МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІСЬКА ПОЛІТЕХНІКА»

Кафедра ІСМ



**Звіт**

До розрахунково-графічної роботи

З дисципліни:

«Веб-технології та веб-дизайн»

Варіант №7

*Виконав:*

*студент групи КН-47*

Іванов Вадим

*Прийняв:*

*доцент* Василюк А. С.

Львів 2018

**ЗМІСТ**

1. Завдання …………………………………………………………………………..3
2. Вступ……………………………………………………………………………....4
3. Розділ 1. Теоретична частина…………………………………………………....5
4. Розділ 2. Практична частина…………………………………………………….10
5. Висновки………………………………………………………………………….14
6. Список використаної літератури………………………………………………..15

**ЗАВДАННЯ**

В ході виконання розрахунково-графічної роботи потрібно в першому розділі викласти теоретичні положення про мову PHP. В другому розділі за допомогою мови гіпертекстової розмітки HTML написати ієрархічний веб-сайт з трьох сторінок на довільну тему. Також в цьому розділі потрібно детально описати процес створення цього сайту, включно з скріншотами. Код сторінки потрібно додати до роботи .

**ВСТУП**

*Актуальність теми*. З огляду на те, що зараз активно розвивається напрямок веб-розробки, з кожним роком все більше нових веб-орієнтованих проектів відкривається. Тому кожному студенту за напрямом “Комп'ютерні науки та інформаційні технології” важливим, популярним та корисним є ознайомлення з сучасними технологіями та практичне застосування певних тохнологій для написання власного веб-орієнтованого додатку.

*Мета і задачі роботи.* Метою роботи є надання коротких, проте повних теоретичних відомостей про мову PHP написання власної веб-орієнтованої аплікації. Для цього потрібно виконати наступні задачі:

* Проаналізувати літературні джерела та віднайти корисну та актуальну інформацію про мову PHP;
* Систематизувати теоретичні знання про мову PHP та викласти основні факти про цю мову та її особливості;
* Вибрати набір технологій для написання аплікації.
* Вибрати тематику для написання аплікації;
* Провести етап написання коду;
* Провести етап тестування аплікації.

**РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА**

**PHP** (Hypertext Preprocessor - PHP: [гіпертекстовий](https://uk.wikipedia.org/wiki/%D0%93%D1%96%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) [препроцесор](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D0%BE%D1%80)), попередня назва: Personal Home Page Tools-[скриптова](https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), була створена для генерації [HTML](https://uk.wikipedia.org/wiki/HTML)-сторінок на стороні [веб-сервера](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із [Java](https://uk.wikipedia.org/wiki/Java), [.NET](https://uk.wikipedia.org/wiki/.NET), [Perl](https://uk.wikipedia.org/wiki/Perl), [Python](https://uk.wikipedia.org/wiki/Python), [Ruby](https://uk.wikipedia.org/wiki/Ruby)). PHP підтримується переважною більшістю [хостинг](https://uk.wikipedia.org/wiki/%D0%A5%D0%BE%D1%81%D1%82%D0%B8%D0%BD%D0%B3)-[провайдерів](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B2%D0%B0%D0%B9%D0%B4%D0%B5%D1%80). PHP - проект [відкритого програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F).

* 1. **Історія розвитку мови PHP**

Історія PHP починається з [1995](https://uk.wikipedia.org/wiki/1995) року, коли Расмус Лердорф ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Rasmus Lerdorf) створив простий [застосунок](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) мовою Perl, що аналізував відвідування користувачами його резюме на веб-сайті. Потім, коли цим [застосунком](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) вже користувалися кілька чоловік, а число охочих одержати його постійно збільшувалося, Лердорф назвав своє творіння Інструменти для особистої домашньої сторінки [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Personal Home Page Tools версія 1 і виставив для вільного завантаження. З цієї миті почався небувалий зліт популярності PHP.

PHP отримав змішані відгуки через відсутність власної підтримки Unicode на рівні основної мови. У 2005 році був започаткований проект, очолюваний Андрієм Змієвським, для залучення рідної підтримки Unicode на PHP, шляхом вбудовування бібліотеки "Міжнародні компоненти для Unicode" (ICU) та вбудованих текстових рядків як UTF-16. Оскільки це призведе до серйозних змін як до внутрішньої частини мови, так і до коду користувача, планувалося випустити його як версію 6.0 мови разом з іншими основними функціями, які розвиваються.

У PHP 5 також включені два нові [модулі](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C) для роботи з [протоколами](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%83%D0%BD%D1%96%D0%BA%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) - [SimpleXML](https://uk.wikipedia.org/wiki/SimpleXML) і [SOAP](https://uk.wikipedia.org/wiki/SOAP). SimpleXML дозволяє значно спростити роботу з XML-даними, представляючи вміст XML-документа у вигляді PHP-об'єкта. Розширення SOAP дозволяє будувати на PHP [сценарії](https://uk.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D1%96%D0%B9), що обмінюються інформацією з іншими [застосунками](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) за допомогою [XML](https://uk.wikipedia.org/wiki/XML)-повідомлень поверх існуючих веб-протоколів, наприклад [HTTP](https://uk.wikipedia.org/wiki/HTTP). Модуль для роботи з [SOAP](https://uk.wikipedia.org/wiki/SOAP) для PHP 5 надає розробникам засіб для достатньо швидкого створення ефективних SOAP-клієнтів і SOAP-серверів.

Новий модуль PHP 5 [MySQLi](https://uk.wikipedia.org/wiki/MySQLi) (MySQL Improved) призначений для роботи з [MySQL](https://uk.wikipedia.org/wiki/MySQL)-сервером версій 4.1.2 і вище, реалізовуючи не тільки процедурний, але і об'єктно-орієнтований інтерфейс до MySQL. Додаткові можливості цього модуля включають - [SSL](https://uk.wikipedia.org/wiki/SSL), [контроль транзакцій](https://uk.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D1%96%D1%8F_(%D0%B1%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)), [підтримка реплікації](https://uk.wikipedia.org/w/index.php?title=%D0%9F%D1%96%D0%B4%D1%82%D1%80%D0%B8%D0%BC%D0%BA%D0%B0_%D1%80%D0%B5%D0%BF%D0%BB%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%97&action=edit&redlink=1) та інші.

Шоста версія PHP розроблялася з жовтня 2006 року. Було зроблено безліч нововведень, як, наприклад, виключення з ядра регулярних виразів POSIX і «довгих» суперглобальних масивів, видалення директив safe\_mode, magic\_quotes\_gpc і register\_globals з конфігураційного файлу php.ini. Одним з основних нововведень повинна була стати підтримка Юнікоду. Однак у березні 2010 року розробка PHP6 була визнана безперспективною через складнощі з підтримкою Юнікоду. Вихідний код PHP6 переміщений на гілку, а основною лінією розробки стала версія 5.4.

У 2014 році було проведено голосування, за результатами якого наступна версія отримала назву PHP 7. Вихід нової версії планувався в середині жовтня 2015 року. У березні 2015 року Zend представили інфографіку в якій описані основні нововведення PHP 7. 3 грудня 2015 року було оголошено про вихід PHP версії 7.0.0. Нова версія грунтується на експериментальній гілці PHP, яка спочатку називалася phpng (PHP Next Generation - наступне покоління), і розроблялася з упором на збільшення продуктивності і зменшення споживання пам'яті. У новій версії додана можливість вказувати тип повертаються з функції даних, доданий контроль переданих типів для скалярних даних, а також нові оператори.

* 1. **Синтаксис мови**

Всі сценарії оформляються у вигляді блоків коду. Ці блоки можуть бути поміщені в HTML-код, але відділені від нього відповідними обмежувачами. Код PHP в HTML повинен знаходитись між початковим тегом <?php та кінцевим ?> (або між <script language="php"> та </script>) Бажаним варіантом виділення PHP коду є варіант <?php ?>, оскільки саме такі початковий та кінцевий теги дозволять використовувати PHP код у документах, які відповідають правилам [XML](https://uk.wikipedia.org/wiki/XML). Також можна використовувати скорочений запис <? ?> (інколи потрібно активізувати даний стиль у файлі налаштувань інтерпретатора php.ini: змінна short\_open\_tag повинна мати значення On) і записом у стилі [ASP](https://uk.wikipedia.org/wiki/Active_Server_Pages): <% %> (в php.ini змінна asp\_tags повинна мати значення On).

Основою будь-якого програмування є [змінні](https://uk.wikipedia.org/wiki/%D0%97%D0%BC%D1%96%D0%BD%D0%BD%D0%B0). PHP, як і деякі інші [UNIX](https://uk.wikipedia.org/wiki/UNIX)-[скриптові](https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) мови, не потребує їх опису. Існує три стилі оформлення змінних у PHP:

* короткий стиль - змінні записуються у вигляді: $variable. Використовується в більшості випадків. Змінна, яка створюється програмним кодом, записується таким чином. Можливо також використання такого стилю для добування змінної із ФОРМИ, якщо ввімкнено register\_globals у файлі конфігурації php.ini.
* середній стиль - $\_POST['variable'], $\_GET['variable'], $\_REQUEST['variable']. Використовується для отримання даних з полів ФОРМИ. В залежності від типу передачі даних, встановлюється POST або GET або REQUEST для обох випадків. Дані змінні називаються також глобальними.
* Довгий стиль - $HTTP\_POST\_VAR['variable'], $HTTP\_GET\_VARS['variable']. Найповніший стиль запису і найменш використовуваний. Починаючи з PHP 5, ці змінні можна вимкнути. Використовувати змінні такого типу не рекомендовано задля кращої сумісності із майбутніми версіями PHP.

До базових типів належать булеві дані, цілі та дійсні числа із плаваючою комою, а також рядки. Булеві дані виражають істинність значення. Цілі числа можуть бути подані у вісімковому, десятковому та шістнадцятковому вигляді. Розмір цілого числа може змінюватись залежно від платформи, зазвичай, розрядність становить 32 біти. PHP не підтримує беззнакові цілі числа. Дійсні числа із плаваючою комою можуть бути подані в десятковій або експоненційній формі. Для кожної змінної можна надати власний тип даних.

[Оператори](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) з погляду мови програмування дозволяють виконувати відповідну дію над одним чи кількома операндами. Оператори бувають трьох типів - [унарні](https://uk.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B0%D1%80%D0%BD%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D1%8F), [бінарні](https://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%BD%D0%B0%D1%80%D0%BD%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D1%8F) та [тернарні](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%BD%D0%B0%D1%80%D0%BD%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D1%8F). Оператори, як і в інших мовах, характеризуються не лише дією, а й [асоціативністю](https://uk.wikipedia.org/wiki/%D0%90%D1%81%D0%BE%D1%86%D1%96%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%96%D1%81%D1%82%D1%8C) та [пріоритетністю](https://uk.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D1%96%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BD%D1%96%D1%81%D1%82%D1%8C&action=edit&redlink=1). Особливістю [булевих операцій](https://uk.wikipedia.org/wiki/%D0%91%D1%83%D0%BB%D0%B5%D0%B2%D0%B0_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0) порівняння - розрізнення двох класів - з урахуванням типу і без нього, при якому відбувається приведення до відповідного типу. Округлення відбувається завжди в менший бік. У мові реалізовані особливі класи операторів - виконання, управління помилками та перевірки належності до класу.

[Функції](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) з погляду мови програмування є контейнерами коду, причому можливе включення інших функцій та класів. На цьому і базується можливість умовного визначення функції. В цьому випадку висувається вимога попередньої декларації викликаної функції, що не обов'язкове в інших випадках. Можливості перевизначення чи деактивації функції не існує. Результат, який повертає функція може мати будь-який [тип](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85).

У мові реалізована функціональність посилань. Можливо створити необмежену кількість псевдонімів, що посилаються на єдиний сегмент даних. При вивільненні будь-якого з псевдонімів, сегмент даних залишається в пам'яті до моменту завершення сценарію або вивільнення усіх посилань. Натомість прийнятого в багатьох мовах принципу [перевантаження функцій](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%B0%D0%BD%D1%82%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97), що дозволяє змінити хід виконання певної функції залежно від типу та кількості вхідних параметрів, у PHP використовується метод динамічних аргументів. Це дає змогу не визначати кількість параметрів функцій при їх оголошенні, а працювати з тими аргументами, які були отримані на момент виклику функції. У тілі функції можливо отримати кількість переданих їй аргументів і проводити відповідні маніпуляції. При оголошенні функції звичайним чином можливе задання значень аргументів за замовчуванням. Функції можуть повертати лише одне значення, проте це обмеження можна оминути, використавши не лише масиви, а й посилання. Передача аргументів за посиланням неможлива під час виконання та оголошення функції.

Після виконання сценаріїв, простір пам'яті, займаної ними, очищується [збирачем сміття](https://uk.wikipedia.org/wiki/%D0%97%D0%B1%D0%B8%D1%80%D0%B0%D1%87_%D1%81%D0%BC%D1%96%D1%82%D1%82%D1%8F). Проте за потреби можливе власноруч кероване виконання очищення пам'яті від надлишкових сегментів даних під час виконання скриптів, але використання функцій очищення пам'яті не є виправданим, хоча така можливість існує.

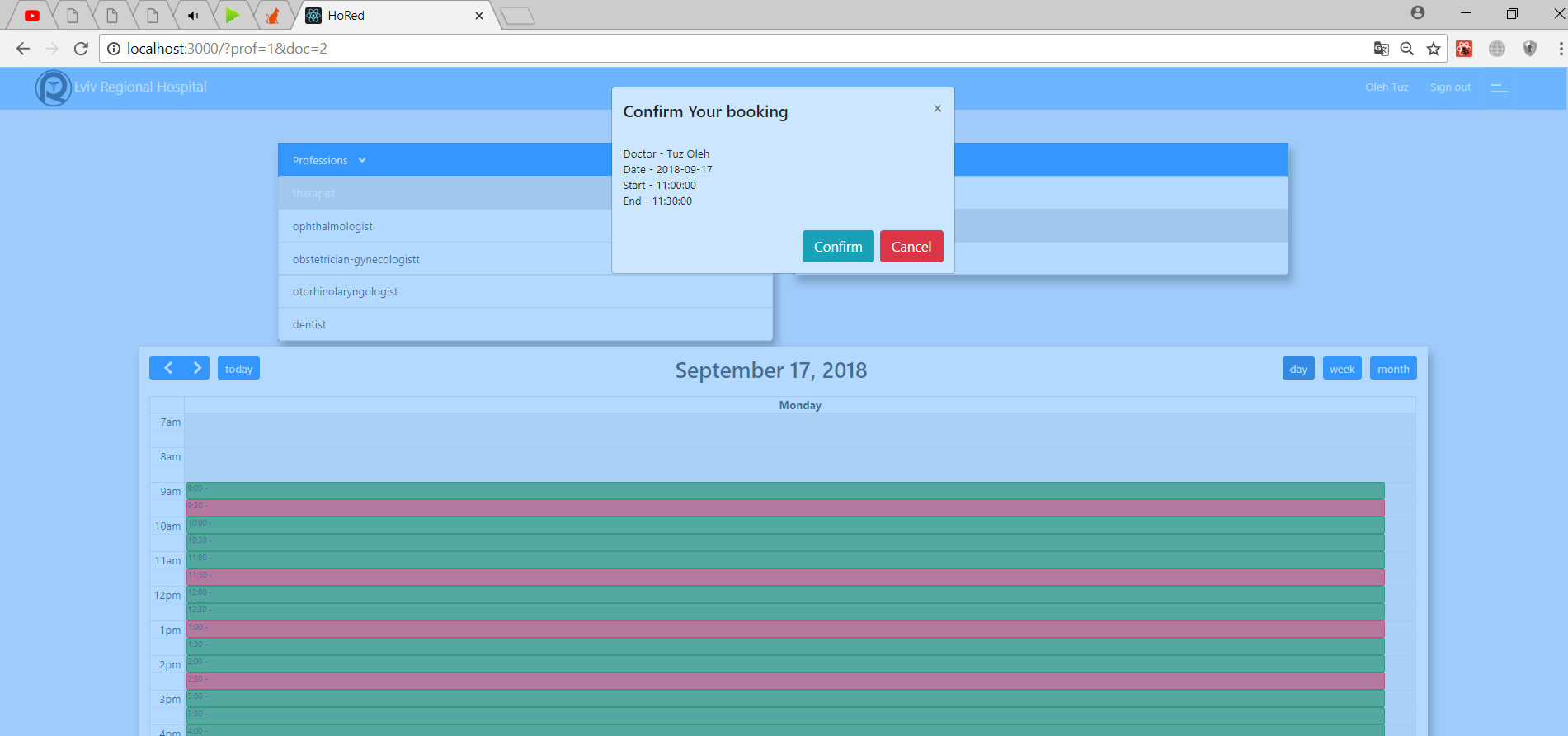
Для побудови програмних комплексів можна використовувати модульний підхід, виконуючи розділення різнорідного коду. При потребі, можливе виконання під'єднання необхідних модулів, причому операція виконання може бути і умовною. Під'єднані до скрипту файли можуть повертати значення.

* 1. **Недоліки мови PHP**
* Незручність дизайну мови;
* Змінні з символом $;
* Складні назви поширених функцій (html\_entity\_decode,  тощо);
* Не підтримується [Unicode](https://uk.wikipedia.org/wiki/Unicode) у версіях до 6.0;
* Непередбачуваність нових версій PHP.

**РОЗДІЛ 2. ПРАКТИЧНА ЧАСТИНА**

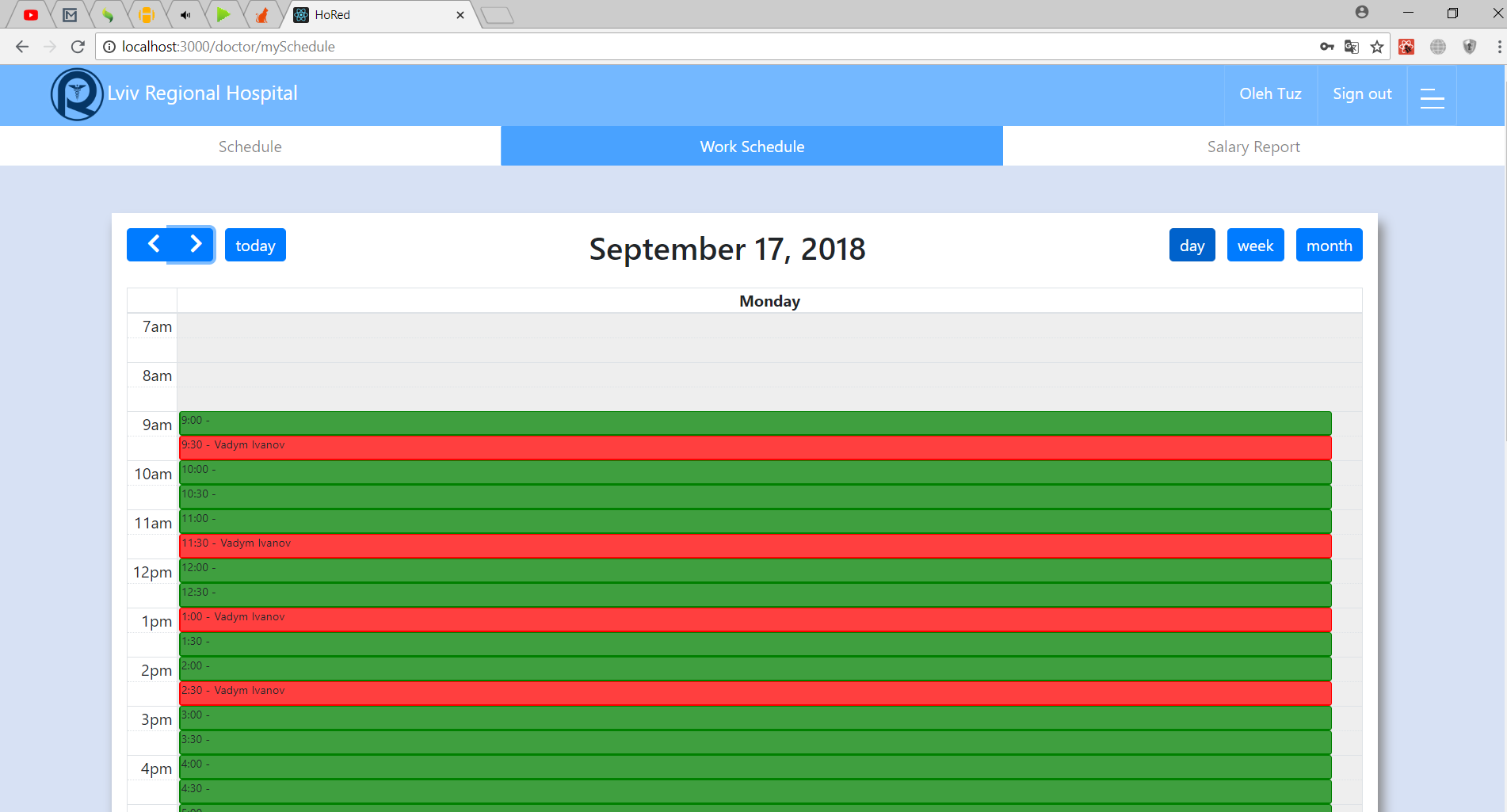
Впродовж виконання розрахункової роботи було розроблено WEB-орієнтовану аплікацію “HoReD”, назва якої означає – медична реєстратура (Hospital Registration Desk). Для написання аплікації було обрано наступний стек технологій – ASP .Net WebAPI для написання серверної частини, React JS для написання клієнтської 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).



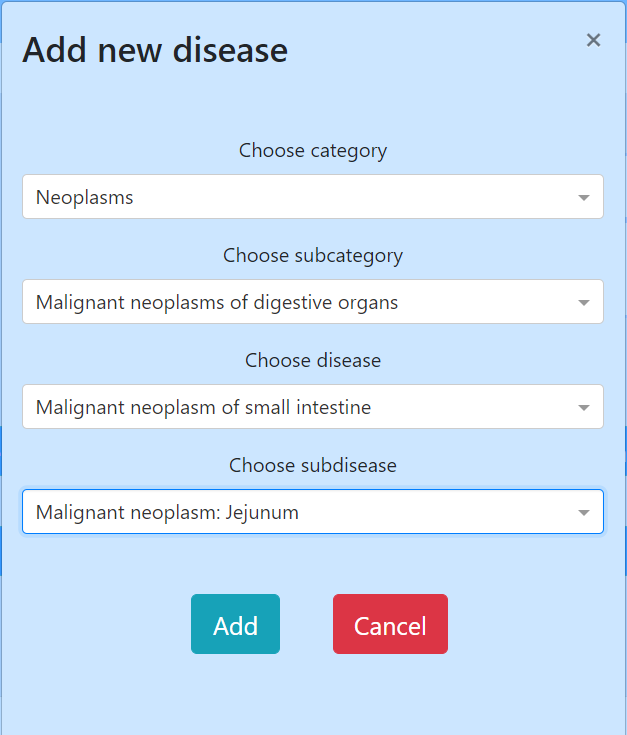
*Рис. 1 Початкова сторінка сайту.*

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).

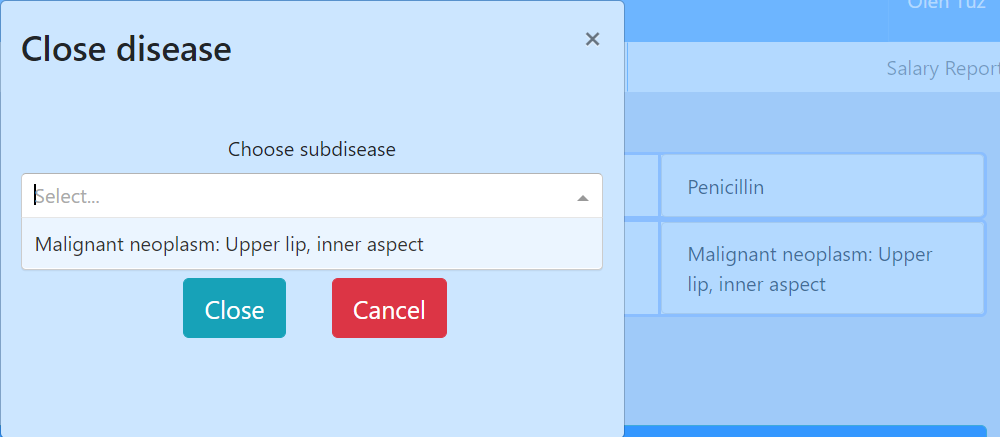


*Рис. 2 Робочий календар лікаря.*

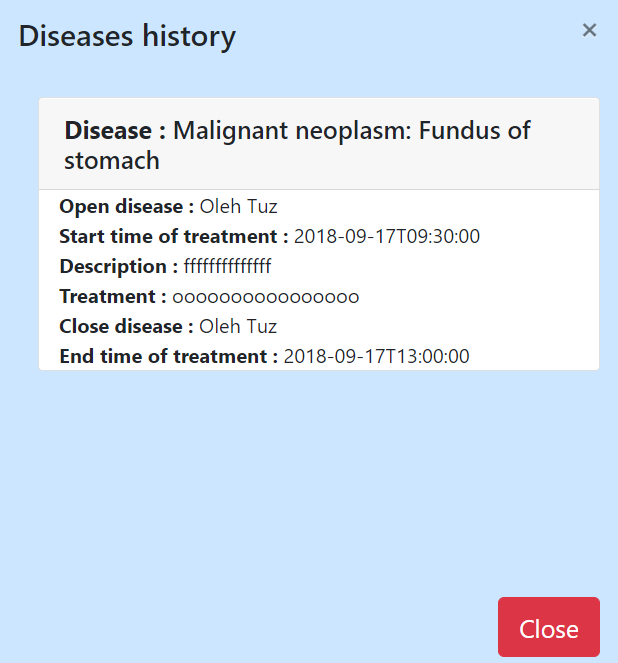
Після вибору відповідного сеансу лікар переходить на сторінку з інформацією про пацієнта, на якій лікар може додати(Рис.3), закрити діагноз(Рис.4) або алергію, додати опис сеансу та лікування, переглянути історію закритих алергій та діагнозів(Рис.5).



*Рис. 3 Додавання нового діагнозу.*

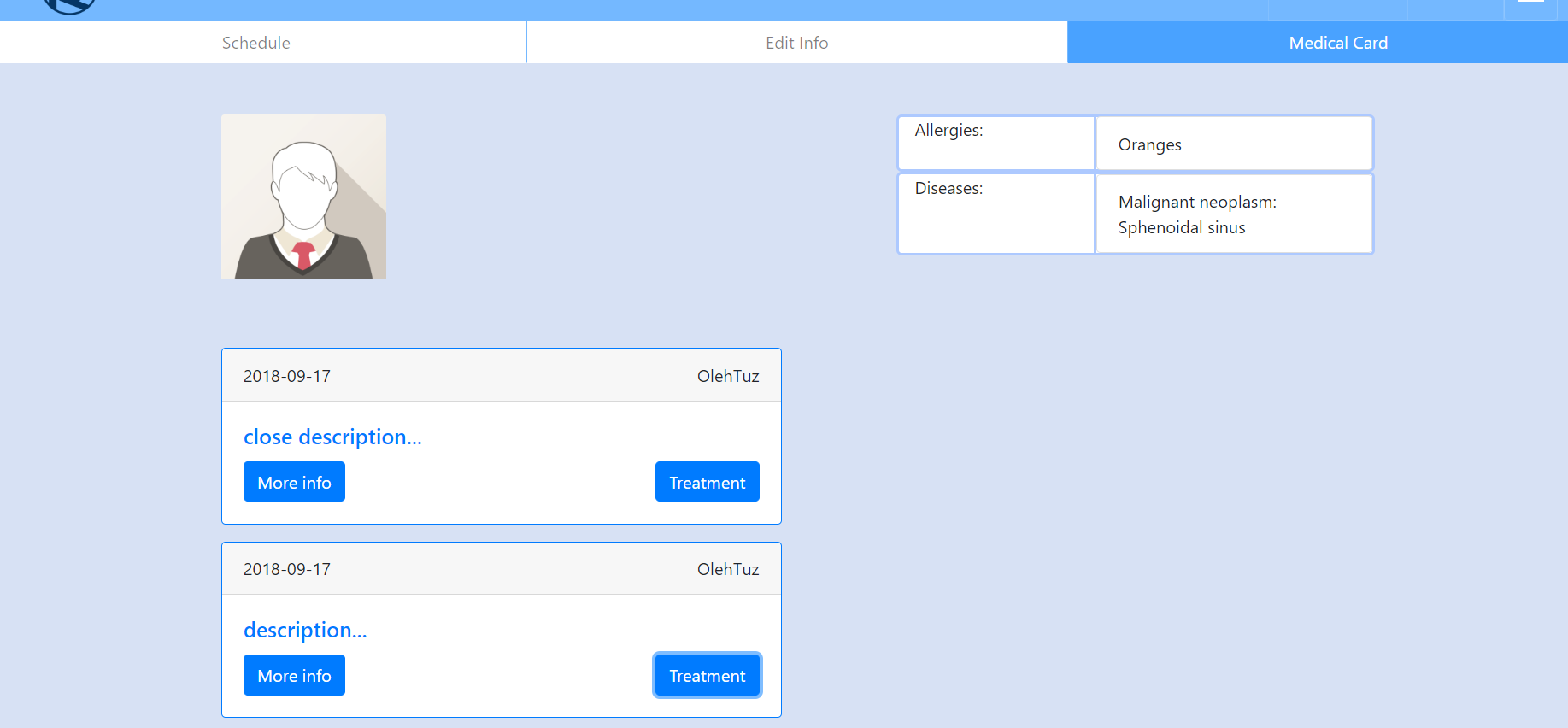


*Рис. 4 Закривання активного діагнозу.*



*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**

Виконання даної розрахунково-графічної роботи дало змогу ознайомитися з мовою PHP та розробити власний сайт з переходами по різних сторінках.

Впродовж практики було здійснено ознайомлення з різними видами прикладного програмного забезпечення, вдосконалено свої навички.

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

Під час виконання роботи підібрано необхідну інформацію для придбання навичок практичної діяльності у розробці веб-сайтів.

**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.
7. Сторінка в ресурсі Вікіпедія. [Електронний ресурс]. Режим доступу URL: https://uk.wikipedia.org/wiki/PHP

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>

/// 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>

Додаток Б

Код клієнтської частини сайту

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

};

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>)

}

}

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 = '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 ='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