРАКТЕРИСТИКА ПІДПРИЄМСТВА**

ТзОВ «СофтСерв Індустрія» є однією з провідних ІТ компаній України та Львова зокрема, заснована в 1993 році вихідцями з НУ «Львівська політехніка». За 25 років існування компанія значно збільшила кількість офісів в Україні, Болгарії та США, співпрацює з різноманітними компаніями закордону, зокрема активно розвивається в напрямку турботи про людське здоров’я.

На місці проходження виробничої практики використовується сучасне програмне забезпечення, в якості якого виступають операційні системи комп’ютерів загального призначення, такі як Windows 7, 8.1, 10, та для серверної підтримки, комп’ютерна мережа класу С, програмні засоби для доступу до глобальних комп’ютерних мереж.

Практика була пройдена на офісі Львів2, що знаходиться за адресою м. Львів, вул. Пастернака 5.

Враховуючи вищесказане можна зробити висновок, що заклад забезпечено сучасним апаратним та програмним забезпеченням, отже цілком задовольняє вимогам проходження виробничої практики.

**2. ХАРАКТЕРИСТИКА ОФІСУ Львів2**

Відповідно до тематичного плану і завдань практики необхідно описати наявні апаратні засобів і системне програмне забезпечення бази практики.

У ТзОВ «СофтСерв Індустрія» було виділено робоче місце та комп’ютер, який має наступні характеристики:

Монітор – Рідкокристалічний монітор Samsung LS24E65UPLX/CI (2 шт.)

Системний блок – Technic-Pro Core i5 2.7GHz/B150M-C/1\*16

Операційна система: Windows

Щодо програмного забезпечення, то тут встановлена операційна система Windows 10.. У комп’ютерах також встановлено Microsoft Visual Studio 2017, SourceTreeVisual Studio Code. Серед веб-браузерів використовуються Google Chrome, Mozilla Firefox, Microsoft Edge.

Також у офісі є доступ до мережі Інтернет, функціонує локальна мережа, а також Wi-Fi мережа.

**Операційні системи:**

**Windows 10**

Операційна система (ОС) від компанії Microsoft для персональних комп’ютерів, ноутбуків, планшетів, лептопів-трансформерів і смартфонів. Ця операційна система є альтернативою попередниці Windows 8, для користувачів, звиклих до старого інтерфейсу. У компанії цю версію операційної системи називають останньою, позаяк надалі вона надаватиметься за моделлю «програмне забезпечення як послуга».

Microsoft представила попередню версію Windows 10 у Сан-Франциско 30 вересня 2014 року, очікувані нововведення підтвердились.

Реліз Windows 10 відбувся влітку 2015 року, а саме 29 липня в 190 країнах і 111-ма мовами. Windows 10 є останньою версією Windows. Протягом першого року після виходу системи користувачі мають змогу безкоштовно оновитися до Windows 10 на будь-якому пристрої під керуванням офіційних версій Windows 7.

**Офісні пакети:**

**Microsoft Office 2010**

Microsoft Office 2010 (кодове ім’я Office 14) – назва версії пакету Microsoft Office для Microsoft Windows. Microsoft почала роботу над Office 2010 в 2006 році, коли закінчувалася робота над пакетом Office 12 (який вийшов під назвою Microsoft Office 2007).

У пакет вбудована покращена анти піратська технологія Office Genuine Advantage, що дебютувала в квітні 2008 року. Всі продукти лінійки (крім серверних і OEM) вимагають активації. Офісний пакет, створений корпорацією Microsoft для операційних систем Microsoft Windows, Apple Мас OS X і Apple iOS (на iPad). До складу цього пакету входить програмне забезпечення для роботи з різними типами документів: текстами, електронними таблицями, презентаціями, базами даних тощо.

Microsoft Office також є сервером OLE об’єктів і його функції можуть використовуватися іншими застосунками, а також самими застосунками

Microsoft Office. Підтримує скрипти і макроси, написані на VBA.

**Антивірусне програмне забезпечення:**

**NOD32**

ESET NOD32 забезпечує захист від вірусів, а також від інших загроз, включаючи троянські програми, черв’яки, spywarе. В ESET NOD32 використовується патентована технологія ThreatSense, призначена для виявлення нових загроз, які виникають у реальному часі, шляхом аналізу виконуваних програм на наявність шкідливого коду, що дозволяє попереджати дії авторів шкідливих програм. Антивірусний пакет, що випускається словацькою фірмою ESET.

Перша версія була випущена в кінці 1987 року. NOD32 – це комплексне антивірусне розв’язання задачі захисту в реальному часі.

**Веб-браузери:**

**Google Chrome**

Розроблений компанією Google на основі веб-переглядача з відкритим кодом Chromium та іншого відкритого програмного забезпечення. Його програмна архітектура була створена з чистого аркуша (однак використовуючи інше програмне забезпечення з відкритим кодом, у тому числі компоненти WebKit та Mozilla) з метою задоволення поточних потреб користувачів.

**Internet Explorer**

Графічний веб-оглядач (браузер), який розробляє корпорація Microsoft.

Найперша версія побачила світ 16 серпня 1995 року. Вона була заснована на браузері Mosaic, права на який корпорація Microsoft придбала у компанії Spyglass. Починаючи з версії 3.0, Microsoft поширювала Internet Explorer у складі операційної системи Windows, що дозволило йому швидко збільшити ринкову частку і навіть витіснити провідний колись веб-оглядач Netscape Navigator.

Останніми версіями для Mac OS і Unix-подібних систем були Internet Explorer 5.2.3 і Internet Explorer 5.0 SP1 Beta відповідно. Зараз розробку Internet Explorer для цих систем припинено.

**Графічні редактори:**

**Paint**

Простий растровий графічний редактор компанії Microsoft, який входить до усіх операційних систем Windows, починаючи з перших версій.

У Windows 95 була введена нова версія Paint. Той же інтерфейс продовжує використовуватися в наступних версіях Windows.

Недоліки:

* Відсутня функція зміни яскравості(насиченості), контрасту тощо.
* Немає можливості при зміні розміру вказати бажаний розмір у пікселях (функція з’явилась в Windows 7).
* Неможливо зробити шрифт з правого краю, чи по центру.
* Неможливо зрушувати кордони області виділення (при цьому змінюються пропорції виділеного фрагменту).
* Неможливо повертати фігури на певну градусну міру.

**Adobe Photoshop**

Часто цю програму називають просто Photoshop. В даний час Photoshop доступний на платформах OS X, Windows, в мобільних системах iOS, Windows Phone і Android. Також існує версія Photoshop Express для Windows Phone 8 і 8.1. Зараз у СІЛА йде бета-тестування продукту для Chrome OS. Графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень, і найвідомішим продуктом фірми Adobe.

**3. ПРАКТИКА НА РОБОЧОМУ МІСЦІ**

Практика проходила з 20 серпня по 3 вересня 2018 року. Основним завданням на практику було розроблення проекту медичної реєстратури.

У ТзОВ «СофтСерв Індустрія» були створені належні умови для проходження виробничої практики. Я був ознайомлений з технікою безпеки на робочому місці та загальними правилами поведіки в офісі.

Впродовж проходження практики було набуто практичні навчки роботи по скраму.

Scrum — це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи — ScrumMaster, той хто опікується процесами, веде їх і працює як керівник [проекту](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82_(%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8)), Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog([документація](https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D1%96%D1%8F) запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана. Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту(sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

* Власник Продукту (Product Owner)

Власник Продукту представляє зацікавлені сторони та є голосом клієнта. Він є відповідальним за забезпечення того що команда додає цінність до бізнесу.

* Керівник (ScrumMaster)

Методологія Scrum застосовується за сприяння Scrum-керівника, який є відповідальним за спроможність команди виконати поставлені цілі і вирішення складнощів, які виникають.

* Команда розробників (Scrum Team)

Команда розробників є відповідальною за доставку потенційно готових частин продукту в кінці кожного спринту (the sprint goal). Команда складається з 3-9 людей що виконують роботу (аналізують, виконують дизайн, пишуть код, тестують, готують документацію і таке інше). У Scrum, команда є самокерованою.

Зустрічі які виникають під час розробки:  
**Планування спринта (Sprint Planning Meeting)**

Проходить на початку нової ітерації Спринта.

* Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;
* На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино-годинах;
* Розв'язання задачі не повинне займати більше 12 годин або одного дня. За необхідності задачу розбивають на підзадачі;
* Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;
* Тривалість наради обмежена зверху 4-8 годинами в залежності від тривалості ітерації, досвіду команди тощо;
  + (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;
  + (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog

**Щоденна нарада (Daily Scrum Meeting)**

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

* починається точно вчасно;
* всі можуть спостерігати, але говорять тільки «свині»;
* триває не більш ніж 15 хвилин;
* проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

* Що зроблено з моменту попередньої щоденної наради?
* Що буде зроблено з моменту поточної наради до наступної?
* Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

**Демонстрація (Sprint Review Meeting)**

* Проходить у кінці ітерації (спринта).
* Команда демонструє внесок функціональності до продукту всім зацікавленим особам.
* Залучається максимальна кількість глядачів.
* Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).
* Обмежена 4-ма годинами в залежності від тривалості ітерації і змін у продукті.

**Ретроспектива (Sprint Retrospective)**

* Члени команди висловлюють свою думку про минулий спринт.
* Відповідають на два основних запитання:
* Що було зроблено добре у минулому спринті?
* Що потрібно покращити в наступному?
* Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).
* Обмежена 1-3ма годинами.

Опис першочергових вимог

* Програма призначена для спрощення роботи лікарів, реєстрації пацієнта, встановлення діагнозу та відстежування стану хворого.

Функціональні вимоги:

* Щоб користуватися функціоналом програми, користувачу необхідно зайти в особистий кабінет CRM системи або запустити додаток на смартфоні.

Нефункціональні вимоги

* Вимоги до продукту:
* Клієнтські ПК з ОС Windows 7 і вище.
* Мінімальні апаратні характеристики: процесор – intelPentium, вільна пам’ять – 30mb.
* Зовнішні вимоги:
* Мова інтерфейсу – англійська, українська.
* Зовнішні апаратні засоби: монітор, клавіатура, мишка.

Користувачі системи

* Користувачами систем являються лікар та ПАЦІЄНТ, які взаємодіють між собою за допомогою інтерфейсу
* 7.1. Відношення з іншими програмами

Система являється кросбраузерною та адаптивною. Прекрасно функціонує на всіх пристроях.

**Якість**

* Розроблена система має зручний і зрозумілий, інтерфейс що дозволить швидко пристосуватися до неї і комфортно користуватись без спеціальних навичок.

Реалізація

* Згідно з поставленими вимогами до етапу реалізації, було проведено розробку продукту. До основних результатів даного етапу належить:
* **Розширений документ, що описує вимоги.**

На даному етапі розробки реалізовані ті вимоги, які було поставлено.

* **Прототип системи.**

Протестований прототип, який показує зручність користування.

* **Дизайн системи**, найважливіший етап у створенні системи, оскільки саме він допоможе користувачам вирішувати свої проблеми швидко та зручно .
* **Код**, що складається з протестованих модулів.

**5.** **ТЕХНІЧНА ЧАСТИНА ВИКОНАННЯ ІНДИВІДУАЛЬНОГО ЗАВДАННЯ**

Впродовж проходження практики було розроблено WEB-орієнтовану аплікацію “HoReD”, назва якої означає – медична реєстратура (Hospital Registration Desk). Для написання аплікації було обрано наступний стек технологій – ASP .Net WebAPI для написання серверної частини, React JS для написання клієнтської частини, Bootstrap для створення адаптивного дизайну, MS SQL Server було обрано, як систему управлянні базами даних, Git, як систему контролю версій. Значною перевагою даної аплікації є те, що вона написана за допомогої технологією REST, що являє собою незалежність сервера та клієнта, тобто можна з легкістю використовувати як серверну та клієнтську частину окремо одна від одної до комбінувати з іншими.

При проектуванні системи було визначено, що в ній буде дві ролі - пацієнт та лікар. При входи в систему навіть незареєстрований користувач може побачити розклад лікарів, спочатку обравши професію та лікаря. Ввійти в систему можна, натиснувши на кнопку входу, де потрібно ввести логін та пароль. Після чого пацієнт може зареєструватися на вільну годину до певного лікаря, при цьому відображається тільки ті сеанси, які є вільними встановленою тривалістю сеансу є 30 хвилин(Рис. 1).

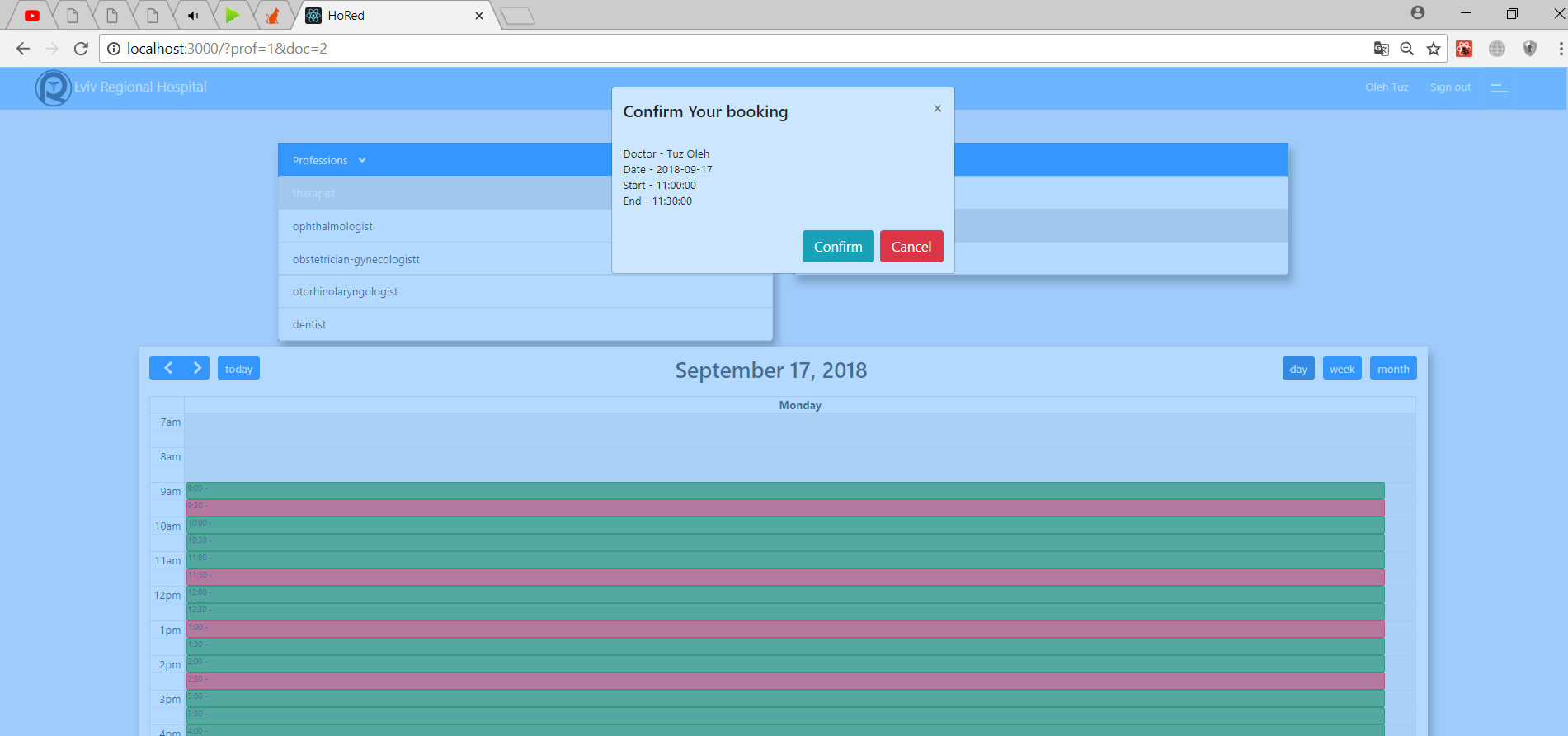


Рис. 1 Початкова сторінка сайту.

Після вибору лікаря та сеансу є форма підтвердження зарезервування сеансу. Після чого лікар, зайшовши на свою сторінку, в робочий календар може побачити прізвище та ім’я пацієнта, який зареєструвався на відповідний сеанс(Рис. 2).

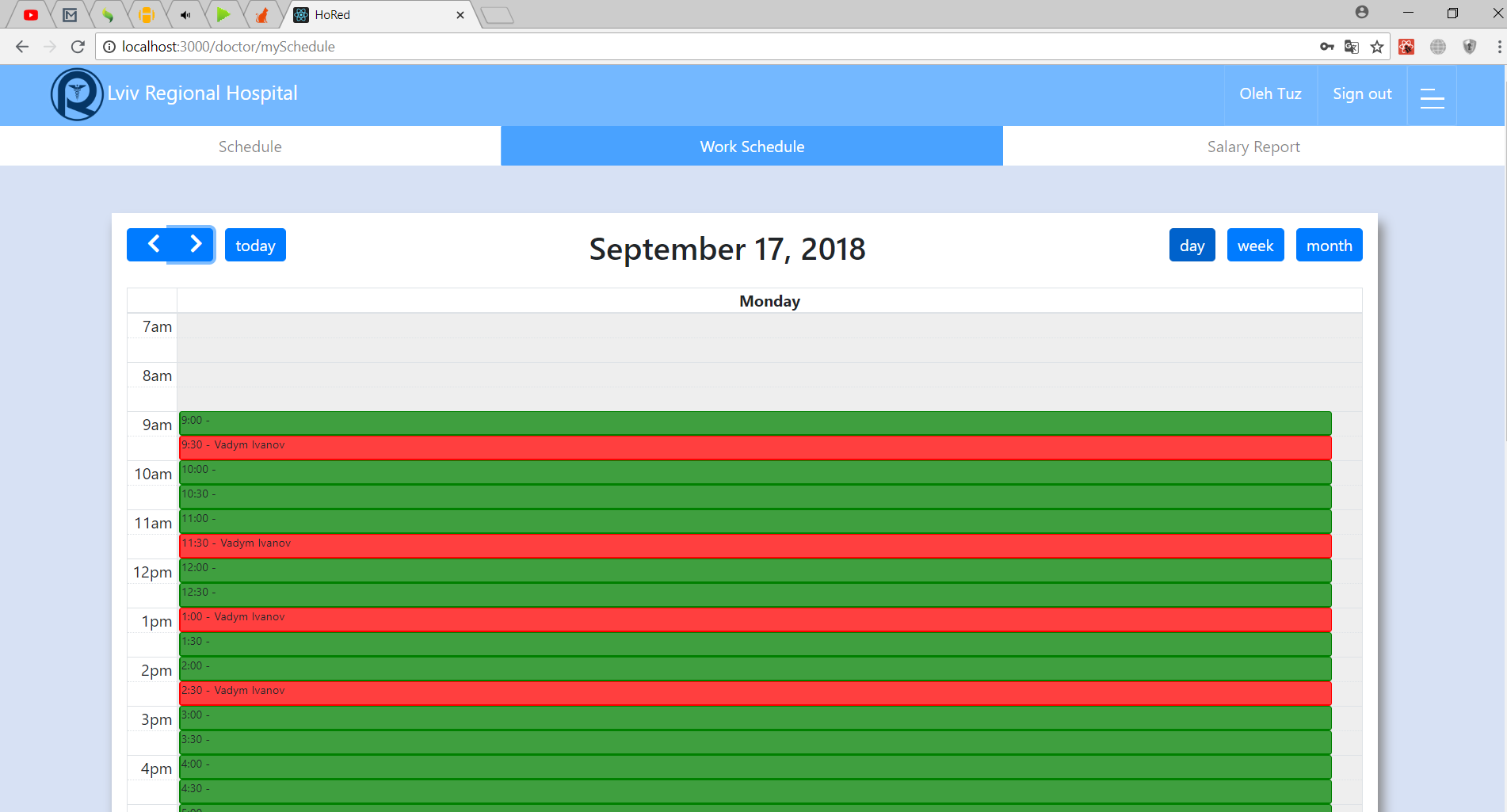


Рис. 2 Робочий календар лікаря.

Після вибору відповідного сеансу лікар переходить на сторінку з інформацією про пацієнта, на якій лікар може додати(Рис.3), закрити діагноз(Рис.4) або алергію, додати опис сеансу та лікування, переглянути історію закритих алергій та діагнозів(Рис.5).

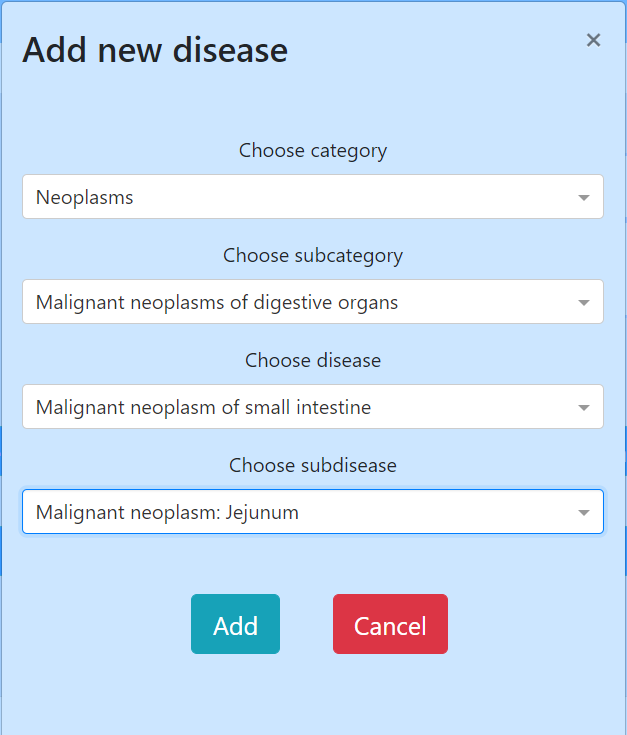


Рис. 3 Додавання нового діагнозу.

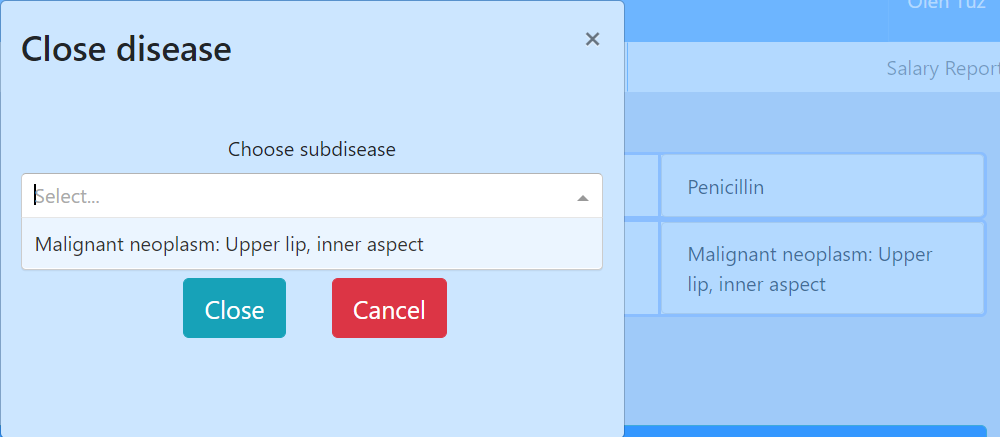


Рис. 4 Закривання активного діагнозу.

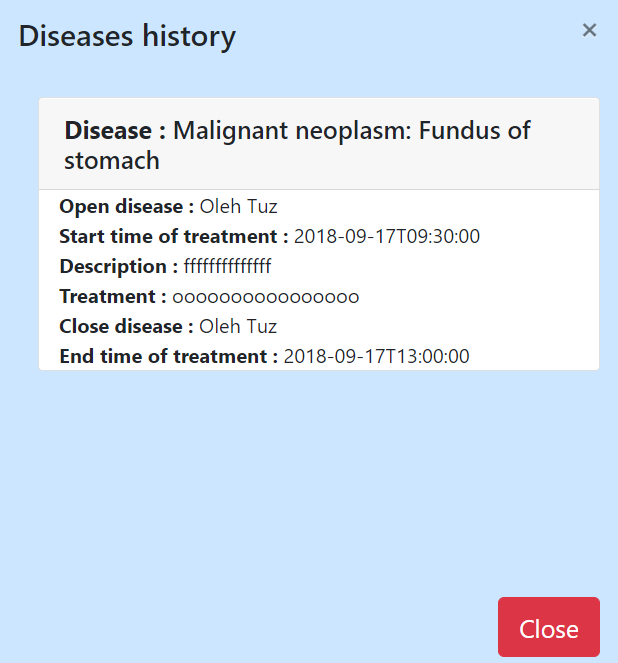


Рис. 5 Історія закритих діагнозів.

Після додання діагнозу пацієнт може зайти на свою сторінку де може побачити інформацію про відвідані сеанси(Рис.6).

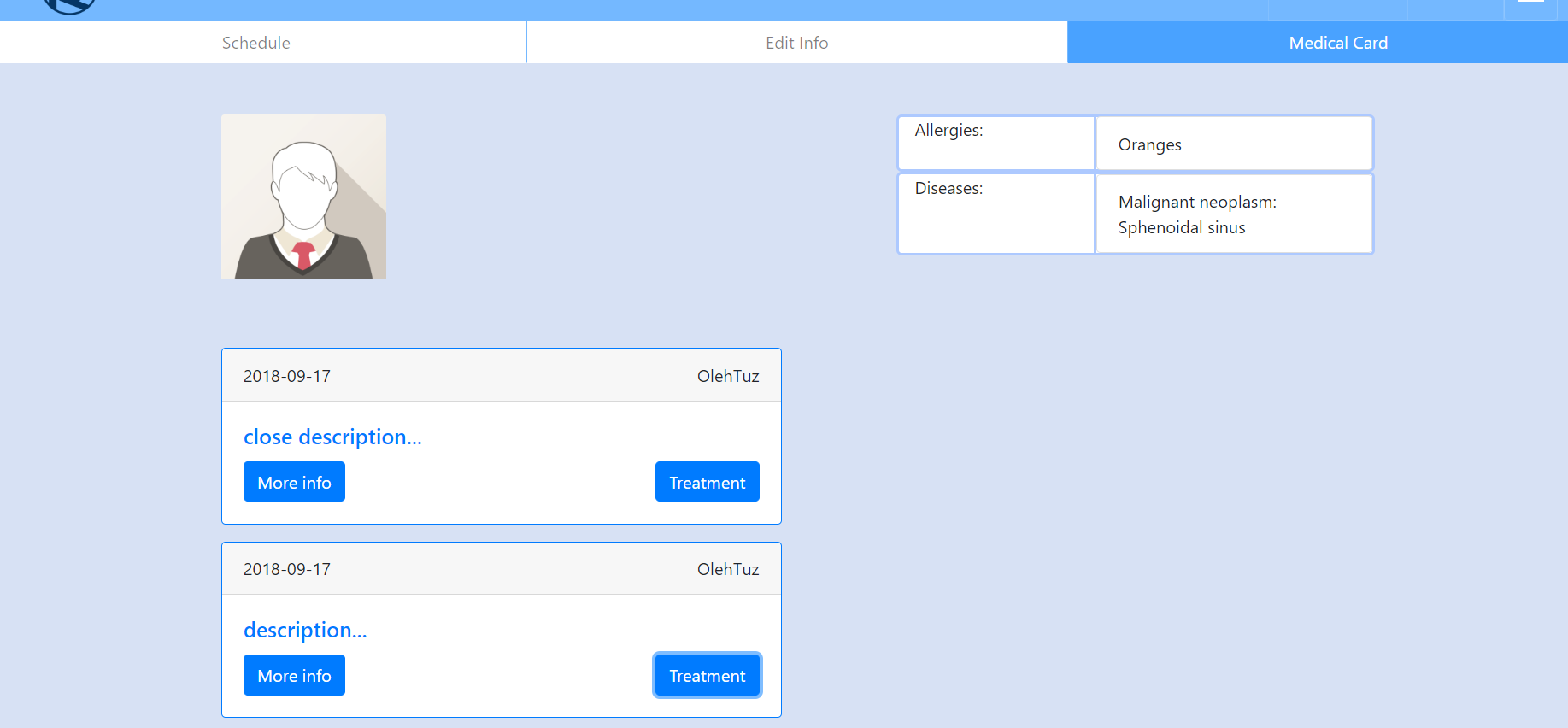


Рис. 6 Історія відвіданих сеансів пацієнта.

**ВИСНОВКИ**

Проходження практики надає професійну підготовку майбутнім спеціалістам, вчить самостійності, ініціативності, набуттю організаційних, дослідницьких навичок, які забезпечують повноцінну роботу випускників за фахом.

Протягом практики було здійснено ознайомлення з різними видами прикладного програмного забезпечення, вдосконалено свої навички. Також вивчено правила внутрішнього розпорядку підприємства; організаційну структуру управління підприємством; функції головних спеціалістів одного із відділів підприємства.

Окрім загальних завдань практики було виконано індивідуальне завдання та набуто практичні навички роботи у написанні сайтів.

Під час практики підібрано необхідну інформацію для придбання навичок практичної діяльності за фахом.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Офіційна документація React JS. [Електронний ресурс]. Режим доступу URL: <https://reactjs.org/> .
2. Документація з використання ASP.NET Web API 2 (C#) [Електронний ресурс]. Режим доступу URL: https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api.
3. Офіційна документація Bootstrap. [Електронний ресурс]. Режим доступу URL: https://getbootstrap.com/.
4. Форум програмістів Електронний ресурс]. Режим доступу URL: https://stackoverflow.com/.
5. ГОСТ 34.201-90. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы**.** Стадии создания. – М.: Изд-во стандартов, 1991. – 16 с.
6. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. – К.: Держстандарт України, 1995. – 36 с.

Додаток А

Код серверної частини сайту

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using Entities;

using Entities.Services;

using HoReD.AuthFilters;

namespace HoReD.Controllers

{

/// <summary>

/// Controller that represents information about patient medical card

/// </summary>

public class PatientDataController : ApiController

{

private readonly IPatientDataService \_patientData;

public PatientDataController(IPatientDataService patientData)

{

\_patientData = patientData;

}

/// <summary>

/// Method that get user id and return his medical card

/// </summary>

/// <param name="id">ID of needed user</param>

/// <returns>Patient Data</returns>

[HttpGet]

[TokenAuthenticate(Role = "patient,doctor")]

public IHttpActionResult GetPatientDataByPatientId(int id)

{

try

{

var result = \_patientData.GetPatientDataById(id);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns active allergies, that current user has

/// </summary>

/// <param name="id">ID of needed user</param>

/// <returns>List of user's allergies names, id and visit</returns>

[HttpGet]

[TokenAuthenticate(Role = "patient,doctor")]

[Route("api/PatientData/ActiveAllergies/{id}")]

public IHttpActionResult GetPatientActiveAllergies(int id)

{

try

{

var result = \_patientData.GetPatientActiveAllergies(id);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns active allergies, that current user don't has

/// </summary>

/// <param name="id">ID of needed user</param>

/// <returns>List of user's allergies names, id</returns>

[HttpGet]

[TokenAuthenticate(Role = "patient,doctor")]

[Route("api/PatientData/NonActiveAllergies/{id}")]

public IHttpActionResult GetPatientNoNActiveAllergies(int id)

{

try

{

var result = \_patientData.GetPatientNonActiveAllergies(id);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns active diseases, that current user has

/// </summary>

/// <param name="id">ID of needed user</param>

/// <returns>List of user's diseases names, id</returns>

[HttpGet]

[TokenAuthenticate(Role = "patient,doctor")]

[Route("api/PatientData/ActiveDiseases/{id}")]

public IHttpActionResult GetPatientActiveDiseases(int id)

{

try

{

var result = \_patientData.GetPatientActiveDiseases(id);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns diseases categories

/// </summary>

/// <returns>List of diseases categories id, name</returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/Categories")]

public IHttpActionResult GetCategories()

{

try

{

var result = \_patientData.GetCategories();

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns diseases subcategories, that current category has

/// </summary>

/// <param name="id">ID of needed category</param>

/// <returns>List of diseases subcategories, id, firstcode, lastcode, name</returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/SubCategories/{id}")]

public IHttpActionResult GetSubCategories(int id )

{

try

{

var result = \_patientData.GetSubCategories(id);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns diseases diseases, that current subcategory has

/// </summary>

/// <param name="id">ID of needed subcategory</param>

/// <returns>List of diseases id, code, name</returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/Diseases/{id}")]

public IHttpActionResult GetDiseases(int id)

{

try

{

var result = \_patientData.GetDiseases(id);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Returns all subdiseases, that current user don't has

/// </summary>

/// <param name="idPatient">ID of needed user</param>

/// <param name="idDisease">ID of needed subdisease</param>

/// <returns>List of user's nonactive subdiseases</returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/SubDiseases/{idPatient}/{idDisease}")]

public IHttpActionResult GetPatientDiseases(int idPatient, int idDisease)

{

try

{

var result = \_patientData.GetPatientSubDiseases(idPatient, idDisease);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Adds allergies for patient

/// </summary>

/// <param name="model">Id patient, starttime of visit and idallergy</param>

/// <returns>Integer: 1 - if allergy added </returns>

[HttpPost]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/AddAllergy")]

public IHttpActionResult AddPatientAllergy(Models.MedicalRecordBindingModel model)

{

try

{

var result = \_patientData.AddPatientAllergy(model.IdPatient, Convert.ToDateTime(model.StartTime), model.Allergy);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Adds disease for patient

/// </summary>

/// <param name="model">Id patient, starttime of visit and iddisease</param>

/// <returns>Integer: 1 - if disease added </returns>

[HttpPost]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/AddDisease")]

public IHttpActionResult AddPatientDisease(Models.MedicalRecordBindingModel model)

{

try

{

var result = \_patientData.AddPatientDisease(model.IdPatient, Convert.ToDateTime(model.StartTime), model.Disease);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Closes disease for patient

/// </summary>

/// <param name="model">Id patient, starttime of visit and iddisease</param>

/// <returns>Integer: 1 - if disease closed </returns>

[HttpPost]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/CloseDisease")]

public IHttpActionResult ClosePatientDisease(Models.MedicalRecordBindingModel model)

{

try

{

var result = \_patientData.ClosePatientDisease(model.IdPatient, model.Disease, Convert.ToDateTime(model.StartTime));

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Closes allergy for patient

/// </summary>

/// <param name="model">Id patient, starttime of visit and iddallergy</param>

/// <returns>Integer: 1 - if allergy closed </returns>

[HttpPost]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/CloseAllergy")]

public IHttpActionResult ClosePatientAllergy(Models.MedicalRecordBindingModel model)

{

try

{

var result = \_patientData.ClosePatientAllergy(model.IdPatient, model.Allergy, Convert.ToDateTime(model.StartTime));

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Adds description and treatment for visit

/// </summary>

/// <param name="model">Id patient, starttime of visit, description and treatment</param>

/// <returns>Integer: 1 - if record added </returns>

[HttpPost]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/AddMedicalRecord")]

public IHttpActionResult AddMedicalRecord(Models.MedicalRecordBindingModel model)

{

try

{

var result = \_patientData.AddMedicalRecord(model.IdPatient, Convert.ToDateTime(model.StartTime), model.Description, model.Treatment);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Get description, treatment, doctor and date for diagnose

/// </summary>

/// <param name="idPatient">ID of needed user</param>

/// <param name="idDisease">ID of needed subdisease</param>

/// <returns></returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/GetDiagnoseInfo/{idPatient}/{idDisease}")]

public IHttpActionResult GetDiagnoseInfo(int idPatient, int idDisease)

{

try

{

var result = \_patientData.GetDiagnoseInfo(idPatient, idDisease);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Get description, treatment, doctor and date for diagnose

/// </summary>

/// <param name="idPatient">ID of needed user</param>

/// <returns></returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/GetClosedDiseaseInfo/{idPatient}")]

public IHttpActionResult GetClosedDiseaseInfo(int idPatient)

{

try

{

var result = \_patientData.GetClosedDiseaseInfo(idPatient);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Get description, treatment, doctor and date for active allergy

/// </summary>

/// <param name="idPatient">ID of needed user</param>

/// <param name="idAllergy">ID of needed allergy</param>

/// <returns></returns>

[HttpGet]

[TokenAuthenticate(Role = "doctor")]

[Route("api/PatientData/GetActiveAllergyInfo/{idPatient}/{idAllergy}")]

public IHttpActionResult GetActiveAllergyInfo(int idPatient, int idAllergy)

{

try

{

var result = \_patientData.GetActiveAllergyInfo(idPatient, idAllergy);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

/// <summary>

/// Get description, treatment, doctor and date for the opening and closing allergy visits

/// </summary>

/// <param name="idPatient">ID of needed user</param>

/// <returns></returns>

[HttpGet]

[TokenAuthenticate(Role = "patient,doctor")]

[Route("api/PatientData/GetClosedAllergiesInfo/{idPatient}")]

public IHttpActionResult GetClosedAllergiesInfo(int idPatient)

{

try

{

var result = \_patientData.GetClosedAllergiesInfo(idPatient);

return Ok(result);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

}

}

Додаток Б

Код клієнтської частини сайту

import React from 'react';

import { Component } from 'react';

import PatientInfo from '../../PatientInfo';

import AllergiesCard from './AllergiesCard';

import AddMedRecord from './../../AddMedRecord';

import Diagnoses from './PatientDiseses/Diagnoses';

import ClosedDiseasesInfo from './../modaldialogs/ClosedDiseasesInfo';

import ClosedAllergiesInfo from './../modaldialogs/ClosedAllergiesInfo';

import './../../../style/Disease.css';

class PatientMedicalCard extends Component {

constructor(){

super();

this.state = {

shouldUpdate: 1,

updateDiseases:1,

updateAllergies:1

}

}

reloadComponent(param){

this.setState({

shouldUpdate: this.state.shouldUpdate + param

})

}

updateDiseases(param){

this.setState({

updateDiseases: this.state.updateDiseases + param

})

}

updateAllergies(param){

this.setState({

updateAllergies: this.state.updateAllergies + param

})

}

updateAllergiesOrDiseasesState(param){

this.setState({

updateDiseases: this.state.updateDiseases + param,

updateAllergies: this.state.updateAllergies + param

})

}

render() {

return(

<div id="mainDiv">

<div className="container">

<PatientInfo PatientId={this.props.match.params.id} shouldUpdate={this.state.shouldUpdate} callback={this.updateAllergiesOrDiseasesState.bind(this)} />

<div className="row mt-5 mx-0">

<div className="col-md-6">

<button type="button" className="btn btn-info btn-lg mb-3 col-12" id="ClosedDiseases" data-toggle="modal" data-target="#ClosedDiseasesInfo">Closed diseases history

</button>

<ClosedDiseasesInfo PatientId={this.props.match.params.id} reload={this.state.updateDiseases}/>

</div>

<div className="col-md-6">

<button type="button" className="btn btn-info btn-lg mb-3 col-12" id="ClosedDiseases" data-toggle="modal" data-target="#ClosedAllergiesInfo">Closed allergies history

</button>

<ClosedAllergiesInfo PatientId={this.props.match.params.id} reload={this.state.updateAllergies}/>

</div>

</div>

<div className="row mx-0 mb-5">

<div className="col-sm-12 col-md-6">

<Diagnoses PatientId={this.props.match.params.id} Visit={this.props.match.params.startDate} callback={this.reloadComponent.bind(this)} reload={this.state.updateDiseases}/>

</div>

<div className="col-sm-12 col-md-6">

<AllergiesCard PatientId={this.props.match.params.id} Visit={this.props.match.params.startDate} callback={this.reloadComponent.bind(this)} reload={this.state.updateAllergies}/>

</div>

</div>

<AddMedRecord PatientId={this.props.match.params.id} Visit={this.props.match.params.startDate}/>

</div>

</div>

);

}

}

export default PatientMedicalCard;

class AddMedRecord extends Component {

constructor(props) {

super(props);

this.state = {

treatment: "",

description: ""

};

this.getText = this.getText.bind(this);

this.SaveAll = this.SaveAll.bind(this);

this.redirectToSchedule = this.redirectToSchedule.bind(this);

}

redirectToSchedule() {

window.location.href = window.location.origin + '/doctor/mySchedule';

}

SaveAll() {

var data = {

IdPatient: this.props.PatientId,

StartTime: this.props.Visit,

Description: this.state.description,

Treatment: this.state.treatment

}

axios({

method: 'post',

url: localStorage.getItem("server\_url") + '/api/PatientData/AddMedicalRecord',

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer ' + localStorage.getItem("accessToken")

},

data: JSON.stringify(data)

})

.then(response => {

notify.show("Successfully saved", "custom", 5000, { background: 'green', text: "#FFFFFF" });

})

.catch(function (response) {

console.log(response);

});

this.redirectToSchedule();

}

getText(t, val) {

if (val == 1) {

this.setState({ description: t });

}

else {

this.setState({ treatment: t });

}

}

render() {

return (

<div>

<div>

<Notifications/>

<VisitDescription getText={this.getText} newTreatment={this.state.treatment} newDescription={this.state.description} />

</div>

<div className="row">

<div className="container">

<div className="btn-group-lg">

<button type="button" className="btn btn-info btn-lg float-left" onClick={this.SaveAll}>Save</button>

<button type="button" className="btn btn-danger btn-lg float-right" onClick={this.redirectToSchedule}>Cancel</button>

</div>

</div>

</div>

</div>

);

}

}

class Diseases extends Component{

constructor(props){

super(props);

this.state = {

professionsArr:[],

id:0,

selectedOption: '',

options: [

{ value: '0', label: '' }

]

}

}

handleChange = (selectedOption) => {

if(selectedOption!==null){

this.setState({ selectedOption });

this.props.callback(selectedOption.value);

}

else{

this.setState({

options: [{ value: '0', label: '' } ]

});

this.props.callback(selectedOption)

}

}

shouldComponentUpdate(nextProps, nextState) {

return (this.state.selectedOption!==nextState.selectedOption || this.props.idSubCategory!==nextProps.idSubCategory || this.state.options!==nextState.options)

}

componentWillUpdate(nextProps, nextState)

{

let \_that=this;

if(this.props.idSubCategory!==nextProps.idSubCategory)

{

this.setState({

selectedOption: null

});

axios({

method: 'get',

url: localStorage.getItem("server\_url") + '/api/PatientData/Diseases/' + nextProps.idSubCategory,

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

'Authorization': 'Bearer ' + localStorage.getItem("accessToken")

}

})

.then(function (response) {

\_that.setState({

options: response.data.map( disease => ({ value: disease.Id, label: disease.Name }))

})

})

\_that.handleChange(null);

}

}

getInitialState () {

        return {

            clearable: true,

        };

}

render() {

return (

<div className="col-sm-12 mt-3 selectdiagnose">

<div className="text-center mb-2">Choose disease</div>

<Select

value={this.state.selectedOption}

name="form-field-name"

onChange={this.handleChange}

options={this.state.options}

clearable={false}

/>

</div>

);

}

}

Diseases.propTypes = {

callback: PropTypes.func

};

import React from 'react';

import { Component } from 'react';

import axios from 'axios';

import DatePicker from 'react-datepicker';

import moment from 'moment';

import 'react-datepicker/dist/react-datepicker.css';

import validator from 'validator';

import PropTypes from 'prop-types';

import $ from 'jquery';

import Alert from 'react-s-alert';

import 'react-s-alert/dist/s-alert-default.css';

import 'react-s-alert/dist/s-alert-css-effects/slide.css';

import Categories from './../PatientMedicalCard/PatientDiseses/Categories';

import SubCategories from './../PatientMedicalCard/PatientDiseses/SubCategories';

import Diseases from './../PatientMedicalCard/PatientDiseses/Diseases';

import SubDiseases from './../PatientMedicalCard/PatientDiseses/SubDiseases';

class AddDisease extends Component {

constructor(props) {

super(props);

this.state = {

idCategory: 0,

idSubCategory: 0,

idDisease: 0,

idSubDisease: 0,

shouldUpdate: 1,

updateChild: 1

}

this.handleAddDisease = this.handleAddDisease.bind(this);

}

getCategoryId(param) {

this.setState({

idCategory: param,

shouldUpdate: this.state.shouldUpdate + 1

})

}

getSubCategoryId(param) {

this.setState({

idSubCategory: param,

shouldUpdate: this.state.shouldUpdate + 1

})

}

getDiseaseId(param) {

this.setState({

idDisease: param,

shouldUpdate: this.state.shouldUpdate + 1

})

}

getSubDiseaseId(param) {

this.setState({

idSubDisease: param,

shouldUpdate: this.state.shouldUpdate + 1

})

}

handleAddDisease() {

var newDisease = {

IdPatient: this.props.PatientId,

StartTime: this.props.Visit,

Disease: this.state.idSubDisease

}

axios({

method: 'post',

url: localStorage.getItem("server\_url") + '/api/PatientData/AddDisease',

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer ' + localStorage.getItem("accessToken")

},

data: JSON.stringify(newDisease)

})

.then(response => {

console.log(response.data);

this.props.callback(response.data);

})

}

handleCancel(){

this.props.callback(1);

}

render() {

return (

<div className="modal fade" id="AddDisease" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div className="modal-dialog" role="document">

<div className="modal-content">

<div className="modal-header">

<h3 className="modal-title" id="exampleModalLabel">

Add new disease

</h3>

<button type="button" className="close" data-dismiss="modal" onClick={() => { this.handleCancel() }} aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div className="modal-body">

<Categories callback={this.getCategoryId.bind(this)} />

<SubCategories idCategory={this.state.idCategory} callback={this.getSubCategoryId.bind(this)} />

<Diseases idSubCategory={this.state.idSubCategory} callback={this.getDiseaseId.bind(this)} />

<SubDiseases idDisease={this.state.idDisease} callback={this.getSubDiseaseId.bind(this)} PatientId={this.props.PatientId} reload={this.props.reload} />

<div className="row mb-3 mt-5 justify-content-center">

<div className="col-sm-3 col-6 text-center">

<button type="button" className="btn btn-info btn-lg mb-3" data-dismiss="modal" onClick={() => { this.handleAddDisease() }}>Add

</button>

</div>

<div className="col-sm-3 col-6 text-center" >

<button type="button" className="btn btn-danger btn-lg" data-dismiss="modal" onClick={() => { this.handleCancel() }}>Cancel

</button>

</div>

</div>

</div>

<div className="modal-footer">

</div>

</div>

</div>

</div>

)

}

}

AddDisease.propTypes = {

callback: PropTypes.func

};

export default AddDisease;

class Calendar extends React.Component{

constructor(props){

super(props);

this.state = { idDoc: 0,

startPeriod: '',

endPeriod: '',

startTime:'',

endTime:'',

shouldUpdate: 1,

bookingModalMessage: 'This time is not available for booking. Choose another time slot, please.'

};

this.handleSubmitBooking=this.handleSubmitBooking.bind(this);

}

handleCloseModal()

{

this.setState({

bookingModalMessage: 'This time is not available for booking. Choose another time slot, please.'

})

}

handleSubmitBooking() {

var bookingEvent = {

IdDoctor: this.state.idDoc,

IdPatient: localStorage.getItem("currentUserId"),

startDateTime: this.state.startTime,

endDateTime:this.state.endTime

}

axios({

method: 'post',

url: localStorage.getItem("server\_url") + '/api/Schedule',

headers: {

'Content-Type': 'application/json',

'Authorization': 'Bearer ' + localStorage.getItem("accessToken")

},

data: JSON.stringify(bookingEvent)

})

.then((response => {

// 1 - booking went fine

// 0 - booking is outdated (start time <= now)

// -1 - doctor already has event on that time slot

// -2 - patient already has event on that time slot

// -3 - doctor and patient is the same person

// -4 - start time >= end tame

// -5 - limit for booking exceeded

var message

switch (response.data) {

case 0:

message = "Booking is outdated.";

break;

case 1:

message = "You have successfully booked.";

break;

case -1:

message = "Sorry, this time slot is not available for booking anymore. Choose another one, please.";

break;

case -2:

message = "Sorry, You have allready booked this time slot to an another doctor. Choose another one, please.";

break;

case -3:

message = "This booking is not availible for You.";

break;

case -5:

message = "Sorry, limit for booking for this day exceeded. Please, choose another day.";

break;

}

this.setState({

bookingModalMessage: message

})

$("#bookingInfoModalButton").trigger("click");

this.setState({

shouldUpdate: this.state.shouldUpdate + 1

})

}))

}

saveCurrentDayStartEnd(start, end){

var url\_string = window.location.href;

var url = new URL(url\_string);

this.setState({

startPeriod: start,

endPeriod: end

})

if (url.search != '') {

var Doctor = url.searchParams.get("doc");

this.setState({

idDoc :Doctor

})

}

}

saveCurrentTimeStartEnd(start, end){

this.setState({

endTime: end,

startTime: start

})

}

setStateIdDoc(){

var url\_string = window.location.href;

var url = new URL(url\_string);

var Doctor = url.searchParams.get("doc");

this.setState({

idDoc :Doctor

})

}

componentWillMount(){

this.setStateIdDoc();

}

componentDidMount(){

this.setStateIdDoc();

var \_that = this;

$('#calendar').fullCalendar('changeView', 'agendaDay');

$(document).ready(function() {

$('#calendar').fullCalendar({

eventLimit:true,

theme: true,

header: {

left: 'prev,next today',

center: 'title',

right: 'agendaDay, agendaWeek, month',

},

defaultView: "agendaDay",

selectable: false,

editable: false,

themeSystem: 'bootstrap4',

allDaySlot: false,

lazyFetching: true,

minTime: '07:00:00',

blocked: true,

eventStartEditable : false,

droppable : false,

viewRender: function(view){

var view = $('#calendar').fullCalendar('getView');

localStorage.setItem("startPeriod", view.intervalStart.format())

localStorage.setItem("endPeriod", view.intervalEnd.format())

\_that.saveCurrentDayStartEnd(view.intervalStart.format(), view.intervalEnd.format())

},

select: function(start, end) {

end = $.fullCalendar.moment(start);

end.add(30, 'minutes');

$('#calendar').fullCalendar('renderEvent',

{

start: start,

end: end,

allDay: false,

},

true

);

$('#calendar').fullCalendar('unselect');

},

eventClick: function(event, jsEvent, view ) {

// need button because issue related with opening modal from fullcalendar

if (event.selectable) {

if(localStorage.getItem("currentUserFirstName") !== null)

{

\_that.saveCurrentTimeStartEnd(event.start.\_i, event.end.\_i);

$("#modButton").trigger("click");

}else{

$("#preventUnauthorizedBookingButton").trigger("click");

}

} else {

$("#bookingInfoModalButton").trigger("click");

}

},

})

});

}

addEvents(newEvents, isMonth){

newEvents.map(event => {

$($('#calendar').fullCalendar('getView').el[0]).find('.fc-day[data-date=' + event.start.slice(0, 10)+ ']');

})

$('#calendar').fullCalendar('addEventSource', newEvents);

}

shouldComponentUpdate(nextProps, nextState) {

return (( this.state!==nextState)

|| (this.props.idDoctor!==nextProps.idDoctor));

}

componentWillUpdate(nextProps, nextState){

if( (this.state.shouldUpdate===nextState.shouldUpdate && (this.state.idDoc!== nextState.idDoc)) || (this.props.idDoctor!==nextProps.idDoctor))

{

this.setStateIdDoc();

}

var getData = (((this.state.startPeriod!== nextState.startPeriod) ||

(this.state.endPeriod!== nextState.endPeriod) ||

(this.state.idDoc!== nextState.idDoc) ||

(this.state.shouldUpdate!==nextState.shouldUpdate)) &&

(nextState.idDoc != null));

if(getData){

$('#calendar').fullCalendar( 'removeEvents');

var isMonth;

if($('#calendar').fullCalendar('getView').name === 'month')

isMonth = true;

else

isMonth = false;

axios({

method: 'get',

url: localStorage.getItem("server\_url")+'/DoctorEvents/' + nextState.idDoc +'/' + nextState.startPeriod+'/' + nextState.endPeriod,

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

'Authorization': 'Bearer ' + localStorage.getItem("accessToken")

}

})

.then(response => {

var col;

var building = $.map(response.data, function(event){

var isSelectable = false;

if($('#calendar').fullCalendar('getView').name==='month')

{

event.isFake?col = 'green': col = 'red';

return{

start: event.dateTime[0],

end: event.dateTime[0],

rendering: 'background',

color : col,

}

}else {

event.isFake ? (col = 'green', isSelectable = true):(col = 'red', isSelectable = false);

if(new Date(event.dateTime[0]+'T'+event.dateTime[1]) <= (new Date()))

{

return{

start: event.dateTime[0]+'T'+event.dateTime[1],

end: event.dateTime[0]+'T'+event.dateTime[2],

rendering: 'background',

color : col,

}

}else{

return{

start: event.dateTime[0]+'T'+event.dateTime[1],

end: event.dateTime[0]+'T'+event.dateTime[2],

selectable: isSelectable,

color : col,

}

}

}

})

this.addEvents(building, isMonth);

})

}

}

render(){

var doctor;

doctor = $("#doc"+this.state.idDoc).text();

if($('#nameDoc').text()==="")

doctor = $("#doc"+this.state.idDoc).text();

else

doctor = $("#nameDoc").text();

return (<div>

<div id = "calendar">

<button data-toggle="modal" data-target="#confirmModal" id = "modButton" style={{display: "none"}}>

</button>

<button data-toggle="modal" data-target="#BookingInfoModal" id = "bookingInfoModalButton" style={{display: "none"}}>

</button>

<button data-toggle="modal" data-target="#ModalToPreventUnauthorizedBooking" id = "preventUnauthorizedBookingButton" style={{display: "none"}}>

</button>

</div>

<div className="modal fade" id="confirmModal">

<div className="modal-dialog">

<div className="modal-content">

<div className="modal-header">

<h4 className="modal-title" id="mModalLabel">Confirm Your booking</h4>

<button type="button" className="close" data-dismiss="modal"><span aria-hidden="true">&times;</span><span className="sr-only">Cancel</span></button>

</div>

<div className="modal-body">

Doctor - {doctor}<br/>

Date - {this.state.startTime.slice(0, 10)}<br/>

Start - {this.state.startTime.slice(-8)}<br/>

End - {this.state.endTime.slice(-8)}<br/>

</div>

<div className="modal-footer">

<button type="button" className="btn btn-info btn-lg" data-dismiss="modal" onClick={() =>{this.handleSubmitBooking()}}>Confirm</button>

<button type="button" className="btn btn-danger btn-lg" data-dismiss="modal">Cancel</button>

</div>

</div>

</div>

</div>

<div className="modal fade" id="BookingInfoModal">

<div className="modal-dialog">

<div className="modal-content">

<div className="modal-header">

<h4 className="modal-title" id="mModalLabel">{this.state.bookingModalMessage}</h4>

<button type="button" className="close" data-dismiss="modal" onClick={() =>{this.handleCloseModal()}}><span aria-hidden="true">&times;</span><span className="sr-only">Close</span></button>

</div>

<div className="modal-footer">

<button type="button" className="btn btn-info btn-lg" data-dismiss="modal" onClick={() =>{this.handleCloseModal()}}>Ok</button>

</div>

</div>

</div>

</div>

<div className="modal fade" id="ModalToPreventUnauthorizedBooking">

<div className="modal-dialog">

<div className="modal-content">

<div className="modal-header">

<h4 className="modal-title" id="mModalLabel">Booking is available only for authorized users. Please, sign up or sign in.</h4>

<button type="button" className="close" data-dismiss="modal"><span aria-hidden="true">&times;</span><span className="sr-only">Close</span></button>

</div>

<div className="modal-footer">

<button type="button" className="btn btn-info btn-lg" data-dismiss="modal">Ok</button>

</div>

</div>

</div>

</div>

</div>)

}

}

class DoctorCalendar extends React.Component {

constructor(props) {

super(props);

this.state = {

idDoc: 0,

startPeriod: '',

endPeriod: '',

events: [],

startTime: '',

endTime: '',

idPatient: 0,

namePatient: ""

};

this.setId = this.setId.bind(this);

}

setId(id, name, startTime) {

this.setState({

idPatient: id,

namePatient: name,

startTime: startTime

});

}

saveCurrentDayStartEnd(start, end) {

this.setState({

startPeriod: start,

endPeriod: end,

idDoc: localStorage.getItem("currentUserId")

})

}

saveCurrentTimeStartEnd(start, end) {

this.setState({

endTime: end,

startTime: start

})

}

componentDidMount() {

var \_that = this;

$('#calendar').fullCalendar('changeView', 'agendaDay');

$(document).ready(function () {

$('#calendar').fullCalendar({

eventLimit: true,

theme: true,

header: {

left: 'prev,next today',

center: 'title',

right: 'agendaDay, agendaWeek, month',

},

defaultView: "agendaDay",

selectable: false,

editable: false,

themeSystem: 'bootstrap4',

allDaySlot: false,

lazyFetching: true,

minTime: '07:00:00',

blocked: true,

eventStartEditable: false,

droppable: false,

viewRender: function (view) {

var view = $('#calendar').fullCalendar('getView');

localStorage.setItem("startPeriod", view.intervalStart.format())

localStorage.setItem("endPeriod", view.intervalEnd.format())

\_that.saveCurrentDayStartEnd(view.intervalStart.format(), view.intervalEnd.format())

},

eventClick: function (event, jsEvent, view) {

if (event.selectable) {

$("#modButton").trigger("click");

} else {

\_that.saveCurrentTimeStartEnd(event.start.\_i, event.end.\_i);

\_that.setId(event.patientId, event.patientName, event.start.\_i);

$("#blockClickButton").trigger("click");

}

},

})

});

}

addEvents(newEvents, isMonth) {

var events

newEvents.map(event => {

$($('#calendar').fullCalendar('getView').el[0]).find('.fc-day[data-date=' + event.start.slice(0, 10) + ']');

})

$('#calendar').fullCalendar('addEventSource', newEvents);

}

shouldComponentUpdate(nextProps, nextState) {

return ((this.state.startPeriod !== nextState.startPeriod)

|| (this.state.endPeriod !== nextState.endPeriod)

|| (this.state.idDoc !== nextState.idDoc)

|| (this.state.startTime !== nextState.startTime)

|| (this.state.endTime !== nextState.endTime)

|| (this.state.idPatient !== nextState.idPatient)

|| (this.state.namePatient !== nextState.namePatient));

}

componentWillUpdate(nextProps, nextState) {

var getData = (this.state.startPeriod !== nextState.startPeriod)

|| (this.state.endPeriod !== nextState.endPeriod)

|| (this.state.idDoc !== nextState.idDoc);

if (getData) {

$('#calendar').fullCalendar('removeEvents');

var isMonth;

if ($('#calendar').fullCalendar('getView').name == 'month')

isMonth = true;

else

isMonth = false;

axios({

method: 'get',

url: localStorage.getItem("server\_url") + '/DoctorEventsForDoctor/' + nextState.idDoc + '/' + nextState.startPeriod + '/' + nextState.endPeriod,

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

'Authorization': 'Bearer ' + localStorage.getItem("accessToken")

}

})

.then(response => {

var col;

var building = $.map(response.data, function (event) {

var isSelectable = false;

if ($('#calendar').fullCalendar('getView').name == 'month') {

if (event.isFake) {

col = 'green';

} else {

col = 'red';

}

return {

start: event.dateTime[0],

end: event.dateTime[0],

selectable: isSelectable,

rendering: 'background',

color: col,

}

} else {

if (event.isFake) {

col = 'green';

isSelectable = true;

} else {

col = 'red';

isSelectable = false;

patientName: event.PatientName;

patientId: event.PatientId;

}

return {

start: event.dateTime[0] + 'T' + event.dateTime[1],

end: event.dateTime[0] + 'T' + event.dateTime[2],

color: col,

selectable: isSelectable,

title: event.PatientName,

patientId: event.PatientId,

patientName: event.PatientName

}

}

})

this.addEvents(building, isMonth);

})

}

}

showError(param) {

let myColor = { background: '#FF0000', text: "#FFFFFF" };

notify.show("There is already record for this visit", "custom", 1000, myColor);

}

render() {

let content;

content =

<div>

<Notifications />

<div className="row justify-content-center" id="doctorCalendarDiv">

<div className="col-sm-11 col-md-10 mt-5" id="calendarDiv">

<div id="calendar">

<button data-toggle="modal" data-target="#mModal" id="modButton" style={{ display: "none" }}>

</button>

<button data-toggle="modal" data-target="#BlockClickModal" id="blockClickButton" style={{ display: "none" }}>

</button>

</div>

</div>

</div>

<div className="modal fade" id="mModal">

<div className="modal-dialog">

<div className="modal-content">

<div className="modal-header">

<h4 className="modal-title" id="mModalLabel">This session is free.</h4>

<button type="button" className="close" data-dismiss="modal"><span aria-hidden="true">&times;</span><span className="sr-only">Cancel</span></button>

</div>

<div className="modal-footer">

<button type="button" className="btn btn-danger btn-lg" data-dismiss="modal">Cancel</button>

</div>

</div>

</div>

</div>

<InfoSchedule idPatient={this.state.idPatient} startTime={this.state.startTime} callback={this.showError.bind(this)} />

</div>

return <div>{content}</div>

}

}

Додаток В

Скрипти SQL

CREATE TABLE ROLES(IDRoles INT PRIMARY KEY NOT NULL IDENTITY(1,1),

ROLENAME NVARCHAR(15));

CREATE TABLE USERS (IDUsers INT PRIMARY KEY NOT NULL IDENTITY(1,1),

FIRSTNAME NVARCHAR(30) NOT NULL,

LASTNAME NVARCHAR(30) NOT NULL,

IDROLE INT FOREIGN KEY REFERENCES ROLES(IDRoles),

--ROLE NVARCHAR(15) NOT NULL, -- OR ROLE INT 1 - ADMIN, 2 - DOCTOR, 3 - PATIENT

LOGIN NVARCHAR(15) NOT NULL,

PASSWORD NVARCHAR(30) NOT NULL,

EMAIL NVARCHAR(40) NOT NULL);

CREATE TABLE PROFESSIONS(IDProfessins INT PRIMARY KEY NOT NULL IDENTITY(1,1),

PROFESSINNAME NVARCHAR(30) UNIQUE,

ISSTATIC BIT);

CREATE TABLE DOCTORS (IDDoctors INT PRIMARY KEY NOT NULL FOREIGN KEY REFERENCES USERS(IDUsers),

IDPROFESSION INT FOREIGN KEY REFERENCES PROFESSIONS(IDProfessins),

HOURSTART TIME(0),

HOURFINISH TIME(0),

DATEOFEMPLOYING DATE,

IMAGEDOC NVARCHAR(100));

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'SCHEDULE')

DROP TABLE SCHEDULE

CREATE TABLE SCHEDULE(

ID INT PRIMARY KEY IDENTITY(1,1),

IDDOCTOR INT NOT NULL FOREIGN KEY REFERENCES DOCTORS(IDDOCTORS),

IDPATIENT INT NOT NULL FOREIGN KEY REFERENCES USERS(IDUSERS),

START\_DATETIME DATETIME NOT NULL,

END\_DATETIME DATETIME NOT NULL

);

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'INSERT\_SCHEDULE\_RECORD')

DROP PROC INSERT\_SCHEDULE\_RECORD

GO

CREATE PROCEDURE INSERT\_SCHEDULE\_RECORD

@IDDOCTOR INT, @IDPATIENT INT, @START\_DATETIME DATETIME, @END\_DATETIME DATETIME

AS

BEGIN

INSERT INTO SCHEDULE (IDDOCTOR, IDPATIENT, START\_DATETIME, END\_DATETIME) VALUES(@IDDOCTOR, @IDPATIENT, @START\_DATETIME, @END\_DATETIME);

END;

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'DOCTORS\_RULES\_SET')

BEGIN

DROP TABLE DOCTORS\_RULES\_SET;

END

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'RULESETS')

BEGIN

DROP TABLE RULESETS;

END

IF EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'RULES')

BEGIN

DROP TABLE RULES

END

BEGIN

CREATE TABLE RULES (

ID INT IDENTITY(1,1) PRIMARY KEY,

NAME NVARCHAR(50) UNIQUE NOT NULL,

HOUR\_START TIME NOT NULL,

HOUR\_END TIME NOT NULL,

PERIOD\_START DATE NOT NULL,

PERIOD\_END DATE NOT NULL,

INCLUSIVE BIT NOT NULL DEFAULT 1,

SUNDAY BIT NOT NULL DEFAULT 0,

MONDAY BIT NOT NULL DEFAULT 1,

TUESDAY BIT NOT NULL DEFAULT 1,

WEDNESDAY BIT NOT NULL DEFAULT 1,

THIRSDAY BIT NOT NULL DEFAULT 1,

FRIDAY BIT NOT NULL DEFAULT 1,

SATURDAY BIT NOT NULL DEFAULT 0,

CONSTRAINT PROPER\_TIME\_FRAME CHECK (HOUR\_END > HOUR\_START AND PERIOD\_END >= PERIOD\_START)

);

END;

BEGIN

CREATE TABLE RULESETS (

ID INT IDENTITY(1,1) PRIMARY KEY,

DOCTOR\_ID INT FOREIGN KEY REFERENCES DOCTORS(IDDoctors) ON DELETE CASCADE,

RULE\_ID INT FOREIGN KEY REFERENCES RULES(ID) ON DELETE CASCADE

)

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_RULES')

DROP PROC GET\_RULES;

GO

CREATE PROCEDURE GET\_RULES

AS

BEGIN

SELECT \* FROM RULES;

END;

GO

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'ADD\_RULE')

DROP PROC ADD\_RULE;

GO

CREATE PROCEDURE ADD\_RULE

@NAME NVARCHAR(50), @HOUR\_START TIME, @HOUR\_END TIME, @PERIOD\_START DATE, @PERIOD\_END DATE, @INCLUSIVE BIT,

@SUNDAY BIT, @MONDAY BIT, @TUESDAY BIT, @WEDNESDAY BIT,

@THIRSDAY BIT, @FRIDAY BIT,@SATURDAY BIT

AS

BEGIN

INSERT INTO RULES VALUES (@NAME, @HOUR\_START, @HOUR\_END, @PERIOD\_START, @PERIOD\_END, @INCLUSIVE, @SUNDAY, @MONDAY, @TUESDAY, @WEDNESDAY, @THIRSDAY, @FRIDAY, @SATURDAY);

END;

GO

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'UPDATE\_RULE')

DROP PROC UPDATE\_RULE;

GO

CREATE PROCEDURE UPDATE\_RULE

@ID INT,@NAME NVARCHAR(50), @HOUR\_START TIME, @HOUR\_END TIME, @PERIOD\_START DATE, @PERIOD\_END DATE, @INCLUSIVE BIT,

@SUNDAY BIT, @MONDAY BIT, @TUESDAY BIT, @WEDNESDAY BIT,

@THIRSDAY BIT, @FRIDAY BIT,@SATURDAY BIT

AS

BEGIN

IF ((SELECT COUNT(\*) FROM RULES WHERE ID = @ID) = 1)

BEGIN

UPDATE RULES SET NAME=@NAME, HOUR\_START=@HOUR\_START, HOUR\_END=@HOUR\_END, PERIOD\_START=@PERIOD\_START,

PERIOD\_END=@PERIOD\_END, INCLUSIVE=@INCLUSIVE, SUNDAY=@SUNDAY, MONDAY=@MONDAY, TUESDAY=@TUESDAY, WEDNESDAY=@WEDNESDAY, THIRSDAY=@THIRSDAY, FRIDAY=@FRIDAY, SATURDAY=@SATURDAY

WHERE ID = @ID;

SELECT 1;

END;

ELSE

SELECT 0;

END;

GO

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'DELETE\_RULE')

DROP PROC GET\_RULES;

GO

CREATE PROCEDURE DELETE\_RULE

@ID INT

AS

BEGIN

IF ((SELECT COUNT(\*) FROM RULES WHERE ID=@ID) = 1)

BEGIN

DELETE FROM RULES WHERE ID = @ID;

SELECT 1;

END;

ELSE

SELECT 0;

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_DOCTORS\_WITH\_RULE')

DROP PROC GET\_DOCTORS\_WITH\_RULE;

GO

CREATE PROCEDURE GET\_DOCTORS\_WITH\_RULE

@ID int

AS

BEGIN

SELECT \* FROM RULESETS WHERE RULE\_ID = @ID;

END;

GO

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'ASSIGN\_DOCTOR\_TO\_RULE')

DROP PROC ASSIGN\_DOCTOR\_TO\_RULE;

GO

CREATE PROCEDURE ASSIGN\_DOCTOR\_TO\_RULE

@DOCTOR\_ID int, @RULE\_ID int

AS

BEGIN

INSERT INTO RULESETS VALUES (@DOCTOR\_ID,@RULE\_ID);

END;

GO

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'DISMISS\_DOCTOR\_FROM\_RULE')

DROP PROC DISMISS\_DOCTOR\_FROM\_RULE;

GO

CREATE PROCEDURE DISMISS\_DOCTOR\_FROM\_RULE

@ID int

AS

BEGIN

DELETE FROM RULESETS WHERE ID=@ID;

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_DOCTOR\_SCHEDULE\_BOOKED')

DROP PROC GET\_DOCTOR\_SCHEDULE\_BOOKED;

GO

CREATE PROCEDURE GET\_DOCTOR\_SCHEDULE\_BOOKED

@IDDOCTOR INT, @PERIOD\_START DATETIME, @PERIOD\_END DATETIME

AS

BEGIN

SELECT IDPATIENT, START\_DATETIME, END\_DATETIME FROM SCHEDULE

WHERE @IDDOCTOR = IDDOCTOR AND (START\_DATETIME BETWEEN @PERIOD\_START AND @PERIOD\_END) AND (END\_DATETIME BETWEEN @PERIOD\_START AND @PERIOD\_END);

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_MEDICAL\_CARD\_BY\_USER\_ID')

DROP PROC GET\_MEDICAL\_CARD\_BY\_USER\_ID

GO

CREATE PROCEDURE GET\_MEDICAL\_CARD\_BY\_USER\_ID @ID\_USER INT, @ELEMENT\_COUNT INT, @PAGE\_NUMBER INT AS

BEGIN

WITH NUMBERED\_MY\_TABLE AS

(

SELECT MEDICAL\_CARD.ID AS "CARD\_ID", MEDICAL\_CARD.DESCRIPTION, MEDICAL\_CARD.CURE, USERS.FIRSTNAME AS DOCTOR\_FIRSTNAME, USERS.LASTNAME AS DOCTOR\_LASTNAME, SCHEDULE.IDPATIENT, SCHEDULE.START\_DATETIME AS SESSION\_START\_DATETIME, SCHEDULE.ID AS "ID\_VISIT", SUBDISEASES.SUBDISEASENAME, ROW\_NUMBER() OVER (ORDER BY SCHEDULE.START\_DATETIME) AS ROWNUMBER

FROM MEDICAL\_CARD

INNER JOIN SCHEDULE ON MEDICAL\_CARD.ID\_VISIT = SCHEDULE.ID

LEFT JOIN SUBDISEASES ON MEDICAL\_CARD.ID\_DISEASE = SUBDISEASES.SUBDISEASEID

INNER JOIN USERS ON IDDOCTOR= USERS.IDUsers

WHERE (SCHEDULE.IDPATIENT = 1)

)

SELECT CARD\_ID, DESCRIPTION, CURE, DOCTOR\_FIRSTNAME, DOCTOR\_LASTNAME, IDPATIENT, SESSION\_START\_DATETIME, ID\_VISIT, SUBDISEASENAME

FROM NUMBERED\_MY\_TABLE

WHERE ROWNUMBER BETWEEN (@PAGE\_NUMBER-1) \* @ELEMENT\_COUNT + 1 AND @ELEMENT\_COUNT \* @PAGE\_NUMBER

ORDER BY SESSION\_START\_DATETIME;

END

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name ='ADD\_DISEASE')

BEGIN

DROP PROC ADD\_DISEASE

END

GO

CREATE PROCEDURE ADD\_DISEASE

@ID\_USER INT,@DATE\_VISIT DATETIME,@ID\_DISEASE INT

AS

BEGIN

IF EXISTS(SELECT \* FROM PATIENT\_DISEASES INNER JOIN SCHEDULE

ON SCHEDULE.ID=PATIENT\_DISEASES.ID\_VISIT

WHERE IDPATIENT=@ID\_USER AND ID\_DISEASE=@ID\_DISEASE AND PATIENT\_DISEASES.END\_DATETIME IS NULL)

RETURN 0;

ELSE

BEGIN

BEGIN TRANSACTION;

INSERT INTO PATIENT\_DISEASES(ID\_VISIT,ID\_DISEASE)

SELECT ID, @ID\_DISEASE

FROM SCHEDULE

WHERE START\_DATETIME=@DATE\_VISIT AND IDPATIENT=@ID\_USER;

COMMIT TRANSACTION;

RETURN 1;

END;

END;

------------------------------------------------------------------------------------------------------------------

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name ='ADD\_ALLERGY')

BEGIN

DROP PROC ADD\_ALLERGY

END

GO

CREATE PROCEDURE ADD\_ALLERGY

@ID\_USER INT,@DATE\_VISIT DATETIME,@ID\_ALLERGY INT

AS

BEGIN

IF EXISTS(SELECT \* FROM USERS\_ALLERGIES INNER JOIN SCHEDULE

ON SCHEDULE.ID=USERS\_ALLERGIES.ID\_VISIT

WHERE IDPATIENT=@ID\_USER AND ALLERGY\_ID=@ID\_ALLERGY AND USERS\_ALLERGIES.END\_DATETIME IS NULL)

RETURN 0;

ELSE

BEGIN

BEGIN TRANSACTION;

INSERT INTO USERS\_ALLERGIES(ID\_VISIT,ALLERGY\_ID)

SELECT ID, @ID\_ALLERGY

FROM SCHEDULE

WHERE START\_DATETIME=@DATE\_VISIT AND IDPATIENT=@ID\_USER;

COMMIT TRANSACTION;

RETURN 1;

END;

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_MEDICAL\_CARD\_BY\_USER\_ID')

DROP PROC GET\_MEDICAL\_CARD\_BY\_USER\_ID

GO

CREATE PROCEDURE GET\_MEDICAL\_CARD\_BY\_USER\_ID @ID\_USER INT, @ELEMENT\_COUNT INT, @PAGE\_NUMBER INT AS

BEGIN

WITH NUMBERED\_MY\_TABLE AS

(

SELECT MEDICAL\_CARD.ID AS "CARD\_ID", MEDICAL\_CARD.DESCRIPTION, MEDICAL\_CARD.CURE, USERS.FIRSTNAME AS DOCTOR\_FIRSTNAME, USERS.LASTNAME AS DOCTOR\_LASTNAME,

SCHEDULE.IDPATIENT, SCHEDULE.START\_DATETIME AS SESSION\_START\_DATETIME,

SCHEDULE.ID AS "ID\_VISIT", SUBDISEASES.SUBDISEASENAME, ROW\_NUMBER() OVER (ORDER BY SCHEDULE.START\_DATETIME DESC) AS ROWNUMBER

FROM MEDICAL\_CARD

INNER JOIN SCHEDULE ON MEDICAL\_CARD.ID\_VISIT = SCHEDULE.ID

INNER JOIN PATIENT\_DISEASES ON SCHEDULE.ID=PATIENT\_DISEASES.ID\_VISIT

LEFT JOIN SUBDISEASES ON PATIENT\_DISEASES.ID\_DISEASE = SUBDISEASES.SUBDISEASEID

INNER JOIN USERS ON IDDOCTOR= USERS.IDUsers

WHERE (SCHEDULE.IDPATIENT = @ID\_USER)

)

SELECT CARD\_ID, DESCRIPTION, CURE, DOCTOR\_FIRSTNAME, DOCTOR\_LASTNAME, IDPATIENT, SESSION\_START\_DATETIME, ID\_VISIT, SUBDISEASENAME

FROM NUMBERED\_MY\_TABLE

WHERE ROWNUMBER BETWEEN (@PAGE\_NUMBER-1) \* @ELEMENT\_COUNT + 1 AND @ELEMENT\_COUNT \* @PAGE\_NUMBER

ORDER BY SESSION\_START\_DATETIME DESC;

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_INFO\_ABOUT\_CLOSE\_DISEASE')

DROP PROC GET\_INFO\_ABOUT\_CLOSE\_DISEASE;

GO

CREATE PROCEDURE GET\_INFO\_ABOUT\_CLOSE\_DISEASE

@PATIENT\_ID INT

AS

BEGIN

SELECT SUBDISEASES.SUBDISEASENAME, USERS1.FIRSTNAME AS DOCTOR\_OPEN\_FIRSTNAME, USERS1.LASTNAME AS DOCTOR\_OPEN\_LASTNAME,

SCHEDULE1.START\_DATETIME,MEDICAL\_CARD1.DESCRIPTION, MEDICAL\_CARD1.CURE,

USERS2.FIRSTNAME AS DOCTOR\_CLOSE\_FIRSTNAME, USERS2.LASTNAME AS DOCTOR\_CLOSE\_LASTNAME,

SCHEDULE2.START\_DATETIME AS END\_TIME

FROM (PATIENT\_DISEASES INNER JOIN SCHEDULE SCHEDULE1 ON PATIENT\_DISEASES.ID\_VISIT = SCHEDULE1.ID

INNER JOIN MEDICAL\_CARD MEDICAL\_CARD1 ON PATIENT\_DISEASES.ID\_VISIT = MEDICAL\_CARD1.ID\_VISIT

INNER JOIN USERS USERS1 on SCHEDULE1.IDDOCTOR=USERS1.IDUsers

INNER JOIN SCHEDULE SCHEDULE2 ON PATIENT\_DISEASES.END\_DATETIME = SCHEDULE2.ID

INNER JOIN USERS USERS2 on SCHEDULE2.IDDOCTOR=USERS2.IDUsers

INNER JOIN SUBDISEASES ON SUBDISEASES.SUBDISEASEID=PATIENT\_DISEASES.ID\_DISEASE

)

WHERE (SCHEDULE1.IDPATIENT = @PATIENT\_ID AND PATIENT\_DISEASES.END\_DATETIME IS NOT NULL)

END;

IF EXISTS (SELECT \* FROM sys.objects WHERE type = 'P' AND name = 'GET\_INFO\_ABOUT\_CLOSED\_ALLERGIES')

DROP PROC GET\_INFO\_ABOUT\_CLOSED\_ALLERGIES;

GO

CREATE PROCEDURE GET\_INFO\_ABOUT\_CLOSED\_ALLERGIES

@PATIENT\_ID INT

AS

BEGIN

SELECT ALLERGY\_NAME, USERS1.FIRSTNAME AS DOCTOR\_OPEN\_FIRSTNAME, USERS1.LASTNAME AS DOCTOR\_OPEN\_LASTNAME,

SCHEDULE1.START\_DATETIME, MEDICAL\_CARD1.DESCRIPTION, MEDICAL\_CARD1.CURE,

USERS2.FIRSTNAME AS DOCTOR\_CLOSE\_FIRSTNAME, USERS2.LASTNAME AS DOCTOR\_CLOSE\_LASTNAME,

SCHEDULE2.START\_DATETIME AS END\_TIME, MEDICAL\_CARD2.DESCRIPTION, MEDICAL\_CARD2.CURE

FROM (USERS\_ALLERGIES INNER JOIN SCHEDULE SCHEDULE1 ON USERS\_ALLERGIES.ID\_VISIT = SCHEDULE1.ID

LEFT JOIN MEDICAL\_CARD MEDICAL\_CARD1 ON USERS\_ALLERGIES.ID\_VISIT = MEDICAL\_CARD1.ID\_VISIT

INNER JOIN USERS USERS1 on SCHEDULE1.IDDOCTOR=USERS1.IDUsers

INNER JOIN SCHEDULE SCHEDULE2 ON USERS\_ALLERGIES.END\_DATETIME = SCHEDULE2.ID

LEFT JOIN MEDICAL\_CARD MEDICAL\_CARD2 ON USERS\_ALLERGIES.ID\_VISIT = MEDICAL\_CARD2.ID\_VISIT

INNER JOIN USERS USERS2 on SCHEDULE2.IDDOCTOR=USERS2.IDUsers

INNER JOIN ALLERGIES ON ALLERGIES.ID\_ALLERGY=USERS\_ALLERGIES.ALLERGY\_ID)

WHERE (SCHEDULE1.IDPATIENT = @PATIENT\_ID AND USERS\_ALLERGIES.END\_DATETIME IS NOT NULL);

END;