Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентки *Михайленко Марії Олександрівни*
(ПІБ)

академічної групи 122-19-4
(шифр)

напряму підготовки *122 Комп'ютерні науки*
(код і назва напряму підготовки)

на тему: *"Розробка веб-орієнтованого застосунку введення дентальної хроніки для запису і аналізу стану ротової порожнини"*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпропетровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О.Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студентці *122-19-4* *Михайленко Марії Олександрівни*
(група) (прізвища та ініціали)

тема кваліфікаційної роботи *«Розробка веб-орієнтованого застосунку введення дентальної хроніки для запису і аналізу стану ротової порожнини»*

затверджена наказом ректора НТУ «ДП» від *16.05.2023 р* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Спірінцев В. В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Михайленко М.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: : *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *11 .06 .2023 р*

РЕФЕРАТ

Пояснювальна записка: 141 с., 42 рис., 6 табл., 5 дод., 39 джерел.

Об’єкт розробки: веб-орієнтований застосунок ведення дентальної хроніки для запису і аналізу стану ротової порожнини.

Мета кваліфікаційної роботи: розробка веб-орієнтованого застосунку для ведення дентальної хроніки, аналізу стану ротової порожнини, що реалізується за допомогою алгоритмів упорядкування, сортування, і приведення до зрозумілою форми сприйняття введених даних.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовано наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-орієнтованого застосунку, описана робота застосунку, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне завдання полягає у створені веб-орієнтованого застосунку, що забезпечує спрощену методику контролю за здоров’ям ротової порожнини, задля підвищення мотивації, організації догляду та відслідковування лікувальних процесів.

Розробка застосунку для ведення дентальної хроніки є актуальною, що не підлягає сумніву, який спростовується фактом відсутності розробок з подібним концептом та постійною потребою у догляді та контролю за показниками власного здоров’я.

Список ключових слів: ВЕБ-ОРІЄНТОВАНИЙ ЗАСТОСУНОК, БАЗА ДАНИХ, ВЕБ-ДИЗАЙН, ІНТЕРФЕЙС КОРИСТУВАЧА, NODE.JS, REACT, ПРЕДПРОЦЕСОРИ, MYSQL, ДОГЛЯД ЗА ЗДОРОВ’ЯМ, КОТРОЛЬ ЗДОРОВ’Я.

ABSTRACT

Explanatory note: 141 pp., 42 fig., 6 tabl., 5 extra., 39 sources.

Object of development: a web-based application for keeping dental records for recording and analyzing the state of the oral cavity.

The purpose of the qualification work: the development of a web-oriented application for keeping dental records, analyzing the state of the oral cavity, which is implemented with the help of algorithms for ordering, sorting, and bringing the entered data to a comprehensible form of perception.

In the introduction, the analysis and current state of the problem is considered, the purpose of the qualification work and the field of its application are specified, the justification of the relevance of the topic is given, and the statement of the task is clarified.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of the development is determined, the task statement is formulated, and the requirements for software implementation, technologies and software tools are specified.

In the second section, the available solutions are analyzed, a platform for development is chosen, the design and development of a web-oriented application is performed, the operation of the aplication is described, the algorithm and structure of its functioning, as well as the application call and download, the input and output data are determined, the composition of the parameters of the technical means is characterized.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating the application is calculated, and the time for its creation is calculated.

The practical task is to create a web-based application that provides a simplified method of monitoring oral health, in order to increase motivation and organization of care and monitoring of treatment processes.

The development of the application for keeping dental records is relevant, that cannot be doubted, owing to the fact of the absence of developments with a similar concept and the constant need for care and control of one's own health indicators.

Keyword List: WEB ORIENTED APPLICATION, DATABASE, WEB DESIGN, USER INTERFACE, NODE.JS, REACT, PREPROCESSORS, MYSQL, HEALTH CARE, HEALTH MONITOR.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

UML – Unified Modeling Language;

MIC – медична інформаційна система;

HER – Electronic Health Record;

ЦП – центральний процесор;

ОП – оперативна пам’ять;

SSD – Solid-State Drive;

ІС – інформаційна система;

ОС – операційна система;

СУКБД – Система управління Картографічними Базами Даних;

JS – JavaScript;

SPA – Single Page Application;

HTML – Hypertext Markup Language;

DOM – Document Object Model;

JSX – JavaScript XML;

SEO – Search engine optimization;

JWT – JSON Web Token;

CORS – Cross-Origin Resource Sharing;

MVC – Model-View-Controller;

ODM – Object Document Mapping;

ORM – Object Relational Mapping;

CRUD – Create Read Update Delete;

БЕМ – Блок Елемент Модифікатор;

SCSS – Sassy Cascading Style Sheets;

CSS – Cascading Style Sheets;

HTTP – HyperText Transfer Protocol.

ВСТУП

На даний момент інформаційні технології відіграють важливу роль у різних питаннях, галузях і сферах. Виключенням не є і медицина. Розробки в даній галузі охоплюють дуже широкий спектр, що включає в себе технології для покращення і спрощення різноманітних процесів, таких як: лікування, ведення історій хвороб, діагностування, організації графіків прийомів, проведення опитувань для наукового аналізу, комунікації лікарів з пацієнтами, контролю приймання ліків, роботи цілих клінік та звичайне спостереження за показниками власного тіла і повсякденних звичок. Крім цього, медицина складається з великої кількості окремих областей, кожна з яких відповідальна за ряд задач, вирішення яких можна автоматизувати та оптимізувати. Тому це саме та галузь, яка постійно розвивається і потребує нових розробок і технологій.

В даній кваліфікаційній роботі мова йде про конкретну галузь медицини, а саме про стоматологію, і про запропонований в роботі веб-орієнтований застосунок, що спрямований на покращення стану ротової порожнини. Головна ідея полягає у тому, щоб надати можливість пацієнту самому вести свою дентальну хроніку, задля підвищення рівня відповідальності, зацікавленості і обізнаності про стан власного здоров’я. Це може стати непоганою мотивацією, а також надасть можливість краще розуміти лікувальні процеси, їх причини і наслідки, та при цьому приймати рішення від яких залежатиме результат. Великою превагою такого підходу є те, що запропонована розробка зможе допомогти покращити і пришвидшити лікування і зі сторони самого лікаря також. Адже постійний доступ до даних всієї клінічної історії, яка не залежить ні від зміни спеціаліста, чи від його рівня відповідальності до якості опису етапів лікування, може допомогти виявити захворювання і їх причини набагато швидше, що може помітно вплинути на результат лікування.

Об’єктом дослідження є веб-орієнтований застосунок ведення дентальної хроніки для запису і аналізу стану ротової порожнини.

Мета кваліфікаційної роботи перш за все є розробка програмного забезпечення веб-орієнтованого застосунку для ведення дентальної хроніки задля забезпечення можливості продуктивного моніторингу за станом ротової порожнини для кожного, кого турбує своє здоров’я.

Для досягнення поставленої мети необхідно вирішити перелік задач, що включає:

– розглянути подібні розробки в контексті медичної галузі задля аналізу недолік та переваг існуючих рішень;

– сформувати основні вимоги до розробки застосунку, які включають вимоги до: функціональних характеристик, інформаційної безпеки, складу та параметрів технічних засобів, інформаційної та програмної сумісності;

– визначити функціональне призначення розроблюваного інформаційної системи;

– обрати стек технологій та мови програмування, які будуть використані при створені застосунку;

– спроектувати структуру, алгоритми та організацію вхідних та вихідних даних системи застосунку задля загального розуміння її роботи та етапів розробки;

– розробити веб-орієнтований застосунок;

– розрахувати трудомісткість, вартість та витрати на створення інформаційної системи.

Практичне значення полягає у створенні веб-орієнтованого застосунку, що забезпечує: можливість продуктивного моніторингу за станом ротової порожнини; спрощений і пришвидшений процес введення записів дентальної хроніки; покращення сприйняття клінічної картини завдяки візуалізації зроблених процедур; можливість і засоби для організації лікувального процесу для фахівців стоматологічної галузі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКИ ЗАВДАННЯ

1.1. Загальні відомості з предметної області

1.1.1 Основні поняття

Предметна область даної кваліфікаційною роботи передусім – інформаційні технології, яка використовує програмні та комп'ютерні засоби для реалізації процесів реєстрації, відбору, захисту, подання, передавання, збереження та опрацювання інформації – інформаційного ресурсу у формі знань та даних – з метою створення інформаційних ресурсів.

В основі цього визначення лежить базове поняття фундаментальної науки – інформатики, яка була подана в 70-х рр. XX ст. Інформатику часто ототожнюють з поняттям комп'ютерних наук. Але варто зауважити, що їхня принципова відмінність полягає в тому, що перша відноситься до категорії фундаментальних наук, у той же час як друга – загалом за своєю сутністю та всіма наявними ознаками належить до категорії прикладних наук, які базуються на фундаментальних законах та закономірностях інформаційних процесів, котрі в свою чергу вивчаються в рамках фундаментальної науки інформатики.

Для комп'ютерних наук характерною ознакою виділення серед спектра прикладних наук є об'єкт прикладення знань, умінь та навичок у контексті конкретного об'єкту – обчислювача (комп'ютера).

Варто зазначити, що одним із важливих понять в даній області є інженерія, наука про проектування та побудову об'єктів певної природи. В даному контексті доречно розглядати наступні види інженерії:

– комп'ютерна інженерія – охоплює проблематику проектування та створення об'єктів комп'ютерної техніки;

– програмна інженерія – піклується проблематикою проектування та створення об’єктів, що іменуються програмними продуктами;

– інженерія даних та знань – опікується проектуванням та створенням інформаційних продуктів;

– системна інженерія – опікується проектуванням та створенням міжкомпонентних (інтерфейсних) взаємозв’язків та формуванням цілісних системних об’єктів.

Інформаційна технологія уособлює собою точку зв’язку всіх зазначених інженерій.

Інформаційна система (ІС) визначається як певний набір інформаційних технологій, що в комплексі зорієнтовані на досягнення певної системної мети, виконуючі задані функції та пропонуючи при цьому споживачам якісні інформаційні продукти та сервіси.

В рамках даної кваліфікаційної роботи, розробляється веб-орієнтована інформаційна система. Одні із ключових понять якої включають:

– веб-браузер – це програмне забезпечення, яке дозволяє користувачеві знаходити, отримувати доступ і відображати веб-сторінки;

– веб-сервер – це програмне та апаратне забезпечення, яке використовує HTTP (протокол передачі гіпертексту) та інші протоколи для відповіді на запити клієнтів , зроблені через Всесвітню павутину.

– клієнт-сервер – взаємозв’язок між програмами, що співпрацюють у застосунку, що складається з клієнтів, які ініціюють запити на послуги, і серверів, які надають цю функцію чи послугу;

– база даних – сукупність взаємопов’язаних (зазвичай складно структурованих) даних, яку можна спільно використовувати та керування якою здійснюється централізовано.

1.1.2. Аналіз рішень

Так як область застосування – це перш за все медицина, і головною функцією застосунку є ведення дентальної хроніки пацієнта, яка походить від записів клінічної історії лікування, спочатку було розглянуто розробки систем медичних карток.

Подібних систем було вже розроблено велику кількість. Всі вони відрізняються архітектурою та реалізацією, але при цьому всі вони створені задля основної цілі – замінити ведення паперових документів на електронні, що в свою чергу має на меті пришвидшити процеси заповнення медичних карток та визначення діагнозів, шляхом об’єднання всієї необхідної інформації в одній базі даних. Також такого типа системи спрямовані оптимізувати не просто процеси лікування окремого лікаря і його пацієнта, вони можуть охоплювати клініки, аптеки, амбулаторії і лікарні відразу цілої держави.

Для початку було розглянуто медичну систему, яка запроваджена в Україні на державному рівні – eHealth [1]. Це двокомпонентна система, що складається з центральної бази даних, взаємодію з якою забезпечує електронна медична інформаційна система, яка має багато різновидностей в залежності від цілей і специфіки області, наприклад: Medics, Medcenter+, Helsi та інші. Загальне призначення МІС – це забезпечення процесів заповнення медичних карток і планування лікувального процесу і прийомів. Одним із прикладів можна навести систему, що називається Medics.

Щодо інших EHR- систем (Electronic Health Record – з англ. Електронні Системи Медичної Документації) то найбільш популярними розробниками яких є: Cerner Corporation, eClinicalWorks, Athenahealth, Vegadigm та інші.

Але існують й більш спрощені системи медичних карток. Одну із функцій, які вони також виконують – це забезпечення комунікацію з лікарем. Представниками таких застосунків є: Healow [2] і MyChart [3]. Перший з яких розроблений вже раніше згадуваною eClinicalWorks LLC. Цей застосунок

дозволяє отримувати доступ до актуальних медичних записів, керувати лікарями та власними особистими даними та записуватися на прийом. Також він забезпечує можливість керувати відразу декількома обліковими записами, що дає змогу отримувати доступ до медичної інформації цієї родини.

Інший застосунок, MyChart, розроблений корпорацією Epic Systems, має ту ж саму функціональність, що і попередній застосунок. Але він відрізняється тим, що також забезпечує: розрахунок приблизної вартості послуги, перегляд та оплата медичних рахунків та надання спільного доступу будь-кому і у будь-який час.

Такого типу застосунки мають один суцільний мінус, для того щоб їх використовувати потрібно заздалегідь узгоджувати вибір застосунку з лікарем, а якщо взяти до уваги, що лікування не обмежується одним лікарем чи однією лікарнею, то це може приносити деякі незручності.

Наступна ланка, що так чи інакше перетинається з ідеєю розробки даного проекту, була розглянута за допомогою застосунків, призначених для ведення щоденника контролю здоров'я. Подібні додатки мають на меті виконувати наступні функції: слідкування за станом здоров'я, використовуючи низку трекерів різних симптомів та характеристик (такі як пульс, тривалість сну, кількість пройдених кроків, оцінка настрою тощо); а також нагадування про прийняття прописаних ліків, попередньо внесених у систему. Для прикладу наводяться такі мобільні додатки, як: Symple Symptom Tracker (розробник – Symple Health, Inc), Health Diary (розробник – PNN Soft), Physical Health Diary (розробник – App Workshop), Health Log (розробник – AR Production Inc.) тощо.

Але якщо розглядати більш локалізовану галузь медицини, саме на яку націлена розробка даної кваліфікаційної роботи, то проаналізувавши наявні розробки, було дійдено висновку, що нічого подібного, в комплексному сенсі, ще не було розроблено. Але при цьому, якщо розглядати кожну функціональну ціль окремо, то все ж таки було знайдено декілька цікавих рішень.

Всі розглянуті застосунки, що направлені на стоматологічну область, було розділено на декілька типів, а саме: довідкові, навчальні, ті що нагадують про чистку зубів, ті що привчають чистити зуби, опитувальники, щоденники дієти і спрямовані на введення стоматологічного журналу та планування робочого процесу.

Довідкові несуть суто інформативну функцію, надаючи інформацію про правила гігієни ротової порожнини, стоматологічних термінів, дієт, спрямованих для здоров'я зубів і також в деяких з них можуть розміщуватися новини і статті зі стоматологічного напрямку, як для спеціалістів так і для людей, які хочуть дізнатися факти і правила догляду за власним здоров'ям. Одним із прикладів є програма My Dental Care [4], яка була створена представником галузі, Шивамом Дівані, і спрямована на навчання пацієнтів різноманітним проблемам із зубами.

Для студентів-стоматологів виділяється окремий вид довідкових застосунків – навчальні. Серед них було розглянуто: тести, що підготують до екзаменів; опис деталей різних процедур; збірник інформації про морфологію зубів; галерея 3D моделей кожної частини ротової порожнини та інші.

Представником категорії додатків, що призначені для нагадування про чистку зубів може назвати себе застосунок Brush DJ [5], розробник якого – Ben Underwood. Він також нагадує про час зміни зубної щітки, про використання зубної нитi, ополіскувала тощо. Його особливість полягає у тому, що він має вбудований таймер для зубної щітки, що дозволяє покращити процес чистки зубів. Зазвичай подібні розробки орієнтовані на мотивацію, і застосовують для цього різні методи заохочення.

Застосунки привчання до чистки зубів зазвичай націлені на дітей. Вони наповнені різними тренувальними демонстраціями і поясненням правильної технології очищення в ігровій формі. Найкращими прикладами є: застосунок Time2Brush, розробник якого – BunnerMobile, і Brusheez – The Little Monsters Toothbrush Timer [6]. Через те що в них використовують різні види заохочень, такі як відкриття нових персонажів, музикальний супровід, різні

ігрові завдання, у дітей з’являються бажання і звичка слідкувати за власною ротовою порожниною, що внаслідок покращує здоров’я їх зубів. Більш детально це було описано в дослідженні [7].

Також один із видів додатків, направлених на стоматологічну галузь є опитувальники. Зазвичай вони використовуються для різних досліджень місцевого, шкільного, державних та інших видів за масштабованістю покриття території рівнів. До таких програм належать: ICADPLUS, PCDS, OHSMA і NutriOdonto [8-11].

Щоденники дієти напряму стосуються галузі стоматології. Тому що різний тип їжі по різному впливають на здоров’я ротової порожнини. Тому це було питанням часу, коли з’являться додаток, що надасть змогу контролювати власний раціон і його вплив на організм, зокрема на зуби. Одним із таких застосунків є FoodForTeeth[12], розробник якого – Prateek Biyani. Його основна мета і функція – визначення харчових джерел проблем із зубами, і подальша зміна для покращення загального стану в майбутньому.

І остання із перерахованих категорій – це застосунки, що призначенні для введення стоматологічного журналу. Вони спрямовані суто для стоматологів і спеціалістів стоматології. Один із прикладів можна навести мобільний застосунок dentalRecord[13], розробником якого є – AppDevCoders. За допомогою якого у користувача є можливість керувати пацієнтами та їх історією хвороби, керувати зустрічами, платежами і різними типами завдань, вимірювати ефективність роботи, обслуговувати декілька спеціалістів і навіть клінік одночасно. Але є і більш простіші розробки, спрямовані суто на планування і організації графіку стоматолога. Вони також надають можливість пацієнтам обирати лікаря і записуватися до нього на прийом. До таких застосунків можна віднести LocalMed [14], розроблений Dental Intelligence.

Але одна з найбільш цікавих розробок – це веб-сервіс для виявлення зубних захворювань за допомогою різних видів обробок панорамних

зображень у реальному часі, що має на меті виявляти ранні симптоми діагнозів і запобігати майбутнім ускладненням.

Звісно список діагнозів, що можна виявити таким чином доволі обмежений, але це зовсім не зменшує важливість внеску даної розробки в стоматологічну галузь. Адже одним із захворювань, яке може бути діагностовано за допомогою цього сервісу – є карієс, що в свою чергу являє собою одним із найпоширенішим хронічним захворюванням у світі. Також перевагою використання штучного інтелекту в такий спосіб є те, що пацієнти мають змогу збільшити свою автономію в плануванні лікування, шляхом отримання актуальної інформації про стан їх ротової порожнини. а лікарі в свою чергу – заощадити значний час на процесі діагностування захворювань. Деталі розробки, переваги, і процес навчання було викладено в статті [15]. Але якщо говорити про реальну розробку, яку можна власноруч протестувати, варто вказати на систему штучного інтелекту під назвою Diagnocat [16]. Ця система спроможна аналізувати наступні види зображення зубів: 3D панорамна томографію, панорамні рентгенівські знімки, прикуси та періапікальні рентгенограми.

Але попри все, незважаючи на цю кількість типів розроблених застосунків, залишається ще багато прогалин з точки зору програмного забезпечення в цій галузі. Однією з яких, що було виділено – недостатня увага пацієнтам. Адже якщо порівняти і проаналізувати орієнтацію додатків на тип користувача, то більшість розробок все ж таки спрямована на фахівців. Зрозуміло, що це в першу чергу їхня сфера роботи, але це також і область здоров’я, що стосується кожної людини. Саме людина відповідає за себе і за догляд за своє здоров’я, обирає лікарів, оплачує лікування, слідкує за дотриманням правил гігієни і несе відповідальність за власне харчування. Тому знання про власну історію хвороби може стати не тільки непоганим мотиватором для дотримання правил і рекомендацій, але і зробить людину більш обізнаною.

Так, з одного боку допускається, що невідома термінологія може тільки налякати і розтривожити, навіть попри використання якісних довідників. Але в свою чергу повна відсутність знань і інтересу до процесів лікування може призвести до пагубних наслідків. Адже якщо говорити про стоматологію, то це сама та область медицини, де найчастіше стикаються з прийняттям рішень з наявністю одразу декількома варіантів дій, починаючи від вибору матеріалу для адгезивної обробки зуба, і закінчуючи дилемою лікування чи все таки видалення. І інколи саме пацієнт має прийняти рішення, і чим більше він матиме критерій, тим ефективніше буде результат.

Варто звернути увагу, на те що більшість розглянутих в цьому розділі застосунків, розроблених для стоматологічної галузі, являють собою саме мобільні додатки. Більш детальний аналіз яких представлено в оглядовій роботі [17].

Роль мобільних пристроїв у житті сучасної людини має велике значення і значний вплив. І це дає можливість якісно це використовувати і маніпулювати поведінкою людини на її користь. Але не дивлячись на це мобільні додатки мають і свої недоліки. Основним з яких це те, що на відміну від веб-додатка, або ж веб-сайту, мобільні застосунки займають пам’ять на пристрої, що може завдавати незручності деяким користувачам.

Також варто відмітити, що в багатьох перелічених розробок не забезпечена кросплатформленість.

Але попри все основними факторами, згідно яких було дійдено висновку, що ступінь розв’язання завдань в даній галузі недостатня, це загальна кількість і популярність розроблень, яких не так вже і багато на даний момент. А якщо говорити саме про завдання, що було поставлене в даній кваліфікаційній роботі, то досі єдиного комплексного вирішення ще не існувало.

1.2. Призначення розробки та область застосування

Як об’єкт впровадження розроблюваного веб-орієнтованого застосунку розглядається веб-сайт «Зубний щоденник», який призначений для ведення дентальної хроніки, запису і аналізу стану ротової порожнини.

Призначення розробки полягає в:

– веденні дентальної хроніки стану ротової порожнини;

– веденні лікувальних процесів з боку фахівця стоматологічної галузі;

– відстеженні наявних симптомів та жалоб на стан здоров’я;

– контролі спливання термінів замін зубних щіток, задля забезпечення дотримання правил гігієни;

– підвищенні мотивації доглядати та слідкувати за здоров’ям ротової порожнини;

– простеженні кількості разів чистки зубів на день, задля впровадження даної звички, що призводить до зменшення появи зубного нальоту;

– покращенні розуміння поточного стану ротової порожнини, за допомогою графічної візуалізації;

– наданні простого і зручного інструменту для догляду і підтримання здоров’я ротової порожнини, а також зберігання даних про її стан.

Основна термінологія містить загальні терміни з областей інформаційної технології і медицини. Ключові слова включають: веб-орієнтований застосунок, веб-сайт, стоматологія, зуби, здоров’я, ротова порожнина, щоденник, дентальний, лікар, пацієнт, інформаційна система, медична інформатика, організація, запис, 2D графіка, веб-дизайн, діагноз, процес лікування, симптом, реляційна база даних, медицина, розробка, зубна щітка.

Терміни, які були використані в даній роботі, і потрібні задля розуміння основних нюансів елементної структури системи:

– зубний щоденник – це значення охоплює поняття всієї розроблюваної системи, і включає усі основні і додаткові структурні елементи системи;

– щоденник пацієнта – основна структурна частина системи «Зубний щоденник», яка призначена для ведення особистої дентальної хроніки кожного заохоченого;

– журнал пацієнтів – це основна частина системи «Зубний щоденник», яка доступна тільки для фахівців стоматологічної області, і призначена для введення клінічної історії кожного пацієнта, що надав для цього доступ;

– щоденник спостерігача – додаткова частина системи «Зубний щоденник», яка спрямована задля спостережень за динамікою розвитку заздалегідь вказаних симптомів;

– щоденник заміни зубної щітки – це додаткова частина системи «Зубний щоденник», в якій зберігаються дані про час використання зубних щіток;

– щоденник чистки зубів – це додаткова частина системи «Зубний щоденник», в якій зберігаються дані про чистку зубів.

Причини виникнення необхідності розробки програмного забезпечення:

– турбота за здоров’я ротової порожнини;

– важливість контролю стану ротовою порожнини;

– потреба в підвищенні відповідальності пацієнтів;

– потреба в підвищенні зацікавленості у здоров’ї власної ротової порожнини;

– потреба в покращенні дисциплінованого ставлення людей до власного здоров’я і підтримання правил гігієни;

– задля формування звички дотримання правил гігієни і догляду ротової порожнини;

– забезпечення зручного доступу до дентальної хроніки пацієнта;

– позбавлення цінності і потреби в паперової версії картки пацієнта;

– задля зручної системи організації і введення карток пацієнтів;

– задля зручного візуального аналізу поточного стану ротової порожнини.

Області, в яких може застосовуватися система, що розроблюється включають: стоматологію, протезування, оральна хірургію, імплантологію, дитяча стоматологію, ортодонтологію тощо.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

– освітня програма спеціальності 122 «Комп'ютерні науки»;

– навчальний план та графік навчального процесу;

– наказ ректора Національного технічного університету «Дніпровська політехніка» №350-с від 16.05.2023;

– завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованого застосунку ведення дентальної хроніки для запису і аналізу стана ротової порожнини»;

1.4. Постановка завдання

Мета: розробити веб-орієнтований застосунок, який надасть змогу покращити догляд за ротовою порожниною і організувати проведення продуктивного лікувального процесу.

Призначення: введення дентальної хроніки, формування звички відповідального догляду за власною ротовою порожниною і організація лікувального процесу зі сторони лікаря.

Технічно-економічна сутність завдання: покращити загальний рівень здоров'я ротової порожнини у якомога більшості людей, шляхом створення і впровадження системи «Зубного щоденника» в масове використання, при

цьому зменшити відсоток ускладнених діагнозів, через невчасне звертання за допомогою. А також надати весь необхідний функціонал для якісного і продуктивного обслуговування пацієнтів, з метою пришвидшення заповнення даних, що має збільшити час для більшої кількості прийомів, і тим самим покращити дохід і продуктивність фахівців галузі.

Цілком доцільним рішенням постає створення системи, що базуватиметься на веденні особистих записів як лікарем, так і пацієнтом. Це дасть змогу покращити порозуміння під час лікування, а також посприяти покращення його якості і продуктивності.

Додаток даної кваліфікаційної роботи призначений для управління медичними записами різних видів операцій, діагнозів, етапів лікування і результатів аналізів в сфері стоматологічного напрямку. Також у випадку якщо користувач системи являє собою фахівця цієї галузі то управління має здійснюватися над медичними картами всіх пацієнтів, що нададуть доступ.

Структура щоденника пацієнта має складатися з лікувань, в які мають додаватися записи, що в свою чергу мають наступні елементи:

– назва процедури або серії процедур та їх дані, включаючи файли;
– ім’я лікаря;
– дата запису;
– місце запису;
– коментар лікаря або ж рецепт;
– оцінка.

Основні показники, які мають відслідковуватися – це час, назва процедури і її область застосування. Адже на основі цих показників мають працювати функціональні складові системи.

Щодо системи журналу пацієнтів, то його структура має складатися з усіх щоденників, до яких пацієнти надали доступ задля редагування.

Також варто відзначити, що в система має включати і додаткові елементи, такі як: щоденник чистити зуби, щоденник заміни зубної щітки і щоденник спостерігача, який має на меті відслідковувати вказані симптомів.

Вихідна інформація – це опрацьовані і доведені до зручного для сприйняття користувачем виду вхідні дані, що стосуються записів процедур, симптомів, зубних щіток та іншого. Одним із ключових видів представлення результатів обробки вхідної інформації має являти собою графічна візуалізація поточного стану ротової порожнини. Основне призначення даного відображення – покращення розуміння стану здоров'я і підвищення мотивації за його доглядом.

Збір вхідної інформації має відбуватися за допомогою розроблених форм для уведення даних. Тому основною вимогою до організації збору та передачі в обробку вхідної інформації є зрозумілий і зручний інтерфейс для користувача. Також задля полегшення процесу заповнення даних і мінімізації ймовірності здійснення помилок, необхідно забезпечити як найбільш широкий спектр переліку інформації, щоб користувач міг її використовувати при заповненні форм, замість того щоб писати власноруч. Адже таким чином буде можливість не тільки збирати дані, а й упровадити автоматизацію в процеси їх організації і аналізу. Попри все це стосується переліку можливих процедур, частин ротової порожнини, лікарів і форми дати.

З причин того, що є ймовірність помилковості і не актуальності введених даних, є необхідність в контролі і коригуванні даних. Тому однією із вимог до системи є те, що важливі записи і дані, які будуть введені в систему мають мати можливість до видалення і коригування.

Розподіл функцій між персоналом і технічними засобами при різних ситуаціях вирішення завдань системи мають свої особливості, так як у системі передбачено, що її будуть використовувати два різних типів користувачів, а саме: пацієнт і лікар. До кожного з яких мають надаватися різні функціональні можливості.

Для користувача типу пацієнт мають бути доступні наступні функції:

– створення запису в щоденнику пацієнта;
– коригування існуючого запису із щоденника пацієнта;
– видалення існуючого запису із щоденника пацієнта;

– пошук необхідних записів щоденника пацієнта за заданим фільтром;

– створення симптому для відслідковування в щоденнику спостерігача;

– видалення симптому і даних про нього із щоденника спостерігача;

– додавання відміток в щоденник чистки зубів;

– редагування даних запису про чистку зубів;

– видалення запису чистки зубів;

– створення записів в журналі замін зубних щіток;

– коригування записів в журналі замін зубних щіток;

– видалення записів в журналі замін зубних щіток.

Користувачам типу лікар мають надаватися доступ до функцій:

– створення записів в журналі пацієнтів;

– редагування записів в журналі пацієнтів;

– видалення записів в журналі пацієнтів.

Задля кращого розуміння розподілу функціональних можливостей відповідно до ролей у системі, було розроблено USE-CASE діаграму на мові UML (рис.1.1).

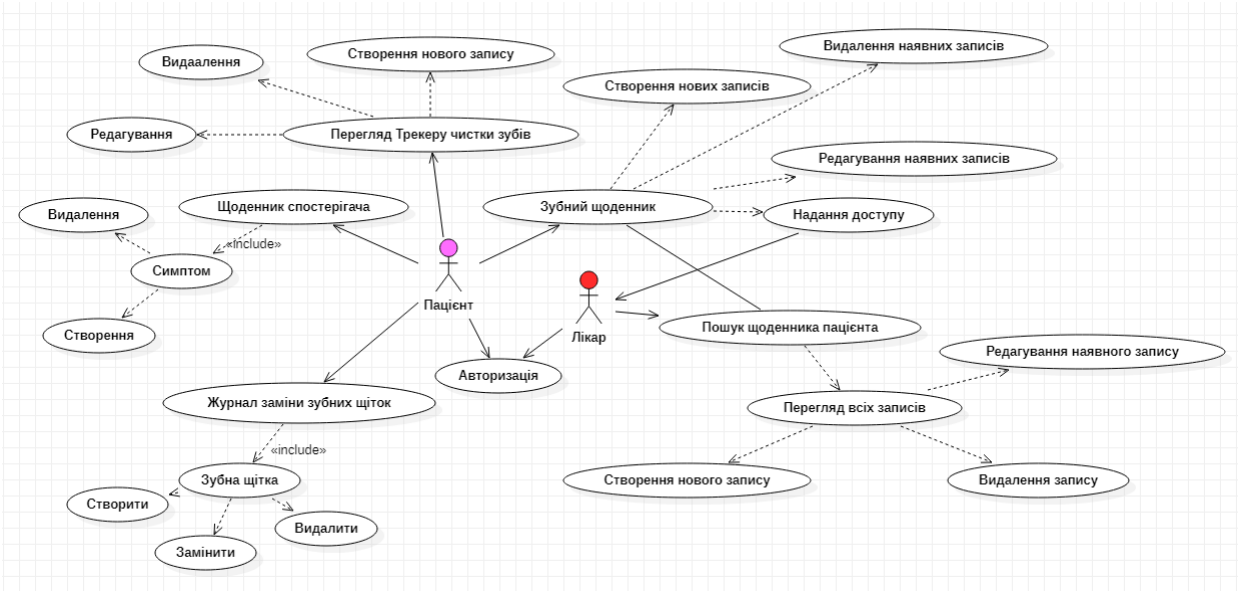

Рис.1.1. Use-case діаграма

1.5. Вимоги до інформаційної системи

1.5.1. Вимоги до функціональних характеристик

Дані мають вводитися користувачем стандартним шляхом, мається на увазі за допомогою клавіатури. Процес введення має відбуватися за допомогою розробленого програмного інтерфейсу де в основному мають використовуватися форми для введення, які можуть включати як відкриті відповіді так і вибір з представлених даних, склад яких був заздалегідь внесений у систему. Наприклад, такими даними являє собою перелік стоматологічних процедур або список всіх частин ротової порожнини. Також при заповнені інформації про лікаря, у випадку якщо той вже зареєстрований в системі, повинна бути наявна можливість пошуку і вибору з наявної інформації, задля мінімізації часу і ймовірності допущення помилок. Має бути реалізована можливість додавати файли з пристроїв, у випадку якщо потрібно зберегти наприклад панорамні знімки, або якийсь файл аналізів, що стосується тієї чи іншої процедури. Дата має обиратися за допомогою спеціально призначених для цього полів введення, з функцією саме вибору, а не введенням власноруч. Формат дати має мати вигляд – РРРР-ММ-ДД.

Всі введенні дані мають бути розділені по структурним елементам системи, які поділяються на основні і додаткові. Основні відповідають за головні процеси і призначення системи. Вони включають в себе: журнал пацієнтів і щоденник пацієнта. Додаткові елементи мають наступний склад: щоденник спостерігача, щоденник замін зубної щітки і щоденник чистки зубів.

Також відносно типу користувача мають відрізнятися не тільки доступні функціональні можливості, а і доступ до даних. Звісно користувачі мають мати можливість коригувати деякі налаштування щодо дозволів. Але в основному дані для різних типів користувачів мають відрізнятися не тільки інформаційним значенням, а і у деяких випадках поданням.

Організація даних має піддаватися фільтрації. В основному елементі системи має бути можливість фільтрувати дані за наступними параметрами: датою, типом процедурою, запису, лікування та області застосування. Так само має бути можливість здійснювати пошук. Це особливо стосується журналу пацієнтів, де пошук має здійснюватися за ім’ям пацієнта.

На основі вхідних даних мають виконуватися функціональні процеси. Основний приклад, який наводиться – це візуальне відображення поточного стану ротової порожнини, що залежачи від введених даних, графічно відображає деякі з проведених процедур, їх результат і наслідки.

Дані в системі мають подаватися користувачеві в декількох видах. Перш за все це список всіх зроблених записів. Це стосується записів у всіх структурних елементів системи.

Якщо ж говорити про подання даних у базі, то там має використовуватися саме реляційна модель.

1.5.2. Вимоги до інформаційної безпеки

Перш за все, варто визначити, що означає термін інформаційна безпека. Цьому питанню присвячена робота [18], де зазначається:

«… інформаційна безпека – це поточний стан захищеності об’єкта від інформаційних загроз, який визначається рівнем шкоди, що може бути заподіяна існуванню, функціонуванню чи діяльності об’єкта в разі реалізації цих загроз через:

а) використання неповної, несвоєчасної і недостовірної інформації;

б) здійснення негативного інформаційного впливу;

в) протиправного застосування інформаційних технологій;

г) несанкціонованого розповсюдження і використання інформації, порушення її цілісності, конфіденційності та доступності.»

В розробці даної роботи, однією з вимог забезпечення безпеки даних користувачів є присутність реєстрації акаунтів для кожного нового

користувача, а також авторизація для входу в акаунт. Основними вимогами до надійності реєстрації висуваються до паролю, а саме:

– кількість символів не менше восьми;

– максимальна допустима довжина – двадцять символів;

– наявність символів верхнього і нижнього регістрів;

– наявність хоча б трьох цифр;

– можливе використання тільки латинських символів;

– недопустимість використання наступних символів: /, \, *, &, %, $, та інших подібних.

Логіном при реєстрації і авторизації має виступати електронна пошта.

Однією із вимог організації безпеки, є метод зберігання паролей користувачів в базі даних, а саме зберігання у зашифрованому вигляді.

Також необхідно зазначити, що доступ до даних та можливих функцій залежить від типу користувача. Так, користувач типу пацієнт, має мати можливість переглядати і редагувати записи в своїх щоденниках. А якщо говорити про користувача типу лікар, то за замовченням, він не має доступу до щоденників його пацієнтів, і тільки після того, як пацієнт надасть доступ, лікар зможе вносити свої записи.

З причин можливості виникнення помилок, важливою умовою є контроль і обробка подібних ситуацій, одним із проявом чого є виведення повідомлень про причини і рекомендації з вирішення.

1.5.3. Вимоги до складу та параметрів технічних засобів

В роботі передбачено, що застосунок буде мати клієнт-серверну архітектуру. Тому кожна із сторін має мати різний склад та параметри технічних засобів. Всі вказані вимоги до параметрів технічних засобів є мінімальними, покращення ніяким чином не нашкодить роботі застосунку. Так, серверна частина складатиметься з двох серверів: сервер самого

застосунку і сервер бази даних. Для серверу застосунку виставляються наступні умови:

– ОП не менше 8Gb;

– SSD з об’ємом пам’яті не менше 250Gb;

– ЦП класу Intel Pentium (тактова частота не менше 2.4 ГГц, кількість ядер не менше 2);

– стабільне підключення до Інтернету.

Сервер бази даних також потребуватиме пам’ять не менше 1Tb.

Щодо використання застосунку користувачами, то так як розроблюваний застосунок веб-орієнтований, для його використання не потрібно якійсь складні технічні засоби, які потрібно біло б попередньо придбати, налаштувати та встановити на них операційну систему чи якесь додаткове програмне забезпечення тощо. Додаток розрахований на масове використання, то для усунень складнощів від встановлення і налаштування було прийнято рішення розроблювати саме веб-сайт.

Основна вимога до складу і параметрів технічних засобів – це наявність справного електронного пристрою, який дозволяє використовувати браузер і надає можливість вводити і виводити дані. Тобто таким пристроєм може бути як і мобільний телефон, персональний комп’ютер, планшет або ж ноутбук тощо. Пристрої вводу і виводу мають включати: дисплей, клавіатуру (фізичну або віртуальну) і мишку або сенсорний екран. До розмірів дисплею ставиться вимога – має бути співвідношення сторін дисплею – 16:10. Найкраще відображення інтерфейсу розраховано на дисплей з розширенням 1788x1005 пікселів. Загалом цей параметр обирається за вподобаннями користувачів. Хоча відмічається, що для більш комфортного введення записів і доступу до всіх можливостей графіки рекомендується використовувати дисплеї з діагоналлю більше середньої.

1.5.4. Вимоги до інформаційного та програмної сумісності

Через те, що планується розробка кросбраузерного застосунку, до браузерів всувається лише дві вимоги. По-перше, застосунок має коректно працювати з наступними браузерами: Google Chrome, FireFox, Opera, Microsoft Edge і Safari. По-друге, задля коректного відображення стилів має використовуватися одна із останніх версій браузера, як мінімум п’ята від останньої.

Для сервера застосунку висувається вимоги до встановлених компонентів, а саме:

– ОС – Windows, не нижче 10 версії;

– React.js;

– Node.js останньої версії.

Для сервера бази даних основною вимогою є наявність встановлення мови MySQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення інформаційної системи

Функціональне призначення розробленого застосунку складається з наступних пунктів:

– організувати зручне і спрощене ведення записів пацієнта про лікування;

– надати можливість лікарям отримувати доступ до записів пацієнтів;

– надати можливість лікарям редагувати внесені дані пацієнтами, за їх згодою;

– забезпечити інформаційну безпеку і відсутність втрати внесених даних користувачами;

– надати користувачу всі необхідні можливості для контролю за здоров’ям ротової порожнини;

– забезпечити користувача всіма засобами задля підтримання дисципліни в питанні дотримання правил гігієни та догляду за ротовою порожниною.

Функції, які реалізовані в розробленій програмі:

– ведення та організація дентальної хроніки, що включає в себе: ведення інформації про низку лікувальних процесів і процедур, з подальшою можливістю редагування та видалення;

– введення щоденника спостерігача, задля відслідковування турбуючих симптомів користувачів;

– введення та організація даних про щоденні чистки зубів;

– введення та організація даних про заміни зубних щіток;

– реєстрація нових користувачів;

– авторизація вже зареєстрованих користувачів;

– надання користувачам можливість переглядати всю введену ними інформацію;

– надання користувачам можливість надати доступ іншим користувачам до інформації про особисту дентальну хроніку;

– графічне відображення клінічної історії на 2D моделі ротової порожнини, в залежності від введених даних користувачем.

Експлуатаційне призначення розробленої системи полягає в наданні користувачу зручний спосіб організувати та контролювати лікувальні процеси, підтримувати мотивацію і дисципліну для дотримання правил гігієни і догляду, та підвищувати рівень відповідального ставлення до здоров’я власної ротової порожнини.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв’язуваної задачі не передбачають застосування математичних методів, при розробці веб-орієнтованого застосунку ведення дентальної хроніки для запису і аналізу стану ротової порожнини математичні методи не використовуються.

2.3. Опис використаних технологій та мов програмування

2.3.1. Клієнт-серверна архітектура

Розроблена інформаційна система являє собою веб-застосунок, який базується на архітектурі типу клієнт-сервер. Це попри все передбачає розробку центрального сервера, до якого в подальшому мають підключатися клієнти. Зв’язок клієнта і сервера відбувається через мережу Інтернет. В даному контексті мова йде саме про web-сервер. Візуальне відображення даної архітектури зображено на рис. 2.1. Також варто звернути увагу, що на

представленій діаграмі також зображено сервер бази даних, що теж є частиною архітектури розробленого застосунку.

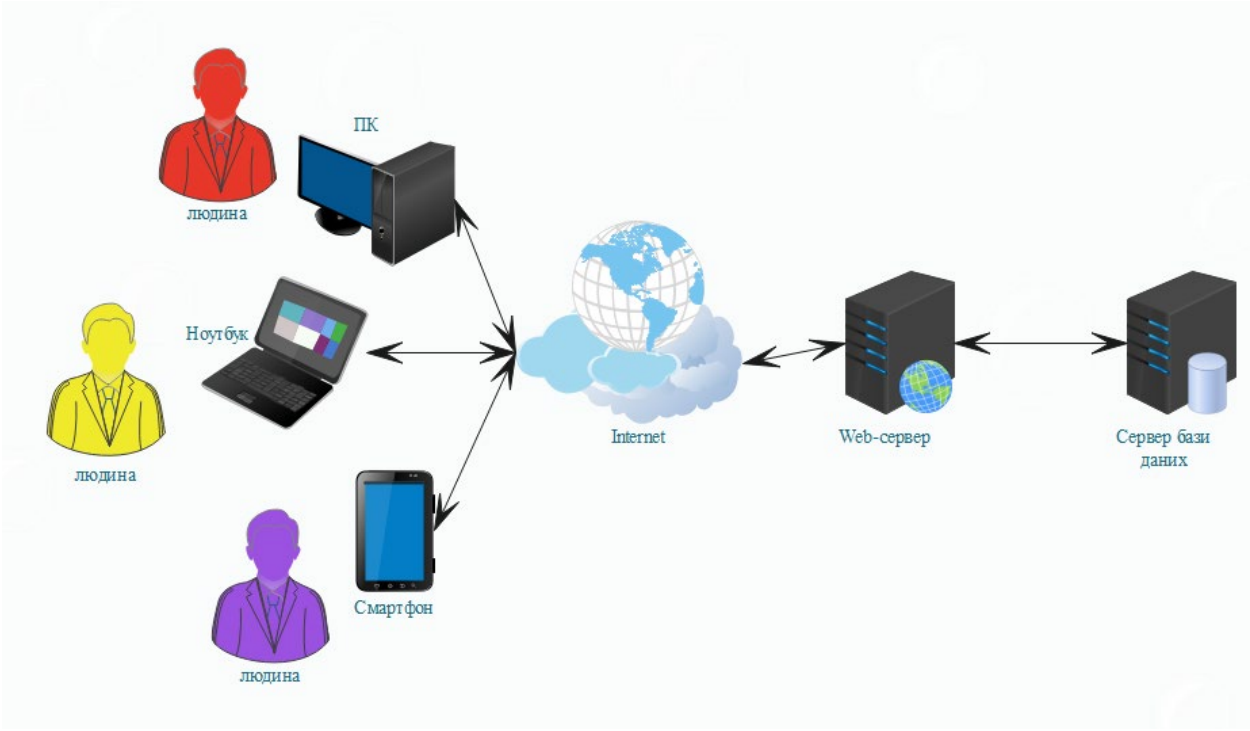

Рис.2.1. Архітектура розробленого застосунку

Перед усім зазначаються, що клієнт-серверна архітектура – це обчислювальна модель, в якій компоненти працюють в строго визначених ролях для комунікації і обміну даними один між одним. Центральний компонент даної архітектури – сервер, до якого підключаються клієнти, або ж клієнтські комп'ютери. В даному контексті сервер діє як виробник, а клієнт – як споживач.

Узагальнено процес взаємодії клієнтів і сервера можна описати наступним чином: клієнт, через мережеве з'єднання, надсилає запит на ресурс чи процес, що знаходиться на сервері, там в свою чергу запит обробляється, після чого відповідь доставляється назад до клієнта.

Прикладом даного зв'язку, може виступати введення користувачем URL-адреси в браузері для отримання доступу до того чи іншого веб-сайту, при цьому DNS-сервер шукає IP-адресу веб-сервера та передає її браузеру,

який в свою чергу генерує запит HTTP або HTTPS, а сервер, як виробник, надсилає файли. Клієнт, отримуючи їх, потім продовжує надсилати наступні запити.

В залежності від рівнів, клієнт-серверна архітектура поділяється на наступні типи: дворівнева, трирівнева та N-рівнева.

Так як в розробленій системі передбачено використання сервера бази даних, окрім веб-сервера і клієнта, даний тип архітектури має назву трирівневої. Тут клієнт надсилає запит на сервер додатків, який оброблюючи отриманий запит, запитує дані з сервера бази даних, отримавши які, опрацьовує, і вже після того відповідає клієнту.

Цей тип архітектури, має ряд переваг. По-перше, централізоване керування полегшує обмін ресурсами та даними між різними платформами. По-друге, це гнучкість, адже розглянуту архітектуру легко комбінувати з іншими типами архітектур на стороні клієнта або сервера. По-третє, доступність. Також, до переваг, можна віднести: розширюваність і масштабованість.

2.3.2. Стек програмування MERN

Перш за все потрібно зазначити, що стек – це комбінація всіх технологій, які були задіяні при розробці застосунку. У даному розробленому додатку за основу було взято стек програмування MERN, хоча і не в повному обсязі.

Стек MERN є популярним стеком технологій для створення одно сторінкової програми (SPA – Single page Application). Він складається з чотирьох ключових технологій:

– MongoDB – база даних NoSQL, яка зберігає дані в формі колекцій і документів;

– Express(.js) – веб-фреймворк Node.js, який робить розробку додатків простішою і швидшою;

– React(.js) – інтерактивна бібліотека JavaScript для створення інтерфейсу користувача (UI – User Interface);

– Node(.js) – серверний фреймворк, який дозволяє виконувати код JavaScript на веб-сервері. Node і Express разом утворюють веб-сервер.

Дана технологія дозволяє легко побудувати динамічні веб-сайти з трирівневою архітектурою.

За інтерфейс клієнтської частини відповідає React.js, і являє собою найвищим рівнем стеку MERN. Також дана бібліотека дозволяє створювати складні інтерфейси за допомогою простих компонентів, підключаючи їх до даних на сервері та відтворюючи їх як HTML. React Client надсилає HTTP-запити та отримує HTTP-відповіді за допомогою Axios. React Router – використовується для переходу до сторінок, і налаштування навігації.

React.js підтримує компоненти, будівельні блоки інтерфейсу користувача, що по суті є функціями JS. Ці компоненти сприяють багаторазовому використанню коду та полегшують загальне розуміння веб-програми.

При застосуванні React.js зникає потреба у використані шаблонів для автоматизації створення елементів HTML або DOM, так як для цим займається повнофункціональна мова програмування JavaScript, використовуючи JSX розширення (JavaScript XML).

Наступний рівень стеку – серверна структура Express.js, яка працює на сервері Node.js. Express.js — це серверний фреймворк програми, який обгортає HTTP-запити та відповіді та дозволяє легко зіставляти URL-адреси з функціями на стороні сервера. Node.js Express експортує REST API і взаємодіє з базою даних.

REST API (також відомий як RESTful API) — це інтерфейс прикладного програмування, який відповідає обмеженням архітектурного стилю REST і дозволяє взаємодіяти з веб-службами RESTful. REST розшифровується як репрезентативна передача стану.

Node.js – це кросплатформне середовище виконання, яке запускає двигун V8 JavaScript, що дозволяє Node.js бути дуже продуктивним. Сам же по собі Node.js працює в одному процесі, без створення нових потоків для кожного запиту. Також варто зазначити, що даний фреймворк використовує асинхронність функцій вводу/виводу, запобігаючи при цьому блокуванню коду JavaScript.

Щодо останнього рівня стеку, який відповідає за базу даних, то у MERN він представлений MongoDB, але в розробленому застосунку цей елемент було замінено на реляційну базу даних MySQL. Тобто використаний стек програмування набуває вигляду, який зображений на рис. 2.2.

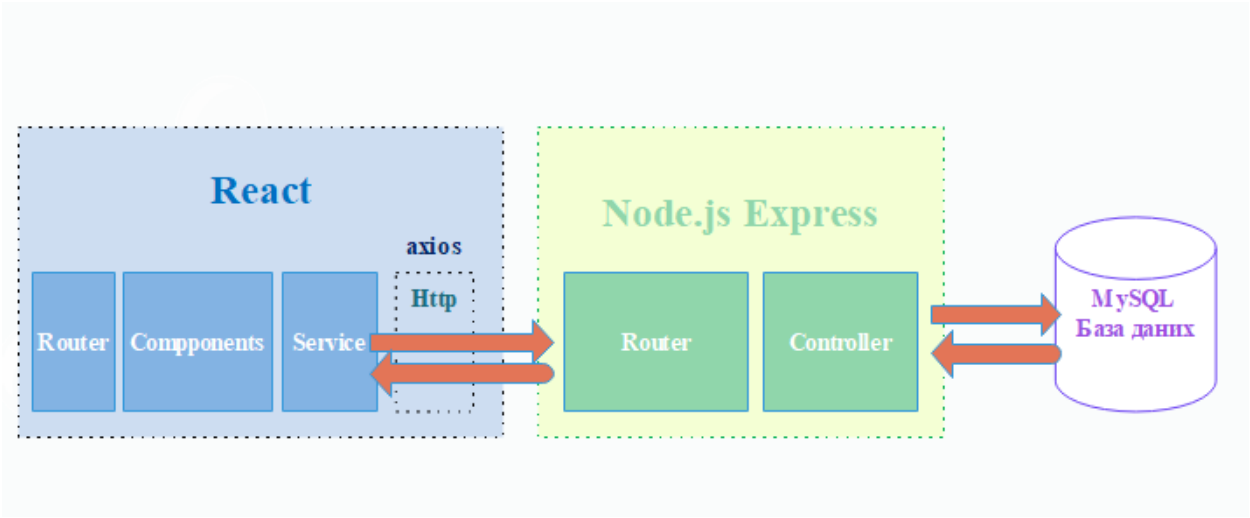

Рис. 2.2. Структура застосованого стеку програмування

MongoDB, що є представником бази даних типу NoSQL, яка в свою чергу має наступний ряд недоліків:

– відсутність підтримки транзакцій ACID (атомарність, узгодженість, ізольованість, довговічність), через що існує ризик помилки синхронізації даних на кінцевих вузлах;

– відсутність стандартизації;

– можливість ризику невдалого розповсюдження на вузлах.

Загалом MongoDB часто використовують саме через полегшення масштабування, що забезпечує цей вид бази даних. Але так як розроблений

додаток не планують масштабувати, то ця перевага втрачає свою цінність. Тому було прийнято рішення замінити дану технологію. Вибір впав саме на стандартного представника реляційної моделі бази даних – MySQL.

SQL орієнтовану базу даних було обрано через наступний ряд преваг, які вона має, а саме:

– відповідність вимогам ACID (атомарність, узгодженість, ізольованість, довговічність), що забезпечує цілісність даних та точність транзакцій;

– у кожній таблиці розміщується по декілька категорій даних у стовбцях;

– кожен рядок таблиці має унікальний екземпляр даних для категорій.

Але загалом цей стек технологій було обрано через наступні його переваги:

– рентабельність;

– підтримка SEO;

– краща продуктивність;

– краща безпека;

– швидкість;

– відкритий код;

– легке перемикання між клієнтом і сервером.

2.3.2.1. Використані модулі стеку MERN

Окрім фреймворку Express, під час розробки було встановлено і використано низку додаткових модулів, до них відносяться:

– nodemon – утиліта, яка відстежує будь-які зміни у проекті, зроблені під час розробки. при цьому автоматично перезапускаючи сервер-застосунку;

– dotenv – модуль, який завантажує змінні середовища з файлу .env в process.env;

– jsonwebtoken – модуль для node.js, що дозволяє безпечно автентифікувати користувачів системи, фактично не зберігаючи жодної інформації про них у системі;

– bcrypt.js – модуль, який забезпечує захист даних, шляхом шифрування;

– cors – node.js модуль для надання проміжного програмного забезпечення Connect/Express, яке використовують для налаштування політики CORS із різними параметрами;

– cookie-parser – модуль node.js, який допомагає створювати та керувати файлами Cookie залежно від надісланим користувачем запиту;

– axios – це бібліотека, яка використовується для зв'язку з серверною частиною, шляхом надсиланням запитів до API, і отримання даних, з подальшим опрацюванням, взаємодіючи з React.js.

2.3.3. Шаблон проектування MVC

Відомо, що MERN походить від стеку програмування MEAN (MongoDB, Express, AngularJS, Node.js), який замість React.js використовує AngularJS, що в свою чергу базується на одному з відомих шаблонів проектування MVC. На відміну від Angular, React не є повноцінним MVC фреймворком, а являє собою лише бібліотеку JavaScript для побудови інтерфейсів користувачів. Але при цьому якщо розглядати структуру MERN комплексно, то можна цілком успішно простежити та інтерпретувати структуру MVC шаблону в розробку, навіть за умови використання MySQL.

MVC (Model-View-Controller) – це шаблон розроби програмного забезпечення, який використовується для реалізації інтерфейсів користувача, даних і логіки керування. Він використовується для сучасних веб-додатків, оскільки він дозволяє програмі бути масштабованою, зручною для обслуговування та легкою для розширення.

Цей шаблон складається з трьох частин:

– Model (модель) – відповідає за керування даними та бізнес-логікою;

– View (представлення) – відповідає за керуванням макетів та відображення;

– Controller (контролер) – спрямовує команди до частин моделі та виду.

Структура даного шаблону візуально описана за допомогою діаграми, яка зображена на рис.2.3.

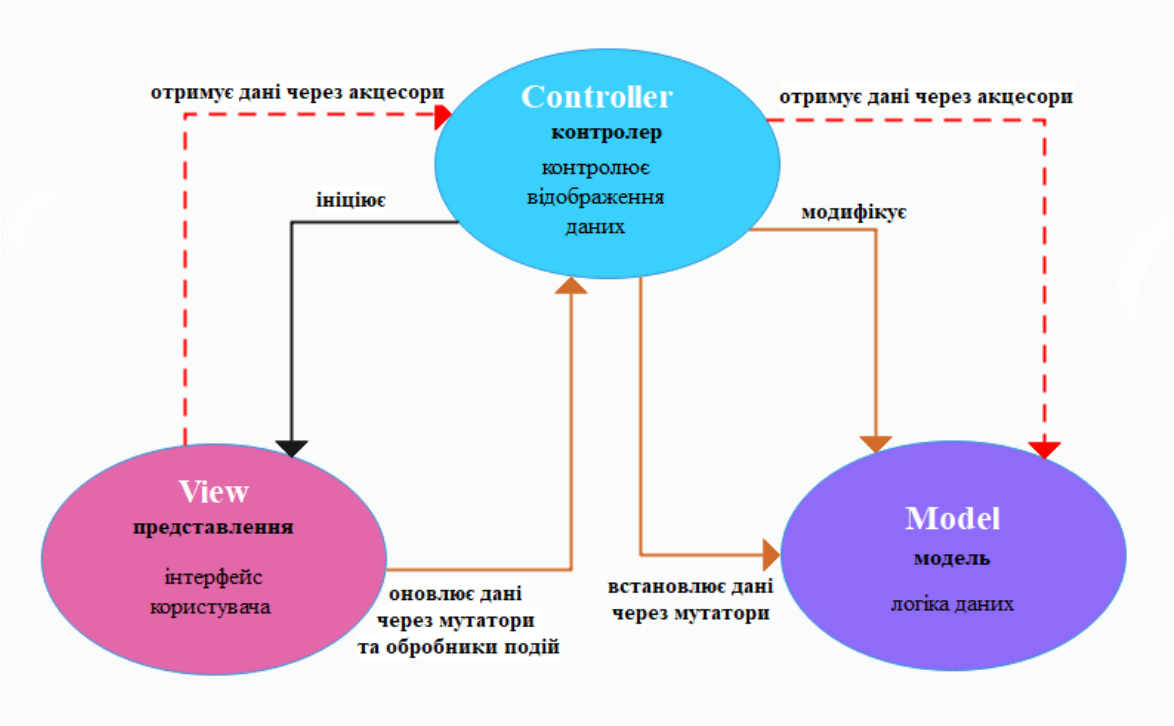

Рис. 2.3. Діаграма шаблону проектування MVC

Робота моделі полягає в тому, щоб просто керувати даними, незалежно від типу ресурсу їх надходження.

Робота представлення – вирішувати, що користувач бачитиме на своєму екрані та як.

Контролер відповідальний за отримання, зміну та надання даних користувачеві. Також, контролер є сполучною ланкою між представленням і моделлю.

Щодо інтерпретування даного шаблону в стек технології MERN. Клієнтська частина, що розробляється за допомогою React.js – реалізує

елемент представлення, і відповідає за інтерфейс користувача. На серверній частині за допомогою Node.js розробляються класи контролерів, які мають на меті контролювати відображення даних. Щодо елемента моделі, яка відповідає за взаємодію з базою даних, то в стандартному стеці MERN, за цю частину на себе може взяти відповідальність Moongoose – ODM бібліотека JavaScript, яка використовується для зв'язку з MongoDB. Але у випадку використання бази даних MySQL ця роль припадає на ORM бібліотеку – Sequelize.

Варто зазначити, що ORM (Object Relational Mapping) – це техніка для запитів або виконання CRUD операцій (Create-Read-Update-Delete – створення, читання, оновлення і видалення) до бази даних, переважно реляційних, використовуючи об'єктно-орієнтовану парадигму.

У свою чергу ODM (Object Document Mapping) теж саме, що і ORM, але призначена для нереляційних або розподілених баз даних, тобто зіставлення об'єктної моделі та бази даних NoSQL.

Але в даній розробці за зв'язок з базою даних і за елемент моделі відповідають власноруч прописані методи і функції, включаючи CRUD операції.

2.3.4 Методологія БЕМ

Для розробки інтерфейсу користувача було застосовано методологію БЕМ. Це компонентний підхід до веб-розробок, в основі якого лежить поділ інтерфейсу на незалежні блоки. Завдяки цьому стає можливо легко та швидко розробляти інтерфейси будь-якої складності, з можливістю повторно використовувати існуючий код.

БЕМ розшифровується як Блок Елемент Модифікатор. Блок – це функціонально незалежний компонент сторінки, який можна повторно використати. За правилами методології назва блоку має характеризувати його сенс, а не стан. Також блоки можна вкладати один в одного.

Елемент – це частина блоку, яка не використовується окремо від нього. Елементи так як і блоки можна вкладати один в одного.

Модифікатор – це сутність, що визначає стан, поведінку чи зовнішній вигляд елемента. Існують наступні типи модифікаторів:

– булевий – використовують, коли важливо зазначити наявність чи відсутність модифікатора;

– ключ-значення – використовують при необхідності зазначити значення модифікатора.

Основними моментами застосування цієї методології являють собою іменування структурних елементів і файлова структура. Існує декілька способів організовувати іменування та файли. Під час розробки застосунку було застосовано стандартне іменування за шаблоном – «блок__елемент_модифікатор». Варто також зазначити, що у всіх технологіях (SCSS, HTML, JS) одна й та сама БЕМ-сутність називається однаково. Адже основна ідея даної методології полягає в тому, щоб вкласти сенс в імена для покращення і пришвидшення розуміння та написання коду.

Щодо файлової структури, то було застосовано підхід з назвою Flex. Він є найбільш гнучкий з усіх, так як поєднує у собі підходи Nested з Flat. При цьому кожен блок відповідає окремому каталогу. Елементи та модифікатори можуть реалізовуватися в окремих файлах.

Найбільш продуктивно розкриває себе БЕМ підхід з використанням препроцесорів при написання стилів для інтерфейсу.

2.3.5 Використані мови програмування

На базі стеку програмування MERN використовується тільки мова програмування JavaScript. Що помітно спрощує налаштування взаємодії між клієнтською та серверною частинами.

JavaScript – це високорівнева динамічна інтерпретована мова програмування, яка добре підходить для стилів об’єктно-орієнтованого та

функціонального програмування. Це найпоширеніша мова для написання веб-сайтів, що і було початковою метою при її створені. Але наразі, за рахунок Node.js, JavaScript вийшов за межі веб-браузерів.

Зміні JS не типізовані. Синтаксис частково базується на мові програмування Java, хоча вони зовсім не пов'язані між собою. JavaScript отримує свої першокласні функції від Scheme, а свій прототип успадковує від маловідомої мови Self. Стандарт мови JavaScript являє собою ECMAScript.

Ця мова була обрана по причині того, що розробляємий застосунок є веб-орієнтованим, доступ до якого клієнтам буде надаватися за допомогою веб-браузера. А для цих цілей JavaScript підходить найбільше у порівнянні з іншими мовами. На базі цього далі було обрано, вже згадуваний, стек програмування MERN і шалон проектування MVC.

Але попри вже згаданих задіяних технологій в попередніх розділах, також варто відмітити використання препроцесора SCSS для створення оформлення для інтерфейсу та мову розмітки HTML, яка будує каркас сторінок з клієнтського боку застосунку, будучи інтегрованою в компоненти JSX.

Препроцесор – це надбудова над CSS, яка розширює його можливості, додаючи нові синтаксичні конструкції. Існує декілька популярних препроцесорів, таких як: SASS, SCSS, LESS тощо. В даному проекті було використаний саме SCSS препроцесор, який розшифровується, як Sassy Cascading Style Sheets.

Дана технологія має наступні переваги:

– зручність редагування коду;

– забезпечення кросбраузерності;

– скорочення коду;

– використання змінних, функцій, операторів, математичних операцій та іншого.

Також однією із основних технологій, що була використана і без якої не обходиться ні одна веб-орієнтована розробка, являє собою мова розмітки

гіпертексту HTML (HyperText Markup Language) – основний будівельний блок Інтернету, що визначає структуру та значення веб-контенту.

2.4. Опис структури програми та алгоритмів її функціонування

2.4.1. Структура системи

2.4.1.1. Декомпозиція

Одним із перших етапів проектування системи, що було використано, являє собою метод декомпозиції. Він був застосований до проектування структури об’єктів та їхнього функціонування в застосунку.

Декомпозиція – це поділ великих елементів проекту на більш дрібні і легко керовані елементи. Зазвичай результат декомпозиції представляють у вигляді ієрархічного графа. Але задля зручного сприйняття і відображення, декомпозиція проекту було представлено за допомоги таблиці (табл. 2.1).

Таблиця 2.1

Декомпозиція структури проекту

<table>
<tr><td rowspan="11">Зубний щоденник</td><td rowspan="6">Особистий кабінет</td><td>Реєстрація/ Логін</td><td colspan="4">Пошта/пароль</td></tr>
<tr><td rowspan="5">Особисті дані</td><td colspan="4">ПІБ</td></tr>
<tr><td colspan="4">Вік</td></tr>
<tr><td colspan="4">Стать</td></tr>
<tr><td colspan="4">Номер телефону</td></tr>
<tr><td rowspan="2">Лікар</td><td colspan="3">Робоча адреса</td></tr>
<tr><td colspan="3">Спеціалізація</td></tr>
<tr><td>Лікар</td><td>Журнал пацієнтів</td><td colspan="4">Щоденник пацієнта</td></tr>
<tr><td rowspan="4">Пацієнт</td><td rowspan="4">Щоденник пацієнтів</td><td rowspan="4">Лікування</td><td rowspan="3">Додати/ Змінити</td><td colspan="2">Назва</td></tr>
<tr><td colspan="2">Тривалість</td></tr>
<tr><td rowspan="2">Запис</td><td>Додати/змінити</td></tr>
<tr><td>Видалити</td></tr>
<tr><td colspan="4">Видалити</td></tr>
</table>

Зубний щоденник	Пацієнт	Щоденник пацієнтів	Запис	Додати/ Змінити	Дата			
					Адреса			
					Лікар	Пошук/ Створити	ПІБ	
							Номер	
							Адреса	
							Спец-я	
					Рецепт лікаря			
					Оцінка			
					Процедура	Додати/Змінити		
						Видалити		
				Видалити				
			Процедура	Додати/ Змінити	Файл	Додати/ Змінити	Дата	
							Назва	
							Місця	
							Файл	
						Видалити		
					Назва			
					Область застосув-я	Частини ротової порожнини		
				Видалити				
			2D прототип ротової порожнини	Анімація	Видалення зуба			
					Екстірпація пульпи			
					Адгезивна реставрація			
					Брекети			
					Зубний наліт			
			Фільтр пошуку	За датою				
				За процедурою				
				За областю застосування				
			Поділитися	Пошук лікаря				
		Щоденник спостерігача	Симптом	Додати/ змінити	Назва			
					Дата			
					Рівень болю	від 1 до 10		
				Видалити				
		Щоденник зубних щіток	Зубна щітка	Додати/	Дата			
					Тип			
					Жорсткість			
					Колір			
				Видалити				
		Щоденник чистки зубів	Трекер	З ранку				
				Після прийоми їжі				
				Перед сном				
			Відео-інструкція чистки зубів					
			Історія	Запис	Змінити			
					Видалити			
		Нагадування	Про заміну щітки					
			Про відвідування лікаря					
			Про чистку зубів					
	Дизайн	Ілюстрація зубних щіток						

2.4.1.2. Логічна структура

Розроблений застосунок представляє собою динамічний веб-сайт, що складається з наступних веб-сторінок:

– головна сторінка;
– реєстрація;
– авторизація;
– особистий кабінет;
– журнал пацієнтів;
– щоденник пацієнта;
– щоденник чистки зубів;
– щоденник зубних щіток;
– щоденник спостерігача;
– лікування;
– запис;
– процедура.

Всі ці компоненти так чи інакше пов’язані між собою, утворюючи логічну структуру застосунку.

Так, з головної сторінки є можливість перейти на сторінки реєстрації та авторизації, після чого залежачи від введених даних користувач переноситься до сторінки особистого кабінету. В залежності від типу користувача (пацієнт або лікар) доступ до сторінок відрізняється. Пацієнтові надається можливість перейти до сторінок «Щоденник пацієнта», «Щоденник чистки зубів», «Щоденник зубних щіток» та «Щоденник спостерігача». Лікарю доступні сторінка «Журнал пацієнтів» . Останній пов’язаний з щоденником пацієнтів. Щодо сторінки «Щоденник пацієнта» тут можливі два сценарії. В першому можна перейти до сторінок «Лікування», «Процедури» та «Записи» відразу. В другому ж відбувається все по наступному сценарію: «Лікування→Запис→Процедура». Описана логічна структура представлена на рис. 2.4.

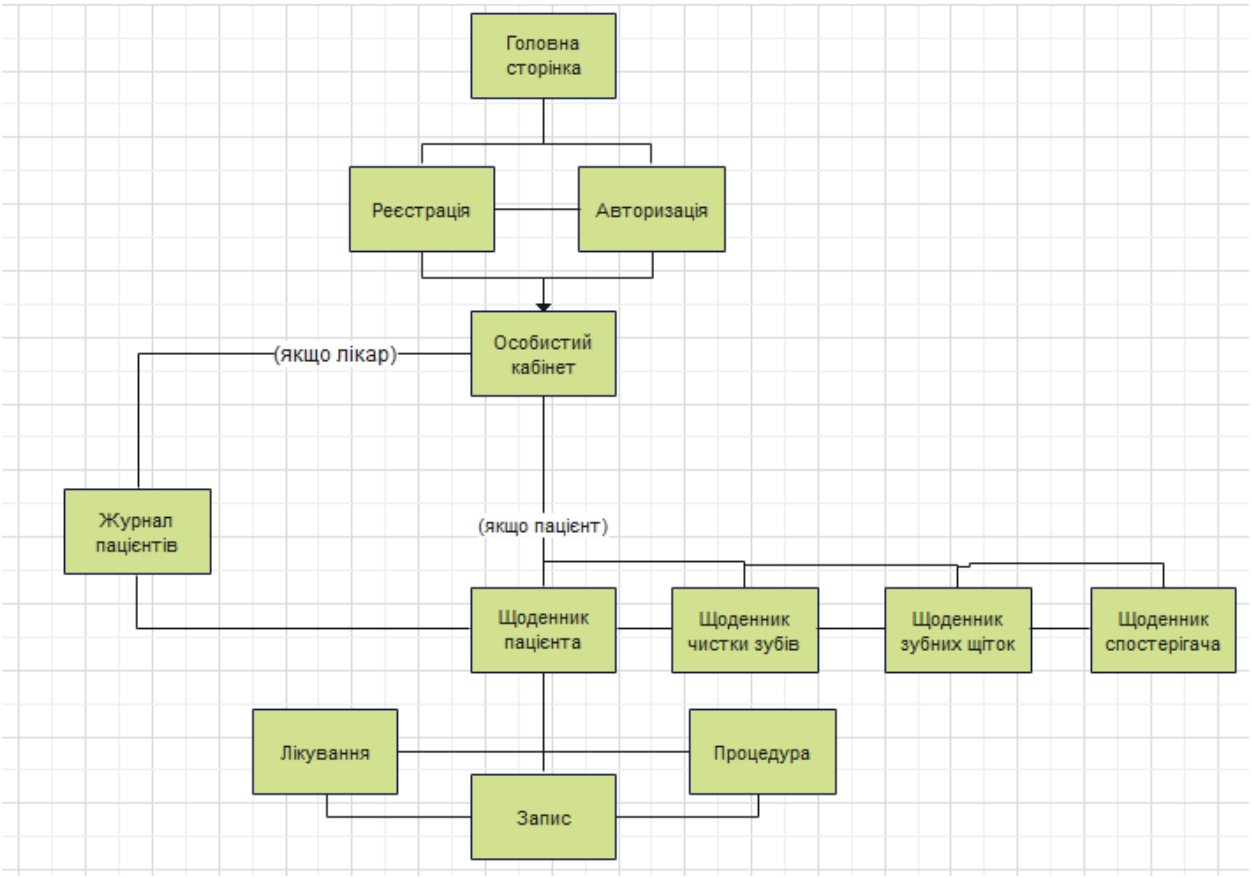

Рис. 2.4. Діаграма логічної структури веб-сайту

2.4.1.3. Сценарії діяльності відносно ролей у системі

Як вже зазначалося в попередніх розділах в розробленій системі присутньо два типи ролей: пацієнт і лікар. Залежно від цього відбувається розподіл доступу до компонентів системи та функціональних можливостей. Деякі з них були вже описані в попередніх розділах за допомогою декомпозиції та діаграми логічної структури застосунку.

Діяльність користувачів в системі також залежить від типу ролі. Задля кращого розуміння процесів та алгоритмів системи було розроблено кілька основних сценаріїв поведінки користувачів у системі, використовуючи діаграми діяльності. Розроблені сценарії діяльності включили: сценарій ведення записів лікарем (рис.2.5) та введення записів пацієнтом в «Щоденнику пацієнта» (рис.2.6).

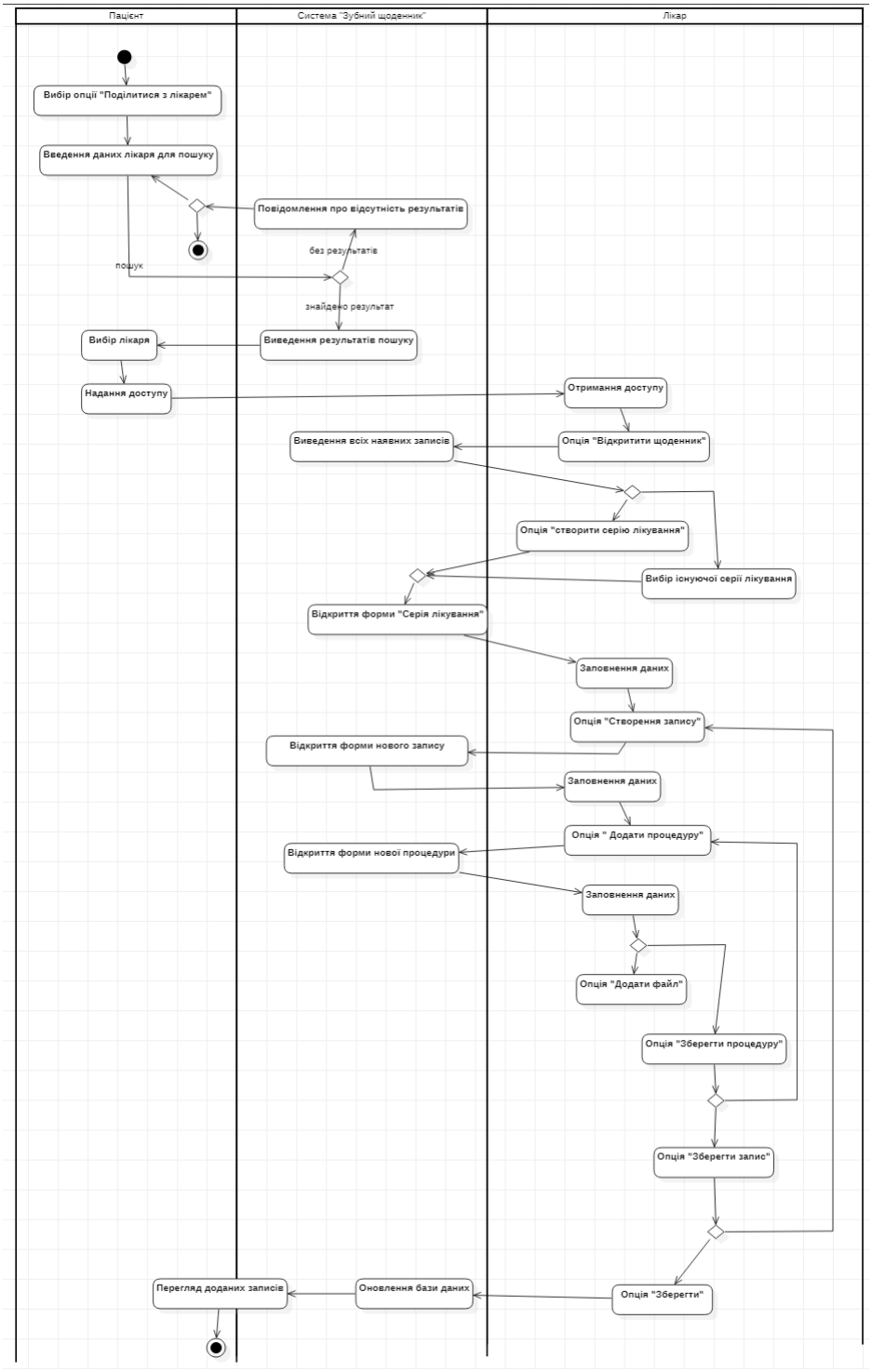

Рис.2.5. Діаграма діяльності «Ведення записів лікарем»

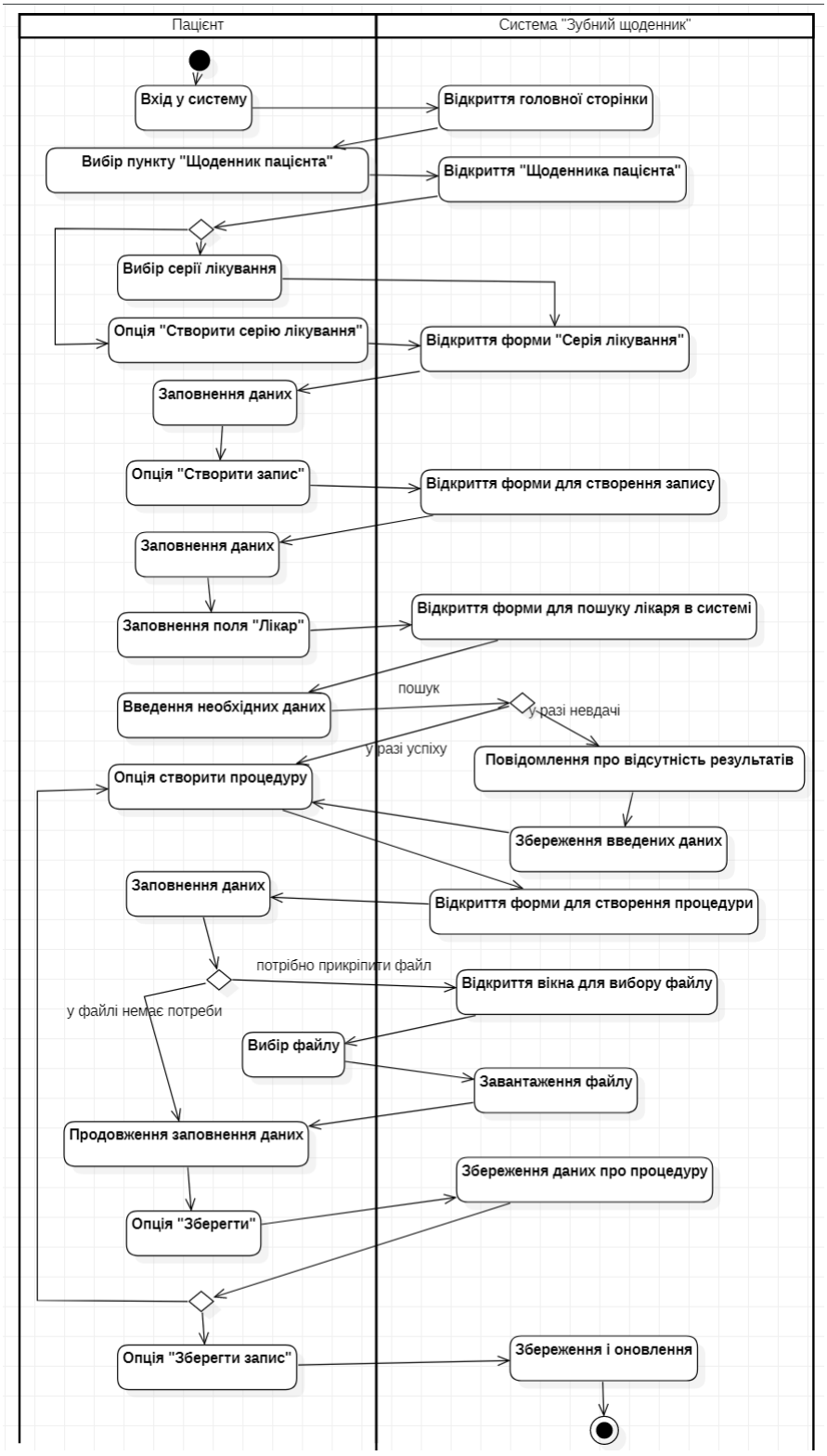

Рис. 2.6. Діаграма діяльності «Введення записів пацієнтом»

2.4.1.4. Файлова структура

Структура застосунку складається з двох частин клієнтської та серверної частини. Серверна частина міститься в файлі під назвою «backend». В цій директорії знаходяться основні файли запуску застосунку: index.js і App.js. В папці «controllers» розміщуються файли контролерів системи, які зв'язують сервер з елементом моделі, що підключається до бази даних. Яка в свою чергу реалізована в директорії «dbconnect». Навігаційні файли розміщуються в папці «routers». В папці «node_modules» знаходяться всі основні і додатково підключені модулі Node.js, які були використані в проекті. Вони включають:

– bcryptjs;

– cookie-parser;

– cors;

– express;

– jws;

– mysql;

– nodemon.

Перелік всіх файлів представлений на рис.2.7.

Взаємодія всіх модулів відбувається наступним чином. Модулі контролерів підключаються до модулів моделі. Навігаційні файли взаємодіють з контролерами. І відповідно основний файл серверу вже використовує модулю навігації.

Клієнтська частина розміщена в директорії «frontend». Так як вона базується на використані бібліотеки React, структура має вигляд відповідно до вимог даної технології, а саме включає папки: «node_modules», «public» і «src». Саме в останній розміщуються всі модулі елемента view, які розділені по папкам «components», «context», «images», «style», «view». Файл запуску клієнтської частини знаходиться в основній директорії. В папці «view» розміщені основні модулі реалізації сторінок інтерфейсу користувача. В

папці «components» знаходяться окремі елементи інтерфейсу і їх оформлення. Детальний перелік всіх файлів можна переглянути на рис.2.8. В папці «style» розміщені функції і змінні препроцесорів і підключені шрифти. Директорія «upload» слугує як локальне сховище, де зберігаються файли завантажені в базу даних, а саме файли прикріплені до процедур.

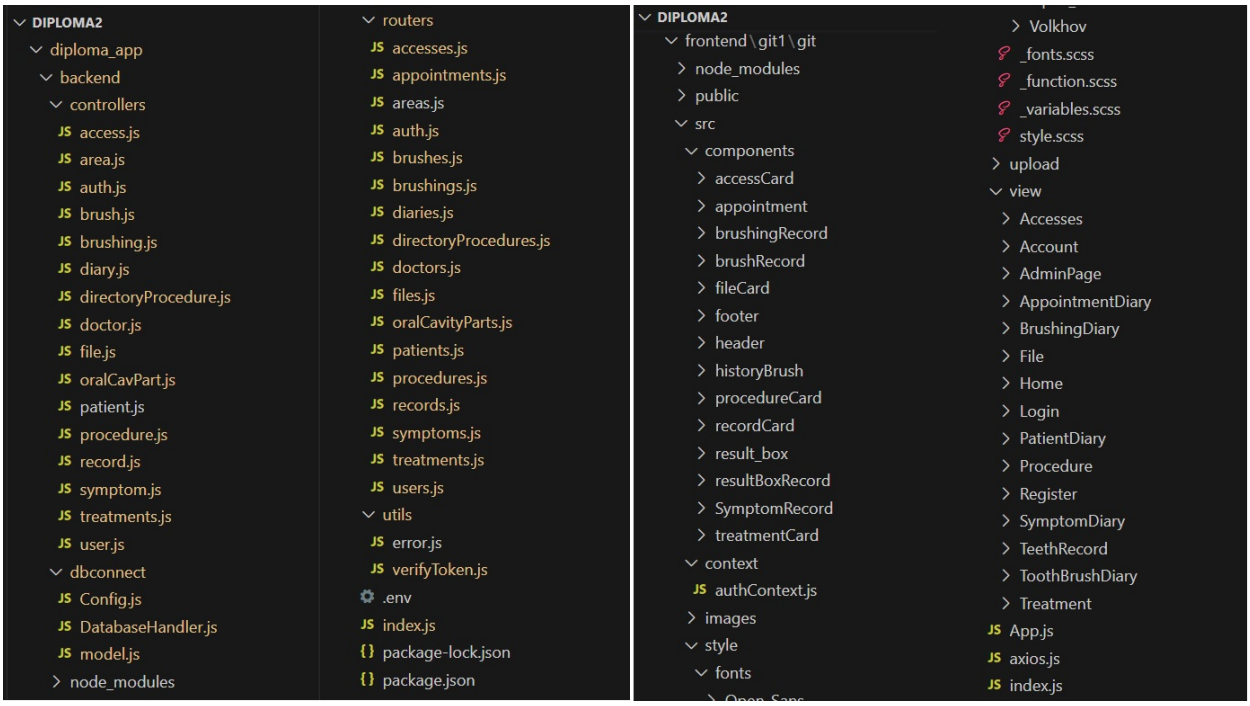

Рис.2.7. Файлова структура проекту

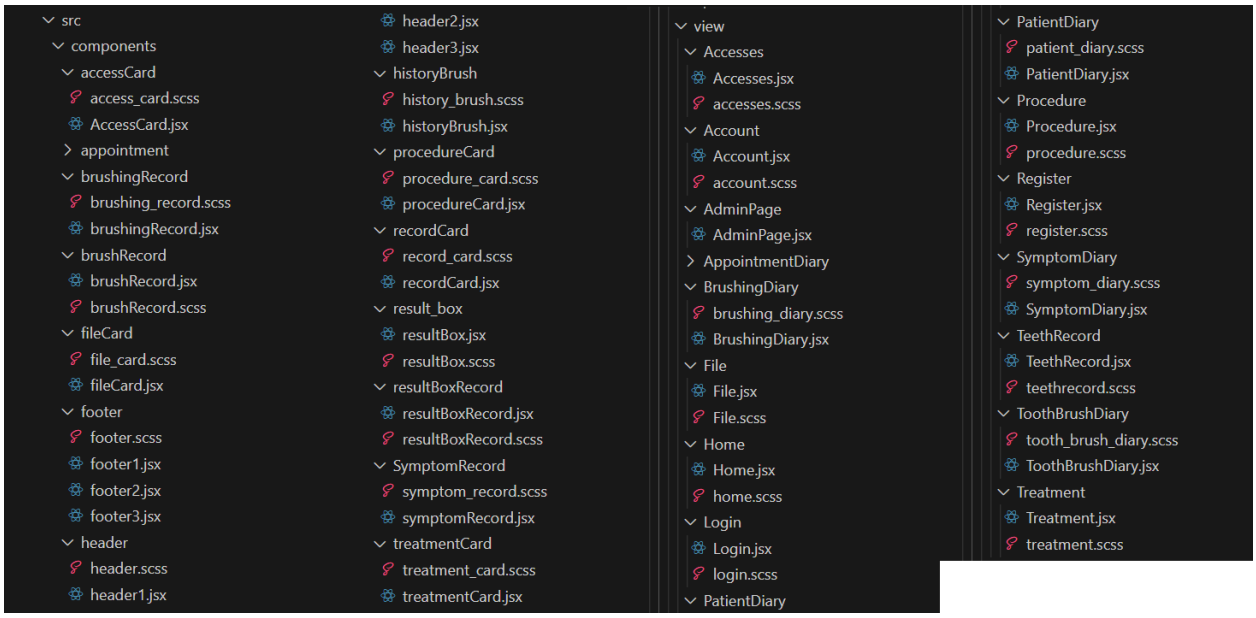

Рис.2.8. Розгорнута файлова структура клієнтської частини

2.4.2. Структура бази даних

База даних для розробленого застосунку має мережеву структуру з розгалуженнями, що дозволяє розробляти складніші типи зв’язків для створення найбільш точної і продуктивної системи.

2.4.2.1. Концептуальна модель бази даних

Структура бази даних повністю залежить від структури системи. Тому базуючись на цьому було виділено наступні елементи:

– пацієнт;
– лікар;
– лікування;
– чистка зубів;
– зубна щітка;
– симптом;
– щоденник спостерігача;
– щоденник пацієнта;
– щоденник чистки зубів;
– щоденник зубних щіток.

Кожні з цих компонентів так чи інакше взаємодіють між собою, тому було встановлено зв’язки і залежності між ними. Узагальнено логіку їх поєднання можна описати наступним чином: пацієнт звітує про чистку зубів, заміну зубних щіток і описує наявні симптоми та жалоби по здоров’ю, всі ці дані зберігаються в визначених для цього місцях, в системі вони мають назву щоденник або журнал. Дані про симптоми зберігаються в щоденнику спостерігача, дані про заміни зубних щіток – в щоденнику зубних щіток, та дані про кількість чистки зубів – в щоденнику чистки зубів. Після відвідування лікаря, пацієнт має змогу занотувати для себе важливі нюанси

лікування в власному щоденнику пацієнта. Лікар теж може додавати записи в щоденник пацієнта, якщо той надасть йому доступ до цього.

Для кращого розуміння концепції бази даних було створено концептуальну схему, яка зображена на рис. 2.9.

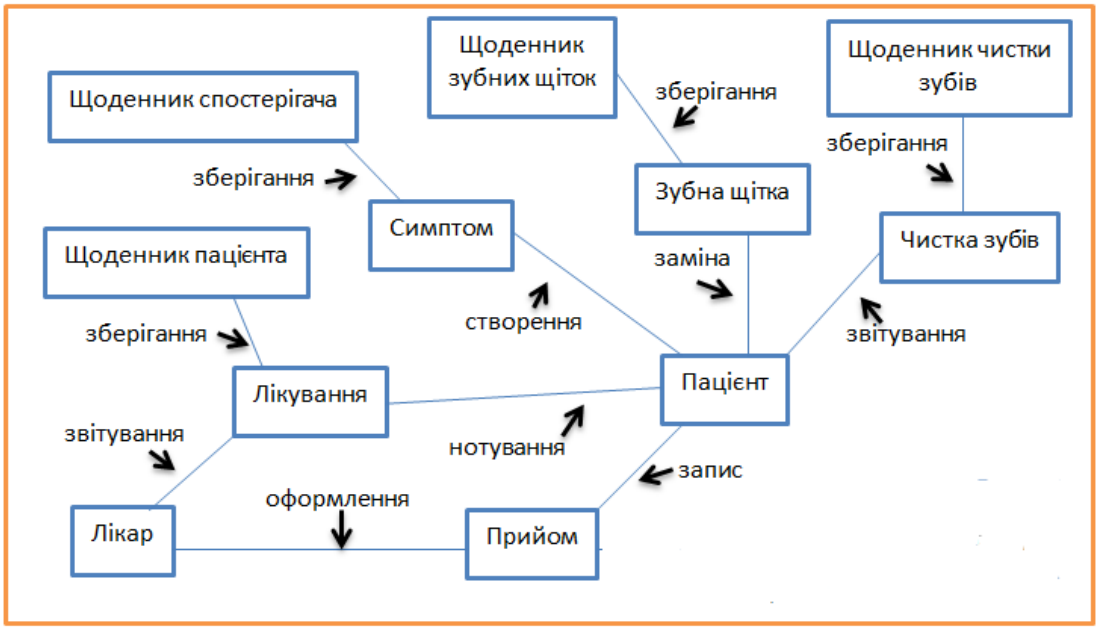

Рис. 2.9. Концептуальна модель

Перелік інформації, що вводиться користувачами:

– персональні дані користувача (ПІБ, номер телефону, адреса електронної пошти, робоча адреса, спеціалізація лікаря);

– дані про лікування (назва, дата початку, дата кінця, перелік зроблених записів);

– дані про записи лікувального процесу (дані про лікаря, дата запису, оцінка прийому, рецепт від лікаря, адреса проведення прийому, перелік процедур);

– дані про проведенні процедури (назва, коментар, прикріплений файл, перелік частин ротової порожнини);

– дані про симптоми (назва, рівень болю, дата початку);

– дані про зубні щітки (дата початку та припинення використання, рівень жорсткості, колір);

– дані про чистку зубів (чистка з ранку, після прийому їжі, перед сном, загальна кількість разів, дата).

2.4.2.2. Логічна модель бази даних

На основі концептуальної схеми було спроектовано логічну модель бази даних і приведення її до необхідного рівня нормалізації. Варто зауважити, що тип розробленої логічної моделі – реляційна.

Тож, було виділено наступні сутності:

– користувач;
– лікар;
– доступ;
– щоденник;
– симптом;
– запис чистки;
– зубна щітка;
– лікування;
– запис;
– процедура;
– файл;
– системна процедура;
– частина ротової порожнини;
– область застосування.

В порівнянні з концептуальною схемою, де присутні елементи різних типів щоденників та журналу, так як вони мають схожу структуру, задля оптимізації бази даних, в логічній моделі було прийнято рішення про об’єднаних даних елементів в одну сутність, під назвою «Щоденник».

Також було визначено взаємозв’язки між сутностями. Після нормалізації всі зв’язки набули типу один до багатьох:

– користувач – лікар;

– користувач – доступ;

– користувач – щоденник;

– доступ – щоденник;

– щоденник – симптом;

– щоденник – запис чистки;

– щоденник – лікування;

– щоденник – зубна щітка;

– лікування – запис;

– запис – лікар;

– запис – процедура;

– процедура – системна процедура;

– процедура – файл;

– процедура – область застосування;

– область застосування – частина ротової порожнини.

В кожній сутності було створено атрибути і первинні ключі, які можна переглянути в таблиці 2.2 або на рис.2.9.

Таблиця 2.2

Атрибути і первинні ключі сутностей

Сутність	Первинний ключ	Атрибути
Користувач	Унікальний ключ користувача	Унікальний ключ користувача
		Ім’я користувача
		Адреса електронної пошти
		Пароль
		Тип користувача
		Дата народження
		Стать
		Робоча адреса
		Спеціалізація
		Телефон
Лікар	Унікальний ключ лікаря	Унікальний ключ лікаря
		Номер телефону
		Робоча адреса
		Ім’я лікаря

Продовж. табл. 2.2

Сутність	Первинний ключ	Атрибути
Лікар	Унікальний ключ лікаря	Спеціалізація
		Унікальний ключ користувача
Доступ	Унікальний ключ доступу	Унікальний ключ доступу
		Унікальний ключ користувача
		Унікальний ключ щоденника
Щоденник	Унікальний ключ щоденника	Унікальний ключ щоденника
		Унікальний ключ користувача
		Тип щоденника
Симптом	Унікальний ключ симптому	Унікальний ключ симптому
		Назва симптому
		Рівень болю
		Дата створення запису про симптом
		Дата початку симптому
		Унікальний ключ щоденника
Запис чистки	Унікальний ключ запису чистки	Унікальний ключ запису чистки
		Дата чистки
		Кількість разів
		Чистка зранку
		Чистка після прийому їжі
		Чистка перед сном
		Унікальний ключ щоденника
Зубна щітка	Унікальний ключ зубної щітки	Унікальний ключ зубної щітки
		Унікальний ключ щоденника
		Жорсткість
		Колір
		Дата початку користування
		Дата припинення користування
		Статус
Лікування	Унікальний ключ лікування	Унікальний ключ лікування
		Назва
		Дата початку
		Дата кінця
		Унікальний ключ щоденника

Закінч. табл. 2.2

Сутність	Первинний ключ	Атрибути
Запис	Унікальний ключ запису	Унікальний ключ запису
		Унікальний ключ лікаря
		Дата запису
		Унікальний ключ лікування
		Оцінка
		Адреса
		Рецепт від лікаря
Процедура	Унікальний ключ процедури	Унікальний ключ процедури
		Унікальний ключ системної процедури
		Унікальний ключ запису
		Коментар
Файл	Унікальний ключ файлу	Унікальний ключ файлу
		Ім’я файлу
		Адреса
		Дата
		Унікальний ключ процедури
Системна процедура	Унікальний ключ системної процедури	Унікальний ключ системної процедури
		Назва процедури
		Тип процедури
Частина ротової порожнини	Унікальний ключ частини ротової порожнини	Унікальний ключ частини ротової порожнини
		Назва частини ротової порожнини
Область застосування	Унікальний ключ області застосування	Унікальний ключ області застосування
		Унікальний ключ процедури
		Унікальний ключ частини ротової порожнини

Проаналізувавши дану таблицю, можна звернути увагу на те, що в деяких сутностях повторюються певний набір атрибутів, а саме в таких парах сутностей: «Лікар» і «Користувач». Це зроблено не випадково. І взагалі виділення сутності «Лікаря» від «Користувача» теж має на це свою причину. Вона полягає у тому, що в розробленій системі передбачено випадок відсутності реєстрації у системі того чи іншого лікаря. При заповнені даних

це може призвести до неточності. Тому було прийнято рішення про забезпечення можливості заповнення даних без синхронізації з зареєстрованими користувачами, що і призвело до відокремлення цієї сутності.

Також може бути незрозуміло навіщо сутності «Запис» такий атрибут як «Адреса». Тут мається на увазі адреса клініки чи стоматологічного кабінету, в якій відбувся прийом, про який відповідно робиться запис.

Атрибут «Адреса» також наявний в сутності «Файл». Він призначений задля заповнення даних про адреси проведення знімків та аналізів, для яких в свою чергу і було створено сутність «Файл».

Щодо сутностей «Процедура» та «Системна процедура», вони мають різне призначення. Системні процедури це перелік існуючих стоматологічних процедур, які заздалегідь були вже внесені до системи. Вони призначені для того, щоб при створені і опису проведеної процедури (сутність «Процедура») користувач мав змогу обрати з наявного переліку, що пришвидшує і полегшує шлях самого заповнення даних. Структура та зміст даних про системні процедури було розроблено за прототипом раніше згадуваної системи Medics. Перелік внесених процедур представлений в Додатку В.

Також заздалегідь вносяться у систему дані про частини ротової порожнини. Перелік цих даних представлені також в Додатку В.

Описана логічна модель бази даних зображена на рис.2.10.

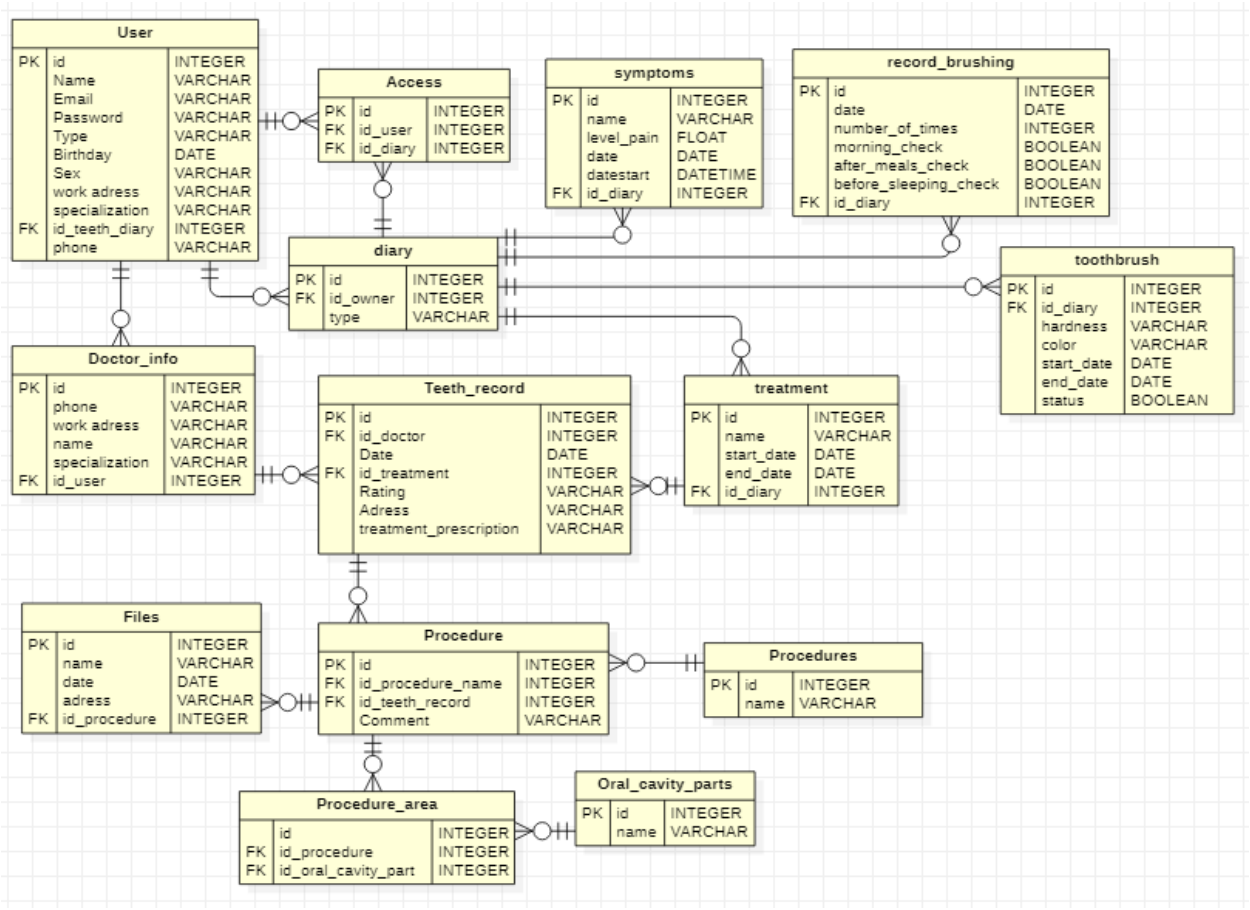

Рис. 2.10. Логічна модель бази даних

2.4.2.3. Фізична модель бази даних

Фізична модель бази даних була розроблена за допомогою СУБД MySQL. Проект таблиць для фізичної моделі з урахуванням типів і розмірів даних наведено в Додатку Г. ER-діаграма представлена на рис. 2.11.

Задля покращення і пришвидшення процесу відображення даних у застосунку, було розроблено ряд представлень, до яких увійшли:

– дані про щоденники;

– дані про лікування;

– дані про записи;

– дані про процедури.

Кожне з представлень об’єднує в собі дані з декількох таблиць. Так, представлення «Дані про лікування» включає дані з наступних сутностей: «Лікування», «Запис» та «Щоденник». В свою чергу «Дані про щоденники»

об’єднують дані з: «Користувач», «Щоденник» та «Доступ». «Дані про процедури» включають сутності: «Область застосування», «Частина ротової порожнини», «Процедура», «Системна процедура», «Запис», «Лікування», «Щоденник» та «Файл». І нарешті, до представлення «Дані про запис» входить дані з таблиць: «Процедура», «Системна процедура», «Щоденник», «Лікар», «Лікування» та «Запис».

Проект таблиць представлень наведено нижче (табл. 2.3-2.6).

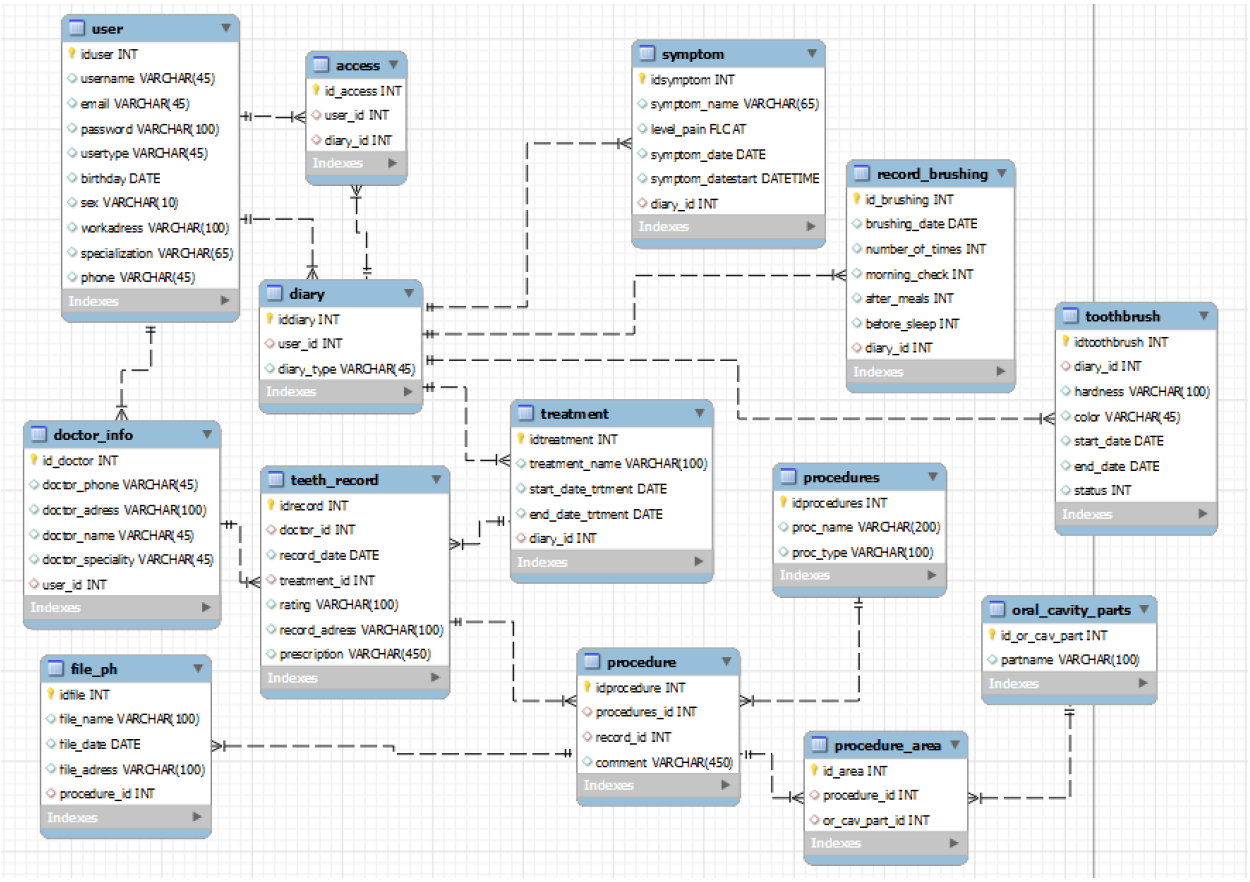

Рис. 2.11. Фізична модель бази даних

Таблиця 2.3

Таблиця представлення «Дані про лікування»

all_treatments (Дані про лікування)				1
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	iddiary	Унікальний ключ щоденника	int	

Продовж. табл. 2.3

№ п/п	Найменування стовбців	Примітка	Тип	Розмір
2	idtreatment	Унікальний ключ лікування	int	
3	treatment_name	Назва лікування	varchar	100
4	start_date_trtment	Дата початку лікування	date	
5	end_date_trtment	Дата закінчення лікування	date	
6	date_difference	Тривалість	int	
7	number_of_record	Кількість записів	bigint	
8	user_id	Унікальний ключ користувача	int	

Таблиця 2.4

Таблиця представлення «Дані про щоденники»

diary_data (Дані про щоденники)				3
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	iddiary	Унікальний ключ щоденника	int	
2	owner	Ім’я власника щоденника	varchar	45
3	owner_id	Унікальний ключ власника	int	
4	with_access	Ім’я користувача, у якого є доступ до щоденника	varchar	45
5	access_id	Унікальний ключ доступу	int	
6	owner_phone	Телефон власника	varchar	45

Таблиця 2.5

Таблиця представлення «Дані про процедури»

procedure_data (Дані про процедури)				4
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idrecord	Унікальний ключ запису	int	
2	idprocedure	Унікальний ключ процедури	int	

Продовж. табл.2.5

№ п/п	Найменування стовбців	Примітка	Тип	Розмір
3	partname	Назва частини ротової порожнини	varchar	100
4	proc_name	Назва процедури	varchar	200
5	file_name	Назва файлу	varchar	100
6	user_id	Унікальний ключ користувача	int	
7	id_or_cav_part	Унікальний ключ частини ротової порожнини	int	
8	idtreatment	Унікальний ключ лікування	int	

Таблиця 2.6

Таблиця представлення «Дані про записи»

record_data (Дані про записи)				5
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idtreatment	Унікальний ключ лікування	int	
2	idrecord	Унікальний ключ запису	int	
3	record_date	Дата запису	date	
4	record_adress	Адреса запису	varchar	100
5	rating	Оцінка	varchar	100
6	prescription	Рецепт	varchar	450
7	doctor_name	Ім’я лікаря	varchar	45
8	doctor_phone	Номер телефону лікаря	varchar	45
9	number_of_procedure	Кількість процедур	bigint	
10	proc_name	Назва процедури	varchar	200
11	user_id	Унікальний ключ користувача	int	

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Основні вхідні дані застосунку надходять через клієнтську частину, при введенні даних користувачем. Вони організовуються за допомогою розроблених форм в інтерфейсі, які мінімізують помилковість ведення даних за рахунок забезпечення, в багатьох випадках, вибору з наявних даних. Ці дані теж є вхідними, вони зберігаються і надходять вже з самої бази даних. До них входять перелік процедур, частин ротової порожнини і лікарів. Варто зазначити, що вони заздалегідь були введенні в базу даних і змінам не піддаються, принаймні через інтерфейс. Але це не стосується даних про лікарів. Ці дані створюються безпосередньо користувачем, чи то при реєстрації чи то при створення медичного запису.

Загалом вхідні дані поділяються на декілька типів: текстові, числові, дата і файли. Текстові дані представлені на українській мові. Деякі з них мають обмеження в розмірі, в залежності від призначення і виду. Числові дані загалом представляються цілими числами. Дата має формат РРРР-ММ-ДД. Щодо файлів, збереження і обробки яких організовано за допомогою локального сховища на стороні клієнта. В базу даних потрапляє тільки назва файлу і додатково веденні про нього відомості, такі як дата і місце створення. Дозволені формати файлів – всі типи зображень.

В застосунку часто використовується локальне сховище браузера задля маніпулюванням вхідними та вихідними даними, включаючи дані cookie.

Вихідні дані надходять до користувача в вже опрацьованому вигляді. Основним прикладом такого опрацювання є виведення даних про відлік часу від важливих параметрів догляду та здоров’я ротової порожнини,які включають:

– кількість зубів;

– час з останнього відвідування стоматолога;

– час з останньої професійної чистки зубів;

– час з останньої заміни зубної щітки;

– сьогоднішня кількість разів чистки зубів.

Також однин із головних вихідних даних є графічне представлення ротової порожнини, яке уособлює її наявний стан. Це означає, що відображення реагує на вхідні дані, і в залежності від цього змінює свій зовнішній вигляд.

Всі інші вихідні дані представлені у вигляді карток, до них відносять записи про: лікування, симптоми, чистки зубів, замін зубних щіток і журнали пацієнтів.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для розробки і тестування застосунку було використано технічний засіб з наступною конфігурацією:

– ЦП: AMD Ryzen 5 5600Uwith Radeon Graphics 2.30GHz;

– ОЗП: 16Гб;

– диск SSD: SAMSUNG MZALQ512HBLU-00BL2;

– відео карта: AMD Radeon(TM) Graphics 9Гб;

– операційна система: 64-розрядна Windows 10;

– дисплей:

  – частота оновлення: 90Гц;

  – глибина кольору: 10біт;

  – формат кольору: RGB;

  – роздільна здатність: 2880x1800;

  – співвідношення сторін: 16:10;

– стандартна клавіатура;

– звичайна миша.

Розроблений додаток може працювати на технічних засобах і з іншими конфігураціями, головне щоб параметри не були нижче за попередньо вказаних у пункті 1.53 «Вимоги до складу та параметрів технічних засобів».

2.6.2. Використані програмні засоби

Під час розробки проекту даної кваліфікаційної роботи було використано ряд допоміжних програмних засобів, а саме:

– Visual Studio Code;
– MySQL Workbench;
– Figma;
– AdobeIllustrator;
– StarUML;
– EdrawSoft;
– Github.

Visual Studio Code[29] – це редактор вихідного коду розроблений Microsoft з вбудованою підтримкою JavaScript, Node.js, TypeScript і має велику кількість розширень для інших мов і середовищ виконання. Під час розробки був використаний для написання коду всієї програми.

MySQL Workbench[30] – інструмент управління базами даними, яка була використана для створення, заповнення і тестування роботи бази даних проекту.

Figma – веб-орієнтований додаток для розробки і дизайну інтерфейсів користувачів та окремих його компонентів. За допомогою нього було розроблено інтерфейс застосунку, згідно якого в подальшому велася розробка.

AdobeIllustrator[31] – професійний редактор для роботи з векторною графікою, а саме створення логотипів, дизайнів, вебграфіки та інше. Розробник якого компанія Adobe Systems. В кваліфікаційній роботі був

використаний для створення графічної моделі ротової порожнини, логотипу сайту та інших ілюстрацій.

StarUML [32]– це застосунок, який призначений для розробки програмного забезпечення та системного моделювання за допомогою діаграм і мови UML. Розробник якого є MKLabs Co. Ltd. В даній роботі був використаний задля створення ряд діаграм, а саме: Use-case діаграми, діаграм діяльності і логічну модель бази даних.

EdrawSoft[33] – застосунок для створення блок-схем, діаграм зв'язків, мережевих діаграм, UML-діаграм, планів поверхів та багато чого іншого. Був використаний задля проектування і опису архітектури застосунку за допомогою діаграм, що відображають структуру шаблону проектування MVC, стеку MERN, і логічну структуру веб-сайту.

Github[34] – система керування та контролю версіями розроблених програмних продуктів. Був використаний задля зберігання і відстеження написання коду застосунку під час розробки.

2.6.3. Виклик та завантаження програми

Задля успішного виклику, завантаження і розгортання програми необхідно виконати наступні дії:

– забезпечити наявність, доступ і працездатність необхідного технічного обладнання для серверів застосунку і бази даних;

– встановити на сервер застосунку Node.js;

– встановити на сервер бази даних – MySQL;

– налаштувати базу даних за допомогою створених скриптів;

– завантажити код і файли програми на сервер;

– увімкнути сервери і налаштувати підключення.

Варто зазначити, що розробка і тестування застосунку велася на локальному сервері. В цьому випадку достатньо мати встановлений Node.js і СУБД MySQL. База даних налаштовується за параметрами, що представлені

на рис.2.12, після чого необхідно запустити написаний скрипт. І за допомогою Node.js і команди виклику «npm start» запускається клієнтська і серверна частини застосунку.

```
export const db = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "GamabUnta6",
    database: "teethdiary"
})
```

Рис.2.12. Параметри підключення до бази даних

2.6.4. Опис інтерфейсу користувача

2.6.4.1. Розробка дизайну та графічних складових

Графічні складові компоненти, як вже раніше зазначалося, були створені за допомогою графічного редактора AdobeIllustrator і представляють собою векторну графіку. Це було зроблено для коректного відображення, що не залежить від масштабу зображень. Розроблені ілюстрації включають:

– логотип (рис.2.13);

– ротова порожнина (рис.2.14);

– набір зубів (рис.2.14);

– набір зубів після адгезивної реставрації (рис.2.15);

– набір зубів після екстирпації пульпи або хірургічної обробки кореневого каналу (рис.2.16);

– набір зубів покритими нальотом (рис.2.17);

– ілюстрація зубних щіток (рис.2.18).

Логотип представляє собою біле перо, яке нагадує посмішку. Цей образ уособлює назву застосунку – «Зубний щоденник».

Кожен зуб являє собою окрему ілюстрацію, це було зроблено задля того, щоб в подальшому була можливість відображати зміни в стані ротової порожнини в залежності від проведених процедур і області їх застосування.

Рис.2.13. Логотип

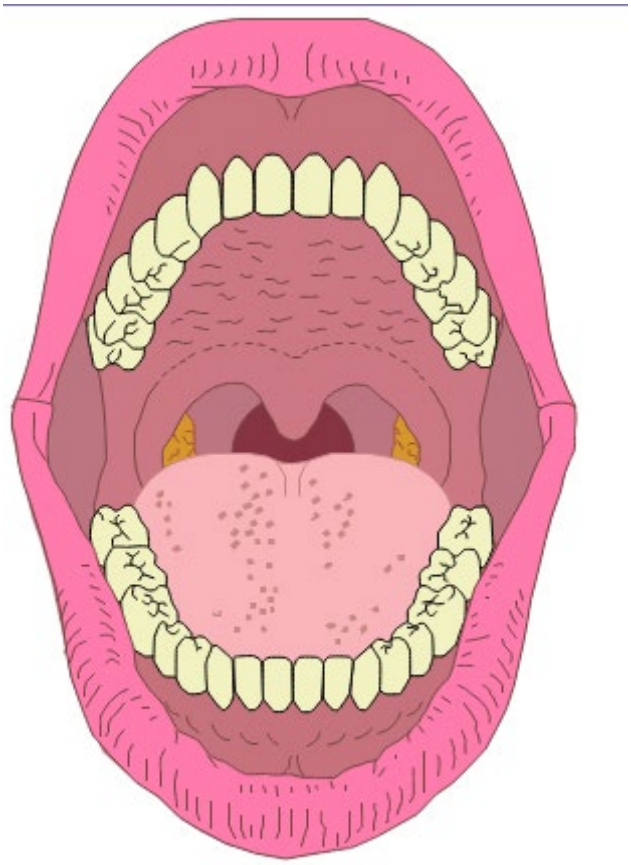

Рис.2.14. 2D модель ротової порожнини з зубами

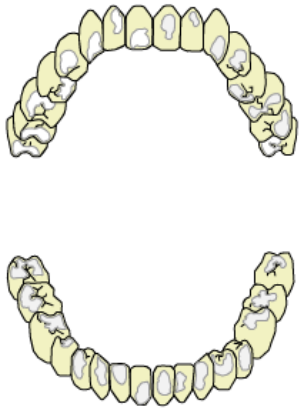

Рис.2.15. Ілюстрація зубів після адгезивної реставрації

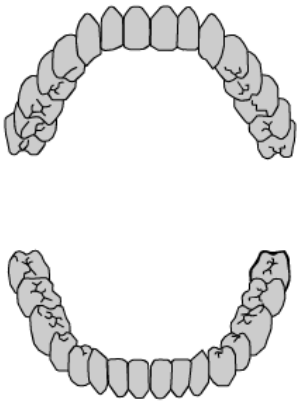

Рис.2.16. Ілюстрація зубів після екстирпації пульпи або хірургічної обробки кореневого каналу

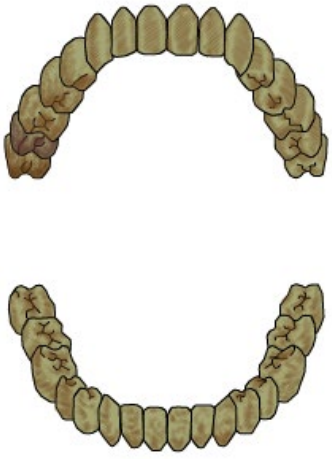

Рис.2.17. Ілюстрація зубів покритих нальотом

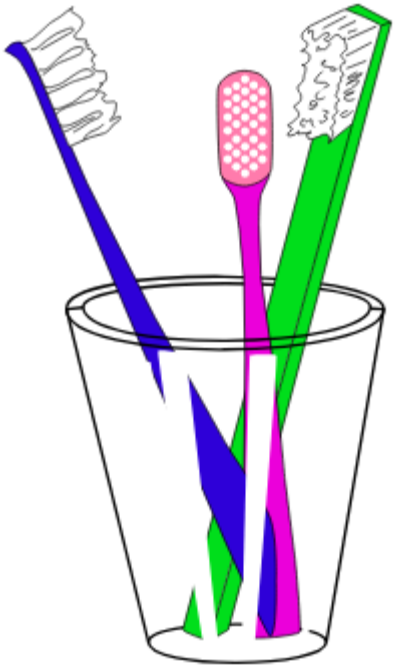

Рис.2.18. Ілюстрація зубних щіток

Дизайн макету сайту складається з 15 фреймів, які зображені на рисунках 2.19-2.22. Варто зазначити, що у процесі розробки були деякі незначні відхилення від дизайну, особливо це стосується сторінки модифікації даних про процедуру.

Загалом прототип дизайну має стандартну структуру і складається з трьох елементів: header, body і footer. Навігаційна панель знаходиться зверху в елементі header. Кольорова гама має не дуже широкий спектр, і складається з дев’яти кольорів. Основний колір застосунку – це блакитний. Фон має звичайний білий колір. Колір тексту заголовків та посилань в основному фіолетовий або білий. Основний шрифт застосунку – це OpenSans. Він використовується на всіх сторінках. виключенням є тільки головна сторінка, де було також застосовано шрифт Volkhov.

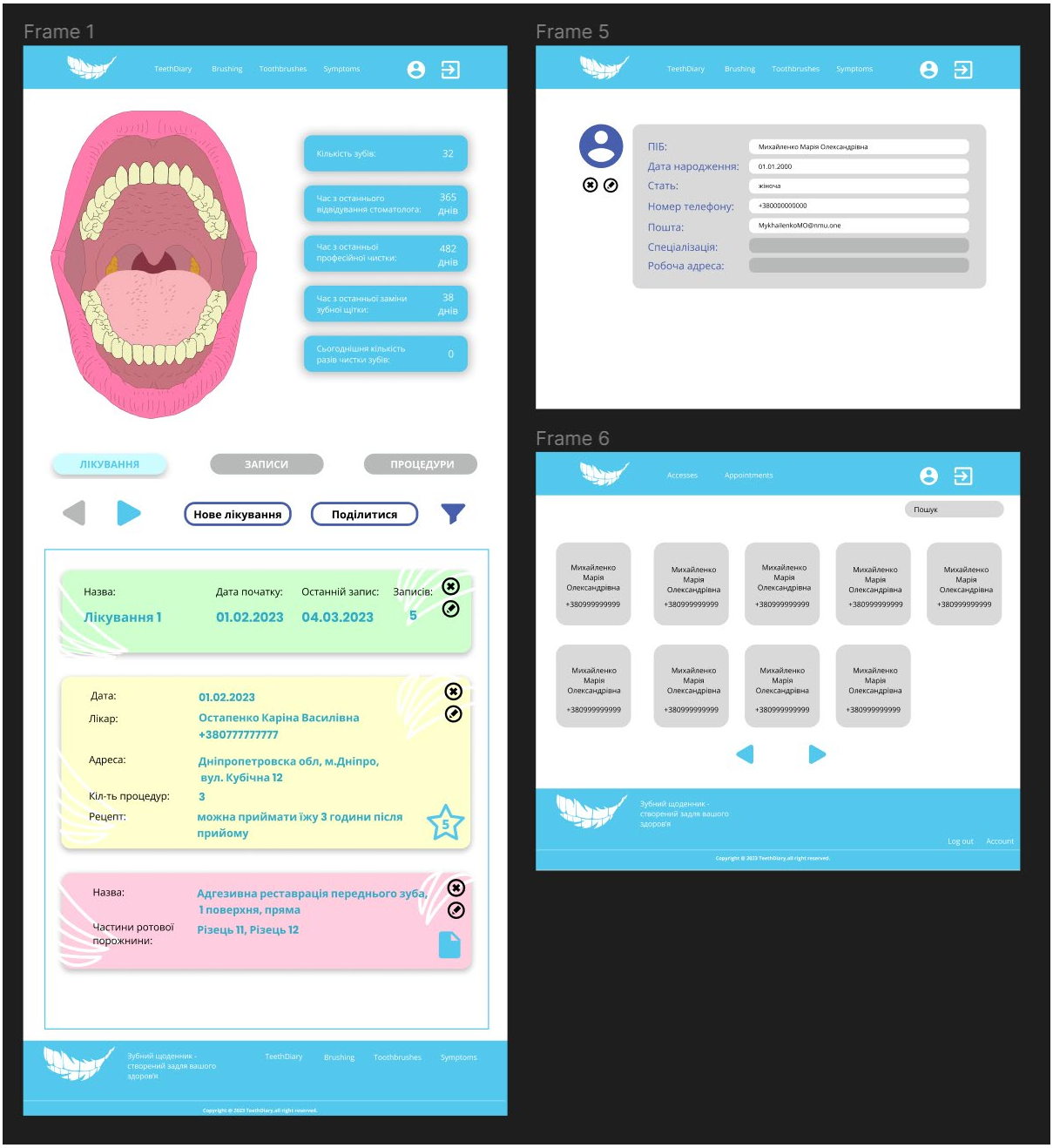

Рис.2.19. Дизайн сторінок «Щоденник пацієнта», «Акаунт» і «Журнал пацієнтів»

Окрім власних ілюстрацій, в застосунку було також використано ряд завантажених елементів. Це стосується саме іконок, які були використані для функціональних цілей, такі як позначення функцій видалення, модифікації, виходу із акаунту, проставлення рейтингу, перехід на іншу сторінку, відкриття файлу та панелі фільтрів.

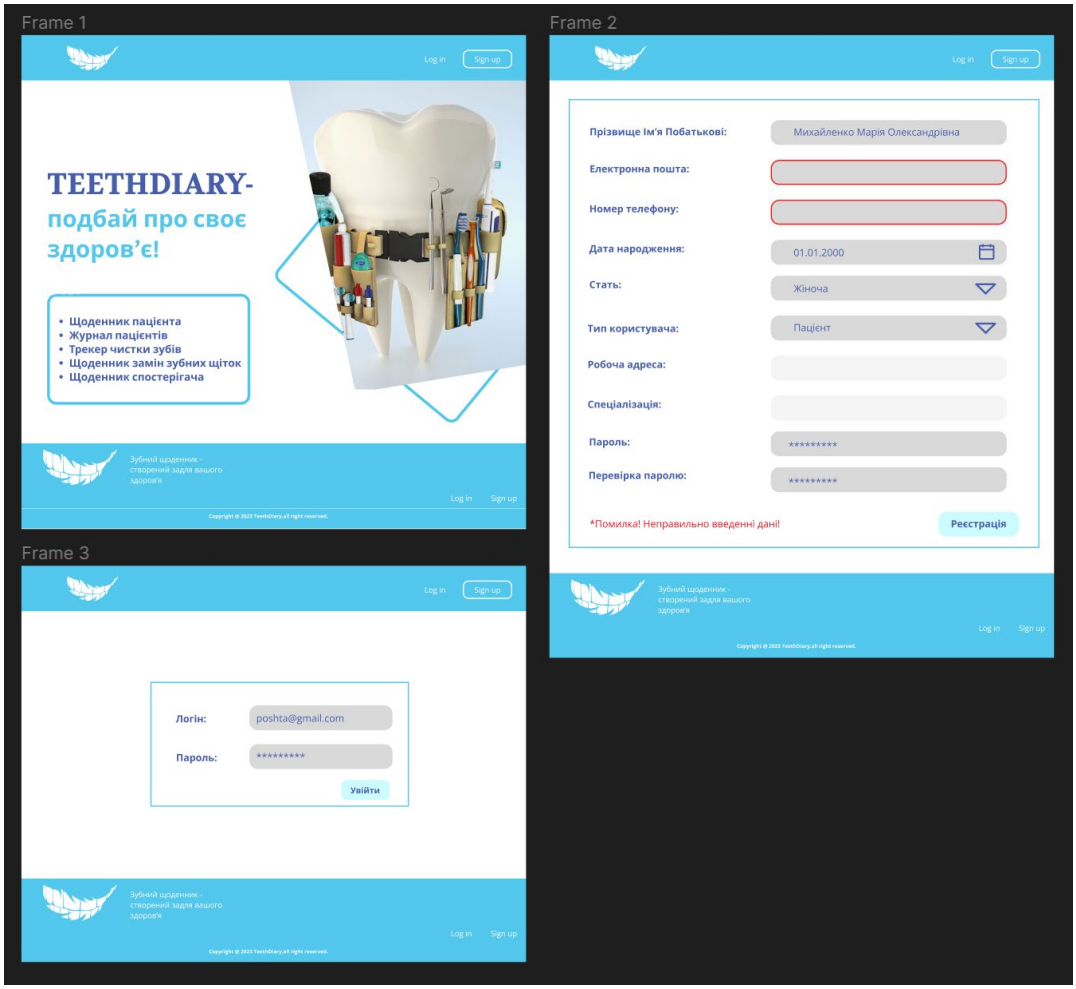

Рис.2.20. Дизайн головної сторінки, та сторінок реєстрації і авторизації

Рис.2.21. Дизайн додаткових елементів системи

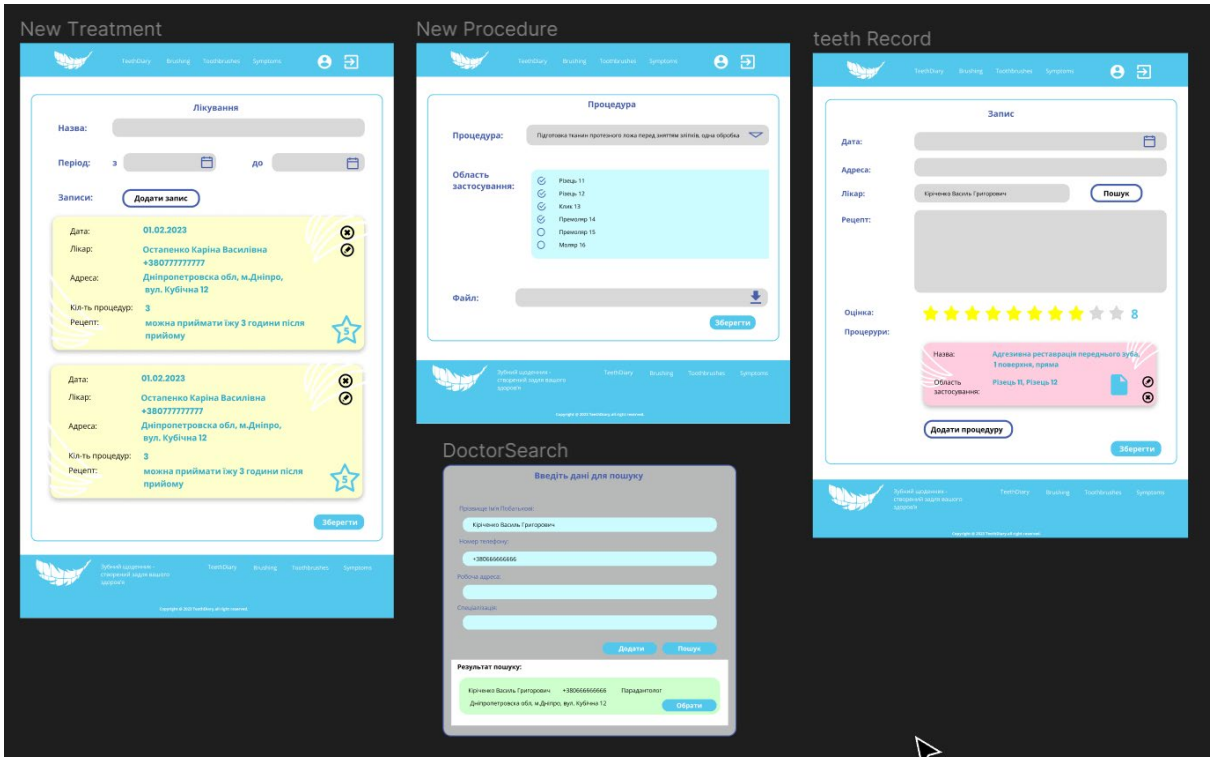

Рис. 2.22. Дизайн форм заповнення даних лікувального процесу

2.6.4.2. Клієнтська частина

Перша сторінка, яка відкривається при вході в застосунок – це головна сторінка. Вона містить у собі коротку інформацію про цілі і призначення застосунку (рис. 2.23). За допомогою навігаційного меню, є можливість перейти до сторінки авторизації (рис. 2.24) або реєстрації (рис.2.25).

Процес реєстрації відбувається на одній сторінці. Реєстраційна форма забезпечує перевірку дотримання вимог безпеки щодо паролю, відповідність паролів, коректність і унікальність введеної пошти. У разі невідповідностей в кінці форми виводиться повідомлення помилки і поле, яке її спричинило виділяється червоним кольором. До поки користувач не виправить помилки, зареєструватися не буде можливості. Також однією з особливостей реєстрації є поля «Робоча адреса» і «Спеціалізація», якщо користувач обирає тип «пацієнт» то ці поля стають недоступними для заповнення. Поля «Тип користувача» і «Стать» надають можливість тільки обирати дані з представлених.

У разі успішної реєстрації, відкривається сторінка авторизації. Авторизація здійснюється за допомогою паролю і логіна, за що виступає електронна пошта.

Рис.2.23. Головна сторінка застосунку

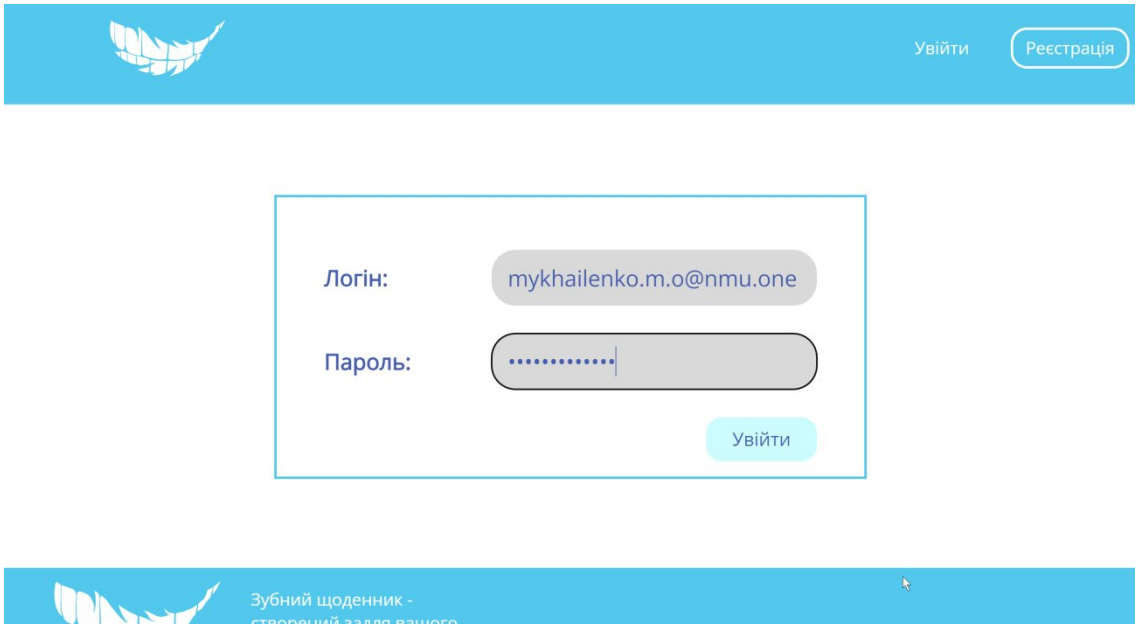

Рис.2.24. Сторінка авторизації

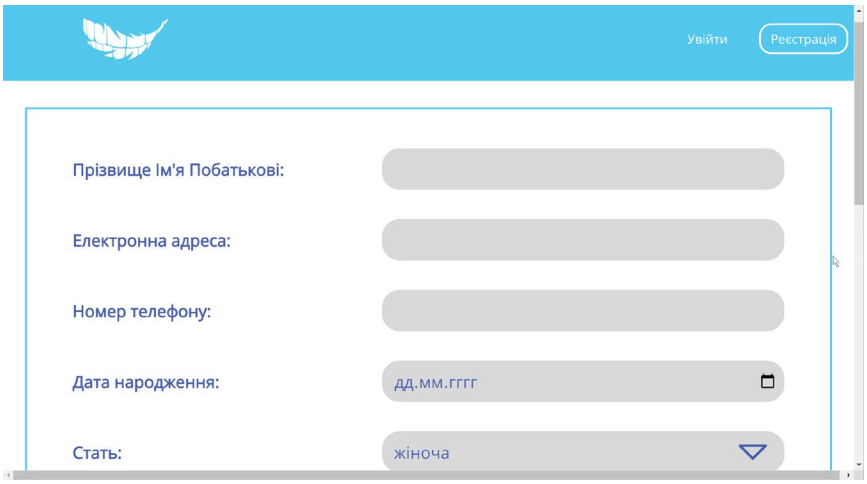

Рис.2.25. Сторінка реєстрації

Після авторизація користувач перенаправляється на сторінку власного акаунту (рис. 2.26), де він може переглянути свої дані. Якщо користувач зареєструвався, як пацієнт, йому стають доступні сторінки: «Щоденник пацієнта» (або «TeethDiary»), «Щоденник чистки зубів» (або «Brushing»), «Журнал замін зубних щіток» (або «Toothbrushes») і «Щоденник спостерігача» (або «Symptoms»).

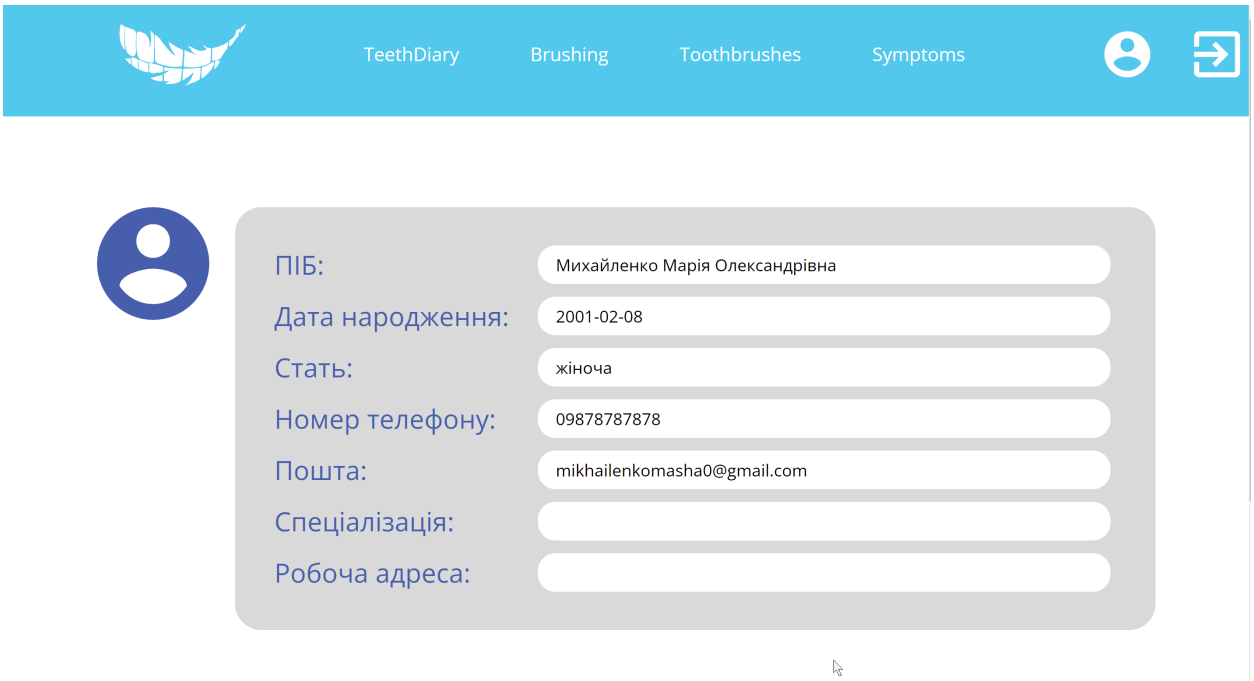

Рис. 2.26. Сторінка акаунту

На сторінці «Щоденник пацієнта» зображена 2D модель ротової порожнини (рис. 2.27), яка змінює свій зовнішній вигляд відносно проведених процедур і області їх застосування. Наразі в застосунку реалізовано відображення наступних процедур:

– видалення зуба;
– поява зубного нальоту;
– адгезивна реставрація;
– екстирпації пульпи або хірургічної обробки кореневого каналу.

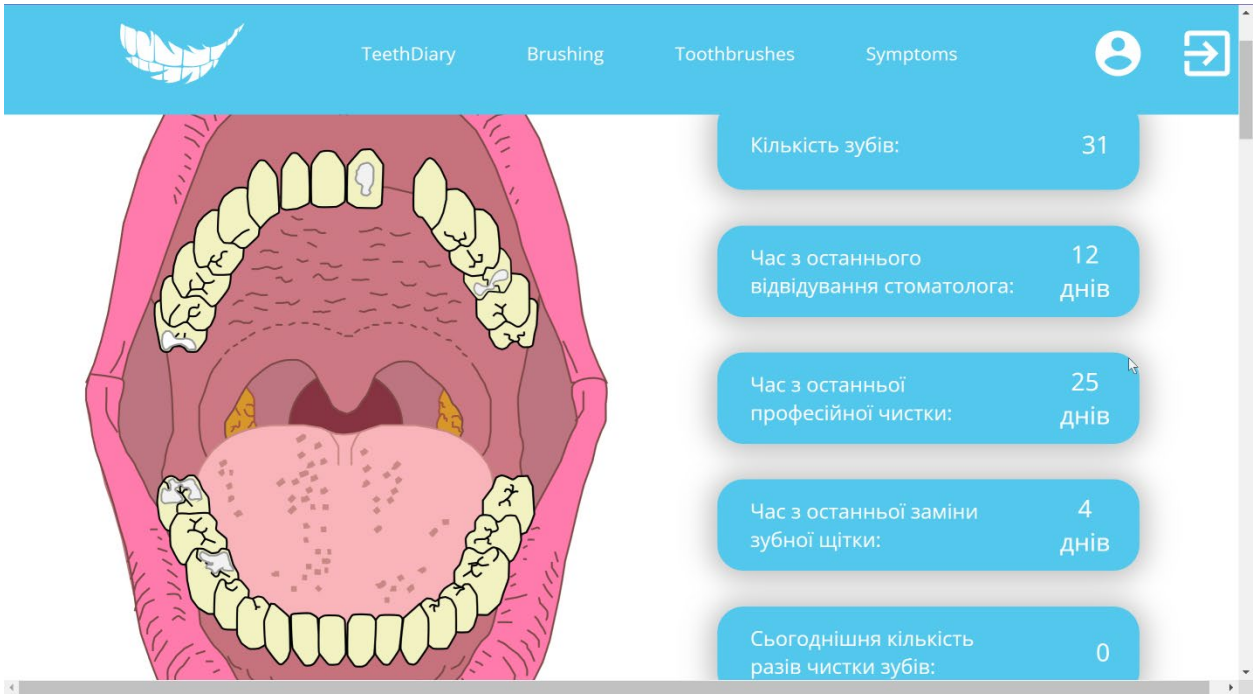

Рис.2.27. Сторінка «Щоденник пацієнта», відображення графічної моделі ротової порожнини і загальної інформації

З усіх представлених відображень, відокремлюється тільки реалізація демонстрації появи зубного нальоту (рис.2.28). Так він з'являється за умови, коли час з останньої професійної чистки зубів перевищує 365 днів.

Також на даній сторінці представлена загальна інформація, про поточний стан ротової порожнини і нюансів дотримання догляду і профілактики.

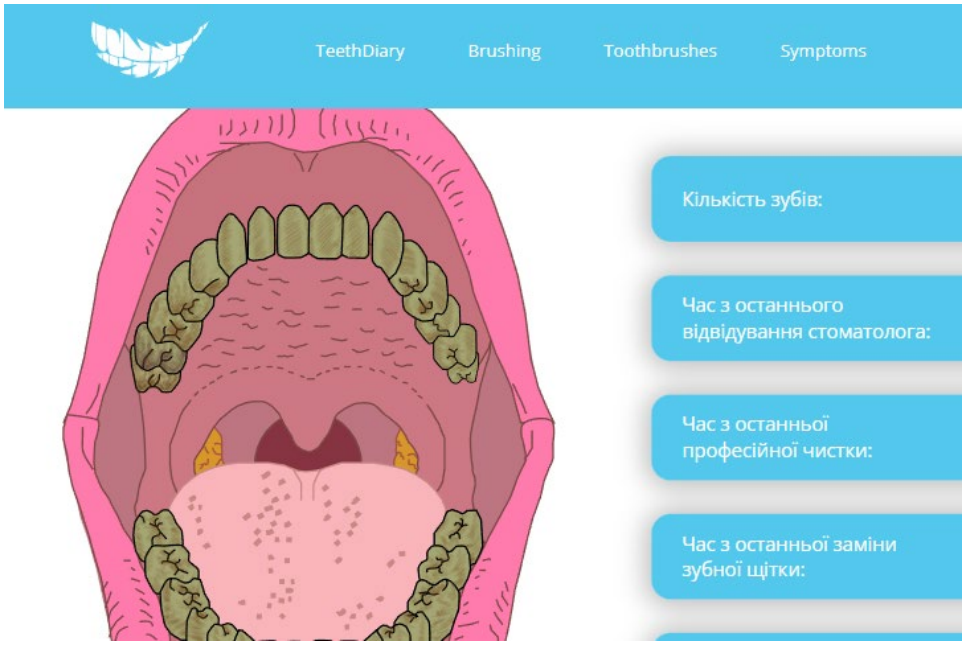

Рис.2.28. Відображення зубного нальоту на моделі ротової порожнини

Нижче виводиться список лікувань, записів і процедур. Для перегляду яких треба скористатися перемикачами, або кнопками стрілками. Також було реалізовано пошук за фільтром (рис.2.29). А саме є можливість знайти записи за датою і процедури за назвою і за областю застосування.

Рис.2.29. Демонстрація фільтру пошуку процедур і записів

Самі записи з різними типами відрізняються за кольором. Так лікування відображаються зеленим кольором (рис.2.30), записи – жовтим (рис.2.31), а процедури – рожевим(рис.2.32). Відображення записів включає основну інформацію. На картці «Лікування» показано: назву, дату початку, дата останнього запису (або кінця лікування) і загальна кількість записів. Картка «Запис» включає дані про: дату запису, ім'я лікаря, адресу, кількість процедур, рецепт від лікаря і оцінка. Картка «Процедура» відображає: назву, область застосування і наявність файлу.

Кожна картка може бути видалена та модифікована.

Процес створення записів відбувається у три етапи:

– створення лікування;

– додання запису;

– додання процедури.

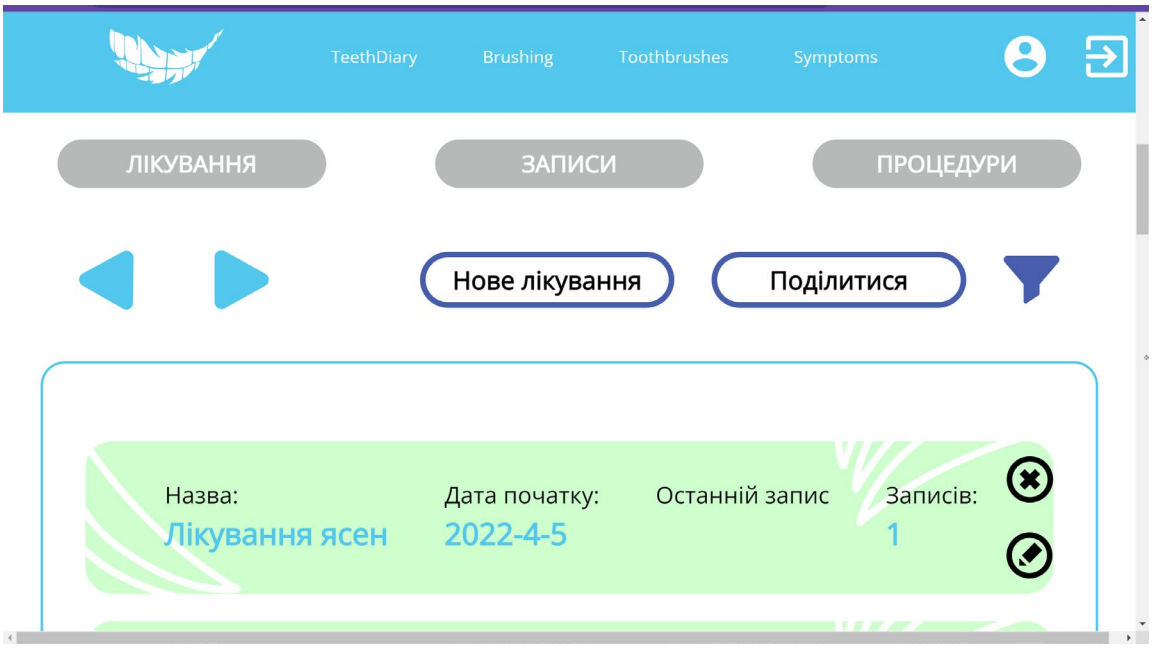

Рис.2.30. Відображення карток «Лікування»

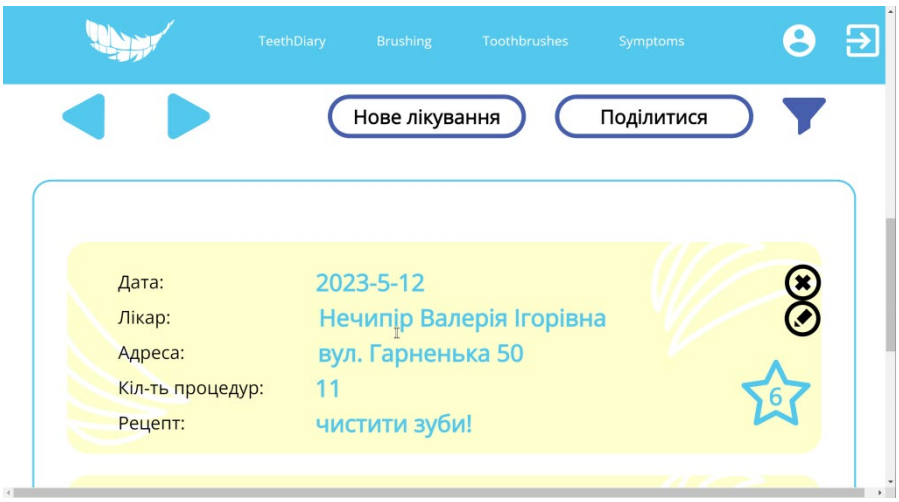

Рис.2.31. Відображення карток «Записів»

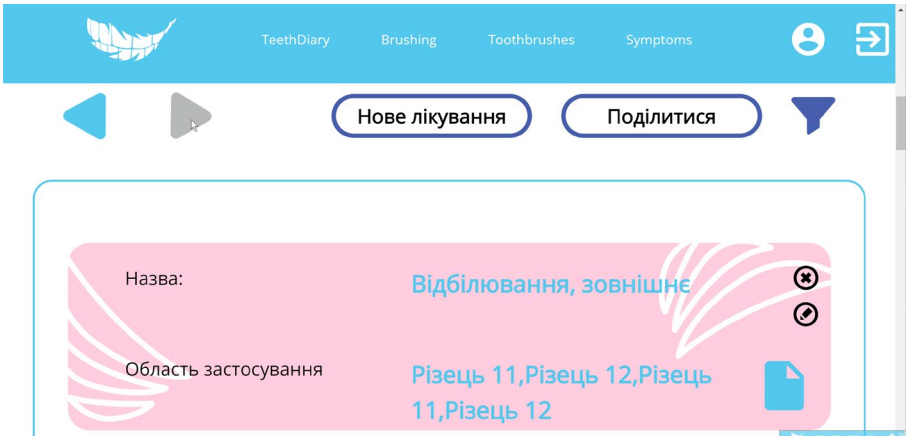

Рис.2.32. Відображення карток «Процедур»

Для кожного з цих етапів була розроблена своя сторінка. Так при створені нового лікування потрібно вести лише назву і дату початку і, при необхідності, закінчення (рис.2.33). Для додання запису необхідно ввести: дату, адресу, рецепт, поставити оцінку і знайти або створити лікаря (рис.2.34). Створення лікаря відбувається у випадку, якщо лікар не зареєстрований в системі. Для пошуку і створення реалізовано спеціальну форму. Після створення запису, надається можливість додати процедуру. Цей процес відбувається шляхом обрання з існуючих даних назву процедури і область застосування, наприклад зуби або ясна тощо (рис.2.35). Також є можливість додати файл зображення до процедури, і вказати дату його створення і адресу. Після чого у подальшому буде можливість перегляду

цього файлу, після натискання призначеної для цього іконки на картці «Процедури» (рис.2.36).

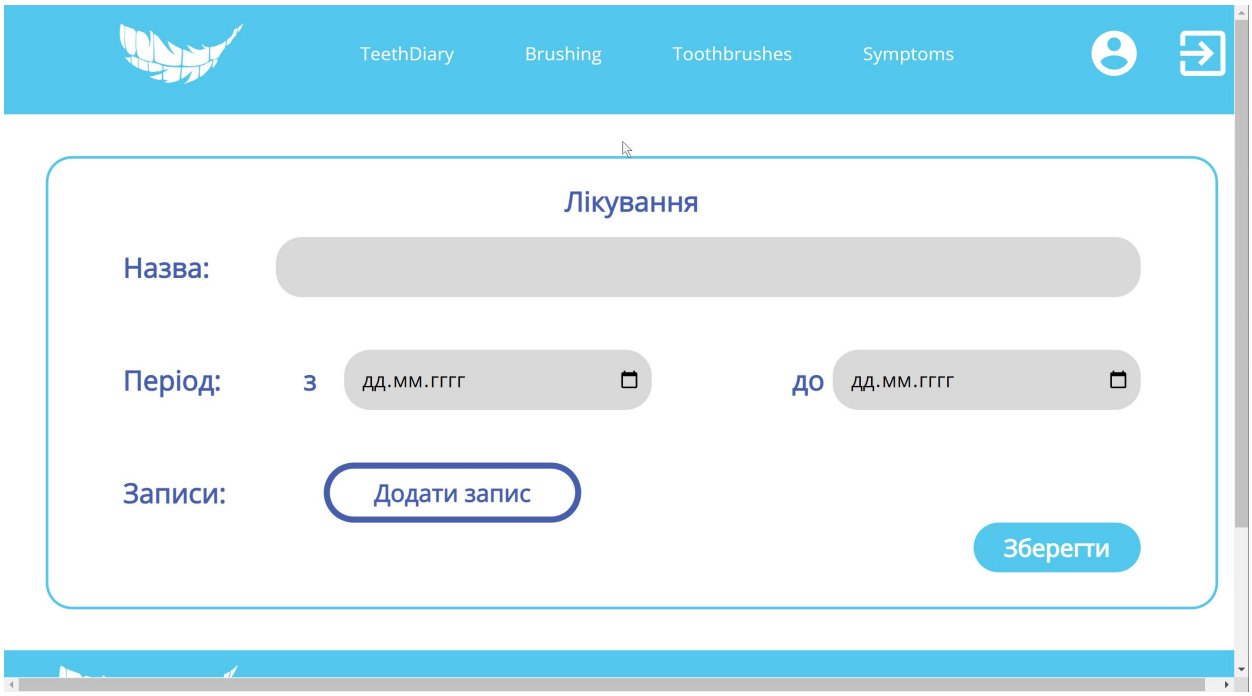

Рис.2.33. Форма створення лікування

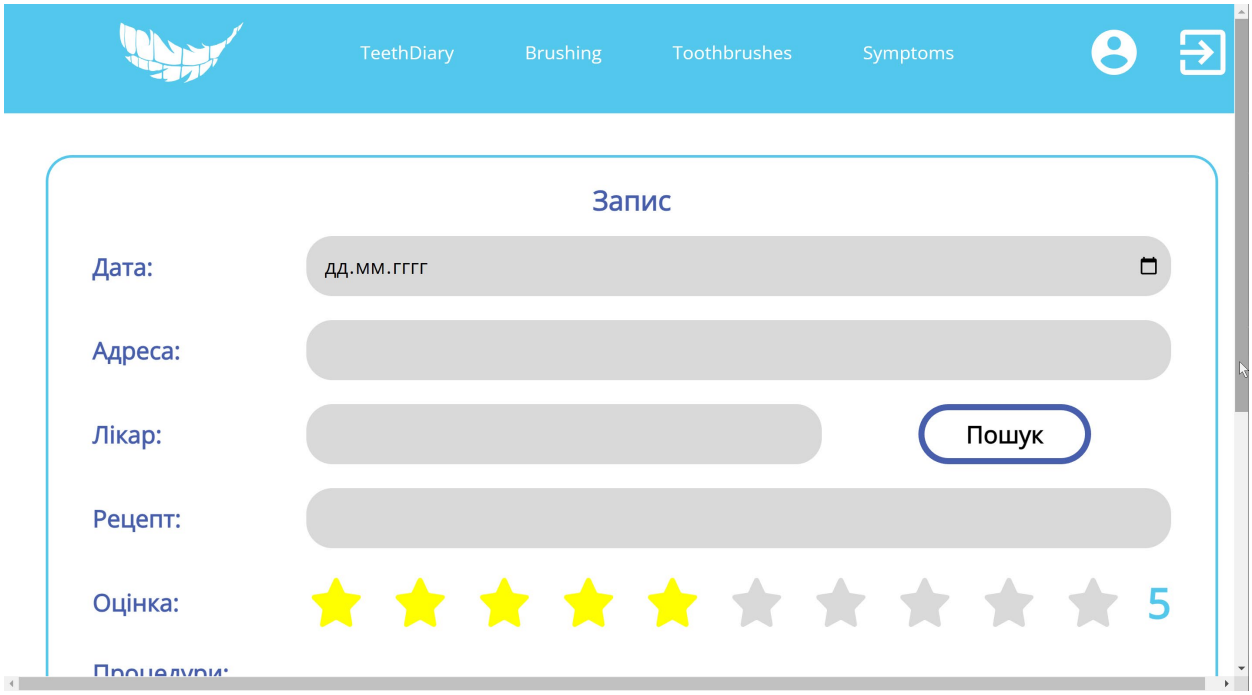

Рис.2.34. Форма створення запису

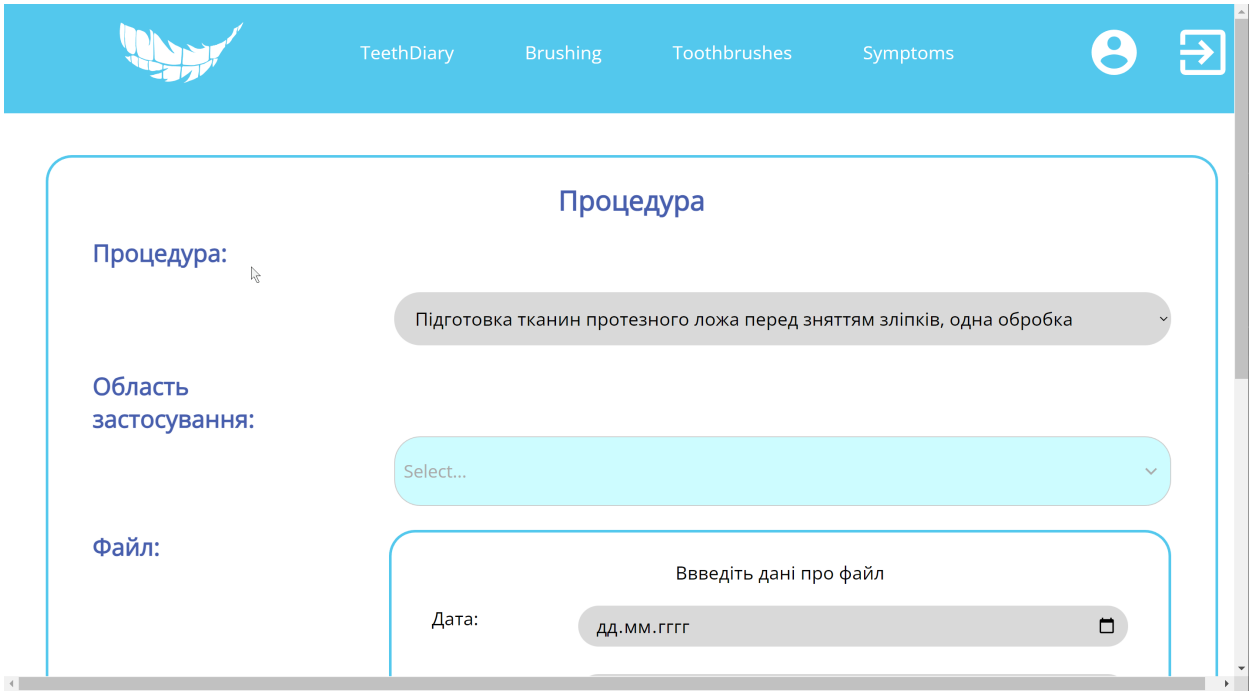

Рис.2.35. Форма створення процедури

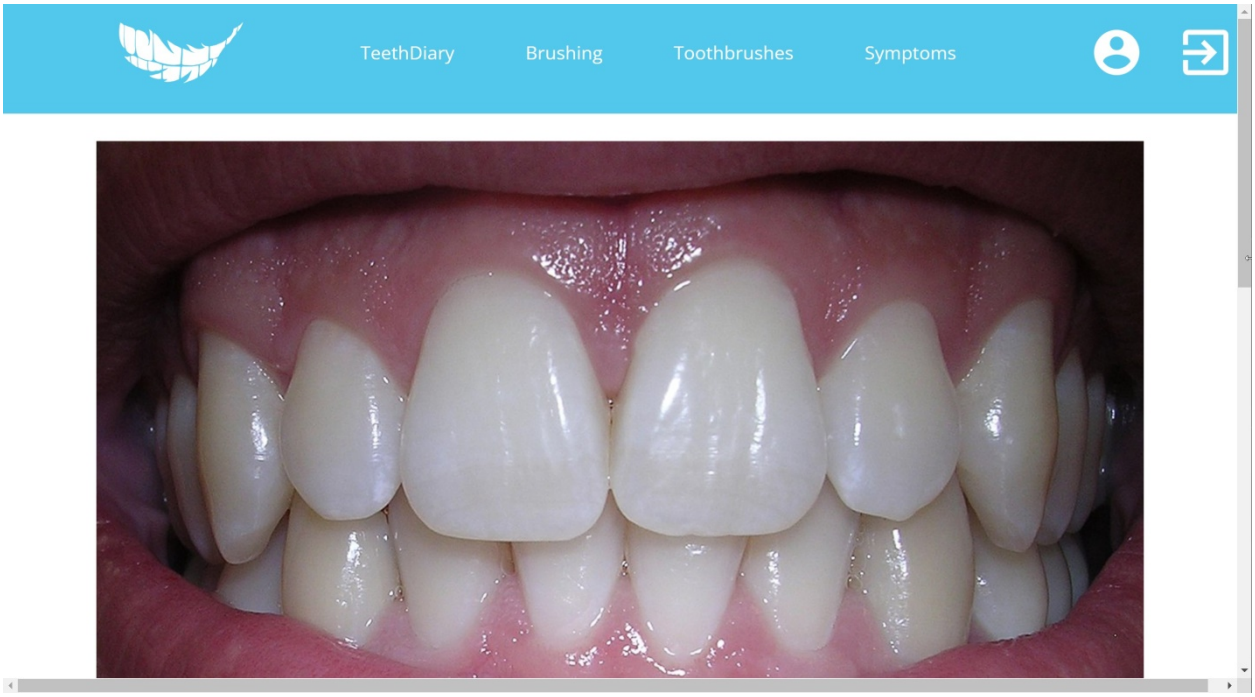

Рис. 2.36. Перегляд файлу, прикріпленого до процедури

Редагування записів відбувається за рахунок тих же самих форм.

Також однією з функціональних можливостей, що була реалізована на даній сторінці – є надання доступу лікарю. Цей процес відбувається за допомогою натискання кнопки «Поділитися», після чого відкривається

форма пошуку лікаря (рис.2.37) і після введення даних, і виведення результатів можна обрати лікаря, якому буде доступна в можливість вносити зміни в «Щоденник пацієнта».

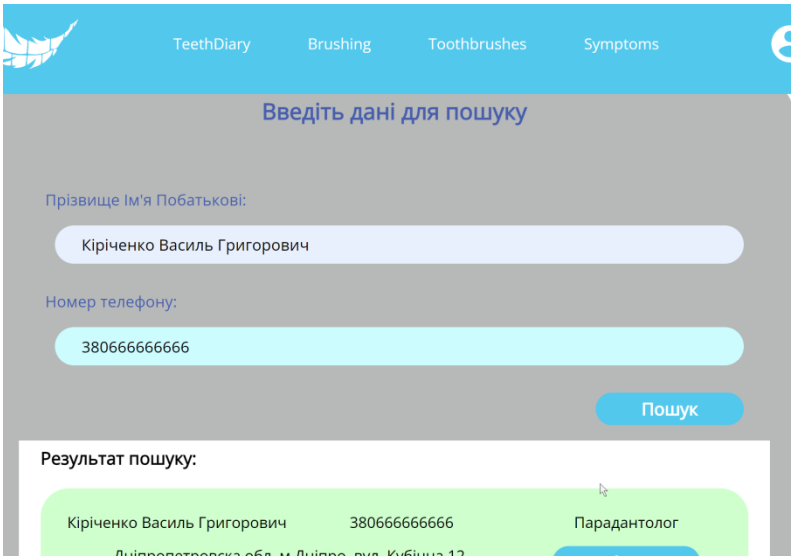

Рис.2.37 Форма пошуку лікаря

Сторінка «Щоденник чистки зубів» (рис.2.38) поділяється на дві частини: форму даних про сьогоднішні чистки і історію попередніх чисток. Форма складається з п’яти прапорців, які визначають кількість і період чистки зубів. Попередні записи можна видалити або модифікувати.

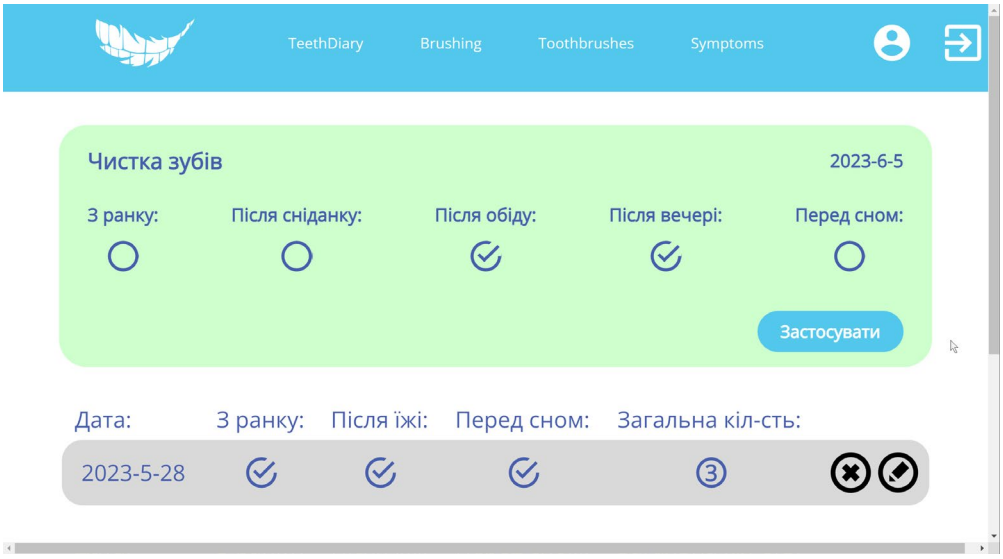

Рис.2.38. Сторінка «Щоденник чистки зубів»

На сторінці «Щоденника замін зубної щітки» виділено місця для виведення активних щіток (рис.2.39) і інформацію про них: колір, дата початку користування і ступінь жорсткості. Також на цій сторінці відображаються дані з попередніх зубних щіток, а саме: дата початку користування, тривалість користування, жорсткість і колір. Зубну щітку можна тільки видалити, створити і змінити статус на неактивну.

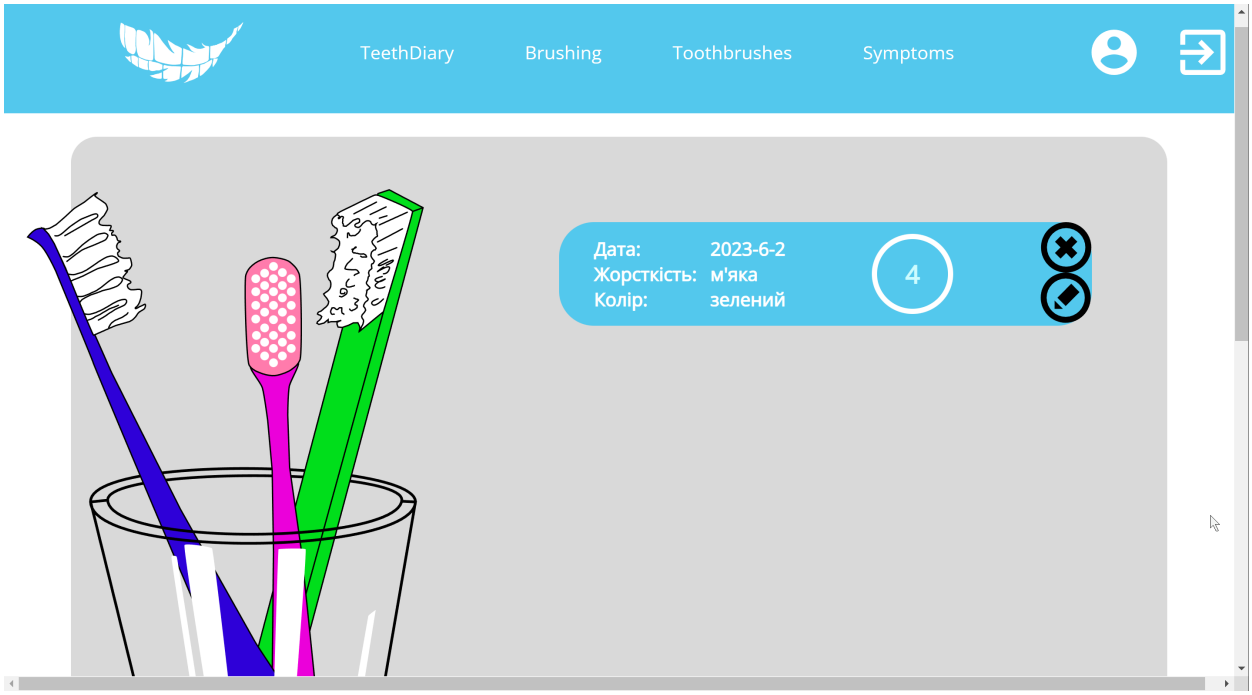

Рис.2.39. Сторінка «Щоденник зубних щіток», активні щітки

Сторінка «Щоденник симптомів» (Рис.2.40) складається лише з історії записів симптомів. При створені нового симптому надається можливість вказати: назву, дату, час і рівень болю. Ці дані також будуть відображатися на картках симптомів в подальшому. Також було реалізовано можливість видалення записів.

Якщо тип користувача – лікар, йому доступна сторінка «Журнал пацієнтів» (рис.2.41), де знаходиться всі щоденники до яких було надано лікарю доступ. Натиснувши на картку щоденника, користувач переходить на сторінку «Щоденника пацієнта».

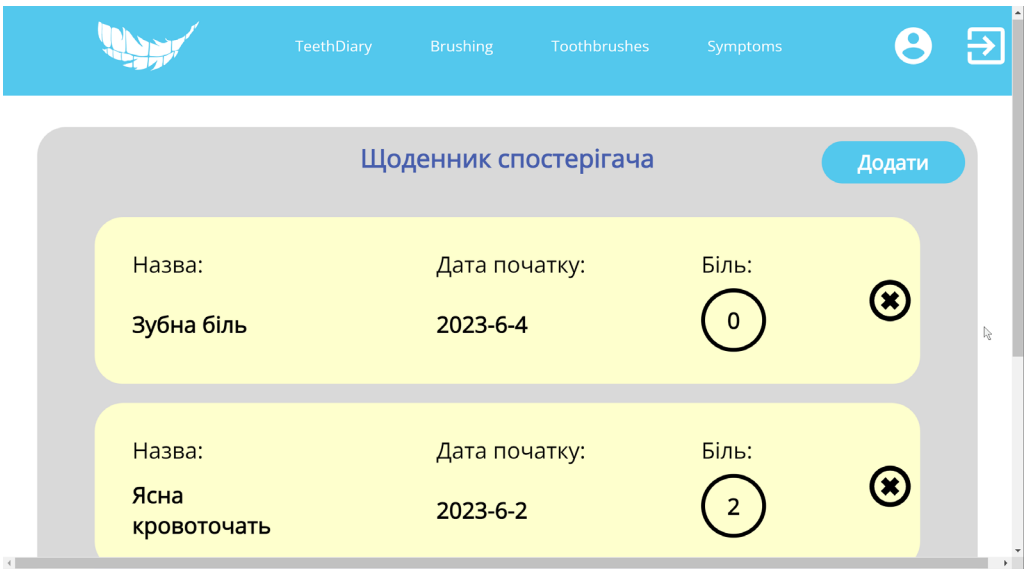

Рис.2.40. Сторінка «Щоденник спостерігача»

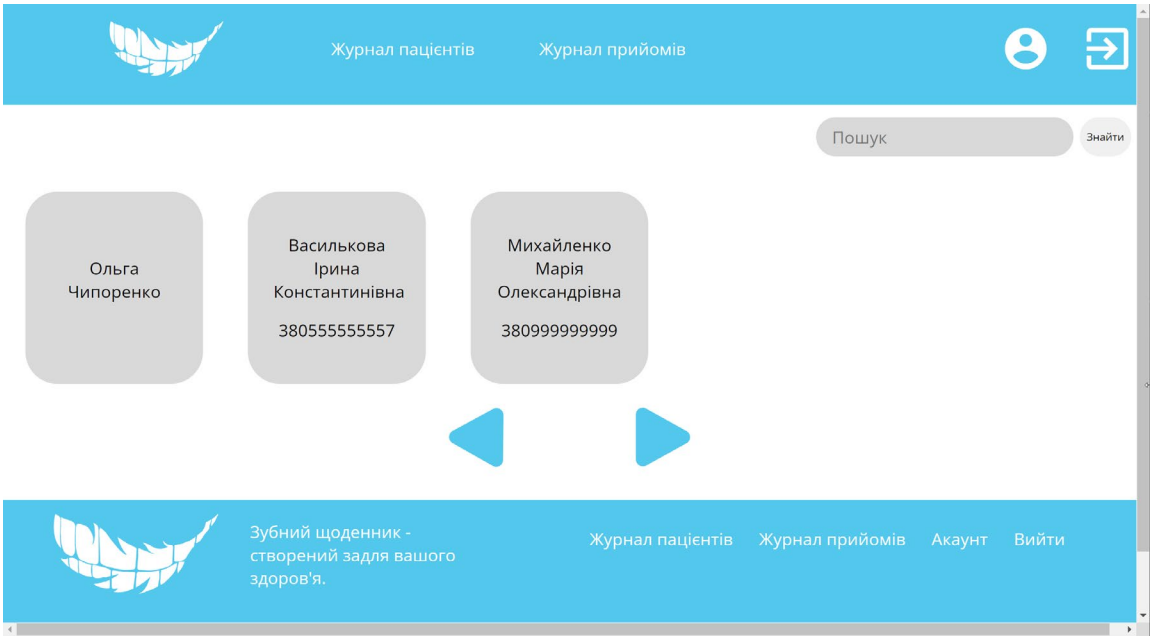

Рис.2.41. Сторінка «Журнал пацієнтів»

Також на цій сторінці реалізовано пошук щоденників за іменем пацієнта і перемикання між сторінками, у випадку якщо кількість щоденників перевищує число десять.

Однією із важливих функцій. яка була реалізована в застосунку є вихід з акаунту. Він відбувається після натискання на крайню кнопку, що знаходиться на навігаційній панелі. Після чого користувача направляють на головну сторінку.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки інформаційної системи

Початкові дані:

1. передбачуване число операторів програми – 1242;

2. коефіцієнт складності програми – 1,3;

3. коефіцієнт корекції програми в ході її розробки – 0,1;

4. годинна заробітна плата програміста – 152,7 грн/год;

За статистичними даними, що були взяті з веб-порталу «Української спільноти програмістів (DOU)»[35], станом на грудень 2022 місячна середня зарплата Junior Software Engineer з технології JavaScript становить 735$, що при офіційному курсі валют Національного банку України за червень (36,5686 грн) прирівнюється до 26877,9 грн. Значення стандартного графіку роботи береться як 176 год/місяць. Тоді годинна заробітна плата програміста становитиме 152,7 грн/год.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,5;

7. вартість машино-годин ЕОМ – 0,4 грн.

Комп’ютер, на якому велася розробка, в середньому використовує 70Вт на годину[36]. Ціна за електроенергію в Україні при споживанні до 250кВт на період розробки становила 1,44грн за 1 кВт[37]. Вартість витрат на електроенергію під час роботи комп’ютера за годину становить 0,07*1,44=0,1 грн. Підключення до інтернету від провайдера Vega проводиться за умовами тарифного плану «Оптика для кожного 2022»[38], щомісячна оплата якого становить 230 грн. Вартість використання інтернету за годину становить

0,3грн, що розраховується з урахуванням, що в місяці на 31 день 744 години. Отже, загальна вартість машинно-годин ЕОМ дорівнює 0,4 грн.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ розраховується за формулою:

$$t = t_o + t_{\text{и}} + t_{\text{п}} + t_{\text{отл}} + t_{\text{д}}, \text{людино} - \text{годин}, \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин),

$t_и$ – витрати праці на дослідження алгоритму рішення задачі,

$t_а$ – витрати праці на розробку блок-схеми алгоритму,

$t_п$ – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів:

$$Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q – передбачуване число операторів (1242),

C – коефіцієнт складності програми (1,3),

p – коефіцієнт корекції програми в ході її розробки (0,1).

Розрахуємо, умовне число операторів в програмі за формулою (3.2)

$$Q = 1242 \cdot 1{,}3 \cdot (1 + 0{,}1) = 1776{,}06$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_{\text{и}} = \frac{Q \cdot B}{(75 \cdots 85) \cdot k}, \text{людино} - \text{годин}, \quad (3.3)$$

де Q – умовне число операторів (1776,06),

B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2),

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1,5).

Розрахунок витрат праці на вивчення опису задачі:

$$t_{\text{и}} = \frac{1776{,}06 \cdot 1{,}2}{85 \cdot 1{,}5} = 16{,}715 \text{ людино} - \text{годин}.$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20 \cdots 25) \cdot k}, \text{людино} - \text{годин}, \quad (3.4)$$

де Q – умовне число операторів (1776,06),

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1,5).

Розрахунок:

$$t_{\text{а}} = \frac{1776{,}06}{25 \cdot 1{,}5} = 54{,}648 \text{ людино} - \text{годин}.$$

Витрати на складання програми по готовій блок-схемі, розраховуються за формулою:

$$t_{\text{п}} = \frac{Q}{(20 \cdots 25) \cdot k}, \text{людино} - \text{годин}, \quad (3.5)$$

де Q – умовне число операторів (1776,06),

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (1,5).

Розрахунок:

$$t_{\text{п}} = \frac{1776{,}06}{24 \cdot 1{,}5} = 49{,}335 \text{людино} - \text{годин}.$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4 \cdots 5) \cdot k}, \text{людино} - \text{годин}, \quad (3.6)$$

- за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1{,}5 \cdot t_{\text{отл}}, \text{людино} - \text{годин}, \quad (3.7)$$

Розрахунок витрат праці на налагодження програми ЕОМ за формулою (3.6):

$$t_{\text{отл}} = \frac{1776{,}06}{5 \cdot 1{,}5} = 236{,}808 \text{ людино} - \text{годин}.$$

Розрахунок витрат праці на налагодження програми ЕОМ за формулою (3.7):

$$t_{\text{отл}}^{k} = 1{,}5 \cdot 236{,}808 = 355{,}212 \text{ людино} - \text{годин}.$$

Витрати праці на підготовку документації, розраховуються за формулою:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{людино} - \text{годин}, \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису,

$$t_{\text{др}} = \frac{Q}{(15 \cdots 20) \cdot k}, \text{людино} - \text{годин}, \quad (3.9)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації.

$$t_{\text{до}} = 0{,}75 \cdot t_{\text{др}}, \text{людино} - \text{годин}, \quad (3.10)$$

Розрахунок трудомісткості підготовки матеріалів і рукопису за формулою (3.9):

$$t_{\text{др}} = \frac{1776{,}06}{20 \cdot 1{,}5} = 59{,}202 \text{ людино} - \text{годин}.$$

Розрахунок трудомісткості редагування, печатки й оформлення документації, за формулою (3.10):

$$t_{\text{до}} = 0{,}75 \cdot 59{,}202 = 44{,}4 \text{ людино} - \text{годин}.$$

Розрахунок праці на підготовку документації, за формулою (3.8):

$$t_д = 59{,}202 + 44{,}4 = 103{,}6 \text{ людино} - \text{годин}.$$

Розрахунок трудомісткості розробки програмного забезпечення за формулою (3.1):

t =50+16,715+54,648+49,335+236,808+103,6=632,469 людино-годин.

3.2. Рахунок витрат на створення інформаційної системи

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу $З_{МЧ}$, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МЧ}, \text{грн}, \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{грн}, \quad (3.12)$$

де t – загальна трудомісткість, людино-годин (632,469),

$С_{пр}$ – середня годинна заробітна плата програміста, грн/годину (152,7).

Розрахунок:

$$З_{ЗП} = 882{,}35 \cdot 152{,}7 = 96578{,}01 \text{ грн}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{Мч} = t_{отп} \cdot С_{Мч}, грн, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год. (236,808),

$C_{мч}$ – вартість машино-години ЕОМ, грн/год. (0,4).

Розрахунок:

$$З_{Мч} = 236{,}808 \cdot 0{,}4 = 94{,}723 \text{ грн}$$

Розрахунок витрат на створення програмного забезпечення за формулою (3.11):

$$К_{ПО} = 96578{,}01 + 94{,}723 = 96672{,}73 \text{ грн}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$Т = \frac{t}{B_k \cdot F_p}, міс, \quad (3.14)$$

де t – загальна трудомісткість, людино-годин (632,469),

B_k – число виконавців (1),

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Розрахунок:

$$Т = \frac{632{,}469}{176} = 3{,}59 \text{ міс.}$$

Висновок: Для розробки веб-орієнтованого застосунку ведення дентальної хроніки для запису і аналізу стану ротової порожнини загальна трудомісткість становить 632,469 людино-годин, згідно чого очікуваний період створення програмного забезпечення дорівнює приблизно 3 з половиною місяців, а витрати на створення – 96672,73 грн.

ВИСНОВКИ

Відповідно завданню і меті даної кваліфікаційної роботи було розроблено веб-орієнтований застосунок ведення дентальної хроніки для запису і аналізу стану ротової порожнини.

Проаналізувавши схожі за призначенням розробки в даній галузі застосування, було зроблено висновок, що до цього ще не було розроблено додатку з подібним призначенням. Основною відмінністю і особливістю якого полягає в наданні можливості ведення власної дентальної хроніки саме пацієнтові і вже при необхідності надавати доступ лікарю.

Практичне значення розробленого застосунку полягає в забезпеченні користувачів зручним інструментом догляду за здоров'ям ротової порожнини. Що також виконує функції мотивації і відслідковування.

Користувачеві надається можливість:

– записувати дані про лікувальні процеси, що складаються з записів і процедур;

– переглядати файли знімків власної ротової порожнини;

– відслідковувати кількість і період чистки зубів;

– контролювати частоту замін зубних щіток;

– спостерігати та відстежувати появу симптомів та жалоб на здоров'я;

З боку фахівця стоматологічної області, надається можливість вести і переглядати історію лікування пацієнтів при їх згоді і розширення доступу.

Також в застосунку було реалізовано реєстрація і авторизація.

Розробка застосунку має трирівневу клієнт-серверну архітектуру і базується на стеку технологій MERN, що включає реляційну базу даних MySQL, веб-фреймворк Express.js, бібліотеку React.js і серверний фреймворк Node.js.

За відомостями економічного розділу, трудомісткість проекту становить 632,469 людино-годи, очікуваний період створення застосунку

приблизна дорівнює три з половиною місяців, і витрати мають розмір – 96672,73 грн.

Щодо подальшого розвитку проекту планується розширення спектра процедур, що відображаються на графічній моделі ротової порожнини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронна система охорони здоров'я в Україні. URL: https://ehealth.gov.ua/ (дата звернення 18.05.2023).

2. Healow. Health and Online Wellness. URL: https://healowhelp.com/home/healowhelp.html (дата звернення: 18.05.2023).

3. MyChart. Secure online health connection. URL: https://www.mychart.org/ (дата звернення: 18.05.2023).

4. My Dental Care. URL: https://dentalcareapp.co.uk/ (дата звернення: 18.05.2023).

5. About BrushDJ. URL: https://www.brushdj.com/#nogo (дата звернення: 18.05.2023).

6. About Brusheez. URL: https://brusheez.com/pages/about-us (дата звернення: 18.05.2023).

7. Zotti F, Pietrobelli A, Malchiodi L, Nocini PF, Albanese M. Apps for oral hygiene in children 4 to 7 years: Fun and effectiveness. J Clin Exp Dent. 2019 Sep 1;11(9):e795-e801. doi: 10.4317/jced.55686. PMID: 31636871; PMCID: PMC6797448.

8. De Barros Lima, A.M.E.; Rodrigues, C.A.Q.; Haikal, D.S.A.; Silveira, M.F.; Mendes, D.C.; Oliviera, P.M.;Andrade, A.F.; de Feritas, C.V.; Pordeus, I.A. Desenvolvimento de um programa de computador para levantamentos epidemiológicos sobre condições de saúde bucal. Unimontes Científica 2012.

9. De Barros Lima, A.M.E.; Rodrigues, C.A.Q.; Haikal, D.S.A.; Silveira, M.F.; Mendes, D.C.; Oliviera, P.M.; Andrade, A.F.; de Feritas, C.V.; Pordeus, I.A. Desenvolvimento de um programa de computador para levantamentosepidemiológicos sobre condições de saúde bucal. Unimontes Científica 2012.

10. Detsomboonrat, P.; Pisarnturakit, P.P. Development and Evaluation: The Satisfaction of Using an Oral Health Survey Mobile Application. Telemed. e-Health 2019.

11. Cavalcante NV, Oliveira AH, Sá BVC, Botelho G, Moreira TR, Costa GDD, Cotta RMM. Computing and Oral Health: Mobile Solution for Collecting, Data Analysis, Managing and Reproducing Epidemiological Research in Population Groups. Int J Environ Res Public Health. 2020 Feb 8;17(3):1076. doi: 10.3390/ijerph17031076. PMID: 32046266; PMCID: PMC7036884.

12. FoodForTeeth – Healthier Teeth. URL: https://apps.apple.com/us/app/foodforteeth-healthier-teeth/id925367825 (дата звренення 07.06.2023).

13. Dental Record – Management app. URL: https://play.google.com/store/apps/details?id=com.app_dev_coders.DentalRecord&hl=en&gl=US (дата звернення: 07.06.2023).

14. About LocalMed. URL: https://www.localmed.com/about/ (дата звернення: 18.05.2023).

15. Kim C, Jeong H, Park W, Kim D. Tooth-Related Disease Detection System Based on Panoramic Images and Optimization Through Automation: Development Study. JMIR MedInform. 2022 Oct 31;10(10):e38640. doi: 10.2196/38640. PMID: 36315222; PMCID: PMC9664332.

16. Diagnocat. URL: https://diagnocat.com/ (дата звернення: 18.05.2023).

17. Tiffany B, Blasi P, Catz SL, McClure JB. Mobile Apps for Oral Health Promotion: Content Review and Heuristic Usability Analysis. JMIR Mhealth Uhealth. 2018 Sep 4;6(9):e11432. doi: 10.2196/11432. PMID: 30181114; PMCID: PMC6231784.

18. Архипов О.Є., Архипова Є.О. Особливості розуміння понять «інформаційна безпека» та «безпека інформації»: Матеріали XIV міжнародної науково-практичної конференції. – К.: ИПРИ НАН України, 2014. – С. 18-30.

19. Швець М.Ю., Заруба Д.С., Хохлов Ю.В. Порівняння SQL та NOSQL баз даних. Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки. 2018. Т.29(68), Ч.2, №6. С. 21-25.

20. Катренко А.В., Пасічник В.В. Прийняття рішень: теорія та практика. Львів: Новий світ – 2000, 2022. 447 с.

21. Techopedia – Client/Server Architecture. URL: https://www.techopedia.com/definition/438/clientserver-architecture (дата зверенення: 17.05.2023).

22. Client Server Architecture. URL: https://www.enjoyalgorithms.com/blog/client-server-architecture (дата зверенення: 17.05.2023).

23. JavaTpoint – MERN Stack. URL: https://www.javatpoint.com/mern-stack (дата звернення: 17.05.2023).

24. freeCodeCamp – The Model View Controller Pattern – MVC Architecture and Frameworks Explaines. URL: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/ (дата звернення: 17.05.2023).

25. David Flanagan JavaScript: The Denitive Guide. Seventh Edition. O'Reilly. 2020. 707p.

26. Sass – Documentation. URL: https://sass-lang.com/documentation/ (дата звернення: 17.05.2023).

27. MDN – JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 17.05.2023).

28. BEM – Documentation. URL: https://en.bem.info/methodology/quick-start/ (дата звернення: 17.05.2023).

29. Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення: 07.06.2023).

30.MySQL Workbench. URL: https://www.mysql.com/products/workbench/ (дата звернення: 09.06.2023)

31. Adobe – AdobeIllustrator. URL: https://www.adobe.com/products/illustrator.html (дата звернення: 07.06.2023).

32. StarUML. URL: https://staruml.io/ (дата звернення:07.06.2023).

33. EdrawSoft. URL: https://www.edrawsoft.com/ (дата звернення: 07.06.2023).

34. Github. URL: https://github.com/ (дата звернення: 07.06.2023).

35. Статистичні дані з розподілу зарплат. – грудень 2022. URL: https://jobs.dou.ua/salaries/?period=2022-12&position=Junior%20SE&technology=JavaScript&education=3 (дата звернення: 05.06.2023).

36. EcoCostSavings. How Many watts does a laptop use. URL: https://ecocostsavings.com/how-many-watts-does-a-laptop-use/ (дата звернення: 05.06.2023).

37. Yasno. Тарифи на електроенергію. URL: https://yasno.com.ua/news/yasno_news/Electricity_tariffs_for_the_population_remain_unchanged (дата звернення: 05.06.2023).

38. Vega. Умови тарифних планів послуги Інтернет для фізичних осіб. URL: https://vega.ua/files/Umovy_tarifnyh_planiv.pdf (дата звернення: 05.06.2023).

39. Михайленко М.О. Інформаційна система ведення дентальної хроніки для запису і аналізу стану ротової порожнини. Тиждень студентської науки – 2023: матеріали 78-ї студент. наук.-тезн. конф.,м. Дніпро, 24-28 квітня 2023. С.374-376.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

index.js // головний файл запуску серверної частини

```
//підключення всіх необхідних модулів
import express from "express";
import cors from "cors";
import db from "./dbconnect/Config.js";
import multer from "multer";
import authRouter from "./routers/auth.js";
import diaryRouter from "./routers/diaries.js";
import dirProcedureRouter from "./routers/directoryProcedures.js";
import doctorRouter from "./routers/doctors.js";
import ocvRouter from "./routers/oralCavityParts.js";
import procedureRouter from "./routers/procedures.js";
import recordRouter from "./routers/records.js";
import treatmentRouter from "./routers/treatments.js";
import userRouter from "./routers/users.js";
import brushingsRouter from "./routers/brushings.js";
import brushesRouter from "./routers/brushes.js";
import accessRouter from "./routers/accesses.js";
import fileRouter from "./routers/files.js";
import areaRouter from "./routers/areas.js"
import symptomRouter from "./routers/symptoms.js";
import cookieParser from "cookie-parser";
const app = express();
//проміжні програмні забезпечення
app.use((req, res, next)=>{
   res.header("Access-Control-Allow-Credentials",true);
   next();
})
app.use(cookieParser())
app.use(express.json())
app.use(cors({
   origin: "http://localhost:3000",
}))
//організація збереження файлів в локальне сховище
const storage = multer.diskStorage({
   destination: function (req, file, callBack) {
     callBack(null, "../frontend/git1/git/src/upload");
   },
   filename: function (req, file, callBack) {
      callBack(null, Date.now() + file.originalname);
   },
 });
 const upload = multer({ storage: storage });
 app.post("/api/upload", upload.single("file"), (req, res) => {
   const file = req.file;
   console.log("файл завантажено");
   res.status(200).json(file.filename);
 });
app.get('/', (req,res)=>{
   res.json("Hello server")
})
//налаштування маршрутизації
app.use("/auth", authRouter);
app.use("/diaries", diaryRouter);
app.use("/directoryProcedures", dirProcedureRouter);
app.use("/doctors", doctorRouter);
app.use("/oralCavityParts", ocvRouter);
```

```
app.use("/procedures", procedureRouter);
app.use("/records", recordRouter);
app.use("/treatments", treatmentRouter);
app.use("/users", userRouter);
app.use("/brushing", brushingsRouter);
app.use("/brushes", brushesRouter);
app.use("/accesses", accessRouter);
app.use("/files", fileRouter);
app.use("/areas", areaRouter);
app.use("/symptoms", symptomRouter);
app.use((err,req,res,next)=>{
   const errorStatus = err.status || 500
   const errorMessage = err.message || "Something went wrong!"
   return res.status(errorStatus).json({
      success: false,
      status: errorStatus,
      message: errorMessage,
      stack:err.stack
   })
})
//запуск серверу і підключення до бази даних
app.listen(8800, ()=>{
   console.log("Сервер працює!");
   db.connect(function(err){
      if (err) throw err;
      console.log("База даних підключена");
   })
})
```

procedures.js
```
// файл маршрутів процедур
import express from "express";
const router =express.Router();
import { getProceduresView, deleteProcedure, createProcedure , getProceduresByRecord, updateProcedure} from
"../controllers/procedure.js";
//отримання даних з представлення
router.get('/view', getProceduresView);
//отримання процедур за записом
router.get('/view/rec', getProceduresByRecord);
//видалення
router.delete('/api/:idprocedure', deleteProcedure);
//створення
router.post('/',createProcedure);
//оновлення
router.put('/api/:idprocedure', updateProcedure );
export default router;
```

procedure.js
```
// файл контролеру процедури
//підключення необхідних модулів
import Model from "../dbconnect/model.js";
import { verifyToken } from "../utils/verifyToken.js";
//моделі таблиць бази даних
let procedureM = new Model("teethdiary.procedure");
let procedureVM = new Model("procedure_data");
let areaV = new Model("procedure_area")
let fileV = new Model("fileph")
let fieldId ="idprocedure";
//отримання даних з представлення
export const getProceduresView = async(req,res,next)=>{
   const userId = req.query.userId;
   let result;
   verifyToken(req, res,()=>{
      if(userId!==undefined){
```

```
            result = procedureVM.find("user_id",userId);
        }else{
            result = procedureVM.find("user_id",req.user.iduser);
        }
        result.then(function(value){
            console.log("All procedures (view)");
            res.json(value)})
            .catch(function(error){
                console.log( error );
                next(error);
            })
    });
}
//отримання процедур за записом
export const getProceduresByRecord = (req,res,next)=>{
    const idrecord = req.query.idrecord;
    if(idrecord>=0){
        const result = procedureVM.find("idrecord", idrecord);
        result.then(function(value){
            console.log("Процедури обраного запису");
            res.json(value)})
            .catch(function(error){
                console.log( error );
                next(error);
            })
    }
}
//видалення процедури
export const deleteProcedure = async(req, res, next)=>{
    let idProcedure= req.params.idprocedure;
    console.log("procedure: "+ idProcedure);
    let proc_area = await areaV.find("procedure_id", idProcedure);
    if(proc_area.length<1){ console.log("Область застосування не знайдена");
    }else{
        proc_area.forEach( async(element) => {
            let areaDel = await areaV.delete(element.id_area, "id_area");
        });
    }
    let files = await fileV.find("procedure_id", idProcedure);
    if(files<1){ console.log("Файлів не знайдено");
    }else{
        files.forEach( async(element) => {
            let fileDel = await fileV.delete(element.idfile, "idfile");
        });
    }
    let result = procedureM.delete(idProcedure, "idprocedure");
    result.then(
        console.log( "Процедура була видалена" )
    )
    .catch( function(error){
        console.log( error );
        next(error);
    });
}
//створення процедури
export const createProcedure = async(req, res, next)=>{
   const params="`procedures_id`,`record_id`";
    const values=[
        req.body.procedures_id,
        req.body.record_id,
    ];
    let result = procedureM.create(params, values, fieldId);
```

```
    result.then( function(value){
        console.log( "Процедуру було створено" );
        res.json( value);})
    .catch( function(error){
        console.log( error )
        next(error);
    });
}
//оновлення даних процедури
export const updateProcedure = (req,res, next)=>{
    const params={
        procedures_id: req.body.procedures_id
    }
    let result = procedureM.update(req.params.idprocedure, params, fieldId);
    result.then( function(value){
        console.log("Процедура була оновлена");
        res.json( value);
    })
    .catch( function(error){
        console.log( error );
        next(error);});
}
```

model.js // файл модуля моделі

```
//підключення модуля бази даних
import db from "./Config.js";
//класс Моделі
class Model{
    constructor(table){
        this.table=table;
    }
//CRUD операції
// пошук всіх даних за вказаною таблицею
    get_all(){
        let field=this.table;
        return new Promise(function(myResolve, myReject){
            db.query('SELECT * FROM ' +field, function (error,result){
                if(error) throw error;
                myResolve(result);
            });
        });
    }
    //пошук за вказаним параметром
    find(fieldId,id){
        let cThis = this;
        return new Promise(function(myResolve, myReject){
            db.query(`SELECT * FROM ?? WHERE ${fieldId} = ?`, [cThis.table,id], function (error,result){
                if(error) throw error;
                myResolve(result);
            });
        });
    }
    //пошук за декількома параметрами
    findPlus(fieldId,id, plus){
        let cThis = this;
        return new Promise(function(myResolve, myReject){
            db.query(`SELECT * FROM ?? WHERE ${fieldId} = ? `+plus, [cThis.table,id], function (error,result){
                if(error) throw error;
                myResolve(result);
            });
        });
    }
```

```
    //створення даних
    create(params, data, fieldId){
        let cThis = this;
        return new Promise(function(myResolve, myReject){
            db.query(`INSERT INTO ${cThis.table} (${params}) VALUES (?)`, [ data], function(error, result,fields){
                if(error) throw error;
                let data1 = cThis.find(fieldId,result.insertId);
                data1.then(function(value){myResolve(value)})
                .catch(function(error){myReject(error)})
            });
        });
    }
    //оновлення даних
    update(id,data,fieldId){
        let cThis=this;
        return new Promise(function(myResolve, myReject){
            db.query(`UPDATE ${cThis.table} SET ? WHERE ${fieldId}=?`,[ data,id], function(error, result,fields){
                if(error) throw error;
                let data1 = cThis.find(fieldId,result.insertId);
                data1.then(function(value){myResolve(value)})
                .catch(function(error){myReject(error)})
            });
        })
    }
    //видалення даних
    delete(id, fieldId){
        let cThis=this;
        return new Promise(function(myResolve, myReject){
            db.query(`DELETE FROM ${cThis.table} WHERE ${fieldId}=${id}`, function(error, result,fields){
                if(error) throw error;
                let data1 = cThis.find(fieldId,result.insertId);
                data1.then(function(value){myResolve(value)})
                .catch(function(error){myReject(error)})
            });
        })
    }
}
export default Model;
```

App.js // головний файл клієнтської частини

```
//підключення необхідних компонентів
import {createBrowserRouter,
  RouterProvider,Outlet,
  Navigate
  } from "react-router-dom"
import { AuthContext } from "./context/authContext";
import { useContext } from "react";
import {QueryClient, QueryClientProvider} from 'react-query';
import Register from "./view/Register/Register";
import Login from "./view/Login/Login"
import BrushingDiary from "./view/BrushingDiary/BrushingDiary"
import Home from "./view/Home/Home"
import PatientDiary from "./view/PatientDiary/PatientDiary"
import Procedure from "./view/Procedure/Procedure"
import TeethRecord from "./view/TeethRecord/TeethRecord"
import ToothBrushDiary from "./view/ToothBrushDiary/ToothBrushDiary"
import Treatment from "./view/Treatment/Treatment"
import Account from "./view/Account/Account"
import Accesses from "./view/Accesses/Accesses"
import SymptomDiary from "./view/SymptomDiary/SymptomDiary"
import FilePF from "./view/File/File";
import Header from "./components/header/header1"
```

```
import Footer from "./components/footer/footer1"
import Header2 from "./components/header/header2"
import Footer2 from "./components/footer/footer2"
import Header3 from "./components/header/header3"
import Footer3 from "./components/footer/footer3"
//реалізація навігації та шляхів сайту
function App() {
  const queryClient = new QueryClient();
   const { currentUser } = useContext(AuthContext);
const Layout = () =>{
  return (
    <QueryClientProvider client={queryClient}>
    <Header/>
    <Outlet/>
    <Footer/>
    </QueryClientProvider>
  )
}
const Layout2 = () =>{
  return (
    <QueryClientProvider client={queryClient}>
    <Header2/>
    <Outlet/>
    <Footer2/>
    </QueryClientProvider>
  )
}
const Layout3 = () =>{
  return (
    <QueryClientProvider client={queryClient}>
    <Header3/>
    <Outlet/>
    <Footer3/>
    </QueryClientProvider>
  )
}
//фільтрування за типом користувача
   const UserFilter =({ children })=>{
    if(!currentUser){
      return  <Navigate to="/" />;
    }else{
      if (currentUser.usertype=='пацієнт'){
        return <Layout2 />;
      }else if (currentUser.usertype=='лікар'){
        return <Layout3 />;
      }
      else{
        return <Layout />;
      }
    }
   }
  const router = createBrowserRouter([
    {
      path:"/",
      element: (
          <Layout />
      ),
      children:[
        {
          path:"/",
          element: <Home/>
        },
```

```
    {
     path:"/register",
     element: <Register/>
    },
    {
     path:"/login",
     element: <Login/>
    }
   ]
},
{
 path:"/",
 element: (
     <UserFilter />
 ),
 children:[
    {
     path:"/brushing",
     element: <BrushingDiary/>
    },
    {
     path:"/diary",
     element: <PatientDiary/>
    },
    {
     path:"/procedure",
     element: <Procedure/>
    },
    {
     path:"/record",
     element: <TeethRecord/>
    },
    {
     path:"/brushes",
     element: <ToothBrushDiary/>
    },
    {
     path:"/treatment",
     element: <Treatment/>
    },
    {
     path:"/symptoms",
     element: <SymptomDiary/>
    },
    {
     path:"/procedure_file",
     element:<FilePF/>
    }
 ]
},
{
 path:"/",
 element: <UserFilter/>,
 children:[
    {
     path:"/accesses",
     element: <Accesses/>
    },
    {
     path:"/account",
     element: <Account/>
    }
```

```
    ]
   },
  ]);
  return (
   <div>
    <RouterProvider router={router}/>
   </div>
  );
}
export default App;
```

PatientDiary.jsx // модуль сторінки щоденника пацієнтів

```
//підключення необхідних модулів
import React, { useEffect, useState, useRef }from 'react'
import {MultiSelect} from "react-multi-select-component";
import "./patient_diary.scss";
import "../../components/procedureCard/procedure_card.scss";
import "../../components/treatmentCard/treatment_card.scss";
import "../../components/recordCard/record_card.scss";
import {useQuery } from 'react-query'
import { Link } from "react-router-dom";
import TreatmentCard from "../../components/treatmentCard/treatmentCard.jsx"
import RecordCard from '../../components/recordCard/recordCard';
import ProcedureCard from '../../components/procedureCard/procedureCard';
import ResultBox from '../../components/result_box/resultBox';
import { makeRequest } from '../../axios';
//підключення ілюстрацій частин ротової порожнини
//стандартний набір
import  mouth_img from '../../images/oralCavity/standart/oral_cavity.svg'
import tooth11_img from '../../images/oralCavity/standart/11.svg'
import  tooth12_img from '../../images/oralCavity/standart/12.svg'
import  tooth13_img from '../../images/oralCavity/standart/13.svg'
import  tooth14_img from '../../images/oralCavity/standart/14.svg'
import  tooth15_img from '../../images/oralCavity/standart/15.svg'
import  tooth16_img from '../../images/oralCavity/standart/16.svg'
import  tooth17_img from '../../images/oralCavity/standart/17.svg'
import  tooth18_img from '../../images/oralCavity/standart/18.svg'
import tooth21_img from '../../images/oralCavity/standart/21.svg'
import  tooth22_img from '../../images/oralCavity/standart/22.svg'
import  tooth23_img from '../../images/oralCavity/standart/23.svg'
import  tooth24_img from '../../images/oralCavity/standart/24.svg'
import  tooth25_img from '../../images/oralCavity/standart/25.svg'
import tooth26_img from '../../images/oralCavity/standart/26.svg'
import tooth27_img from '../../images/oralCavity/standart/27.svg'
import tooth28_img from '../../images/oralCavity/standart/28.svg'
import tooth31_img from '../../images/oralCavity/standart/31.svg'
import tooth32_img from '../../images/oralCavity/standart/32.svg'
import tooth33_img from '../../images/oralCavity/standart/33.svg'
import tooth34_img from '../../images/oralCavity/standart/34.svg'
import tooth35_img from '../../images/oralCavity/standart/35.svg'
import tooth36_img from '../../images/oralCavity/standart/36.svg'
import tooth37_img from '../../images/oralCavity/standart/37.svg'
import tooth38_img from '../../images/oralCavity/standart/38.svg'
import tooth41_img from '../../images/oralCavity/standart/41.svg'
import tooth42_img from '../../images/oralCavity/standart/42.svg'
import tooth43_img from '../../images/oralCavity/standart/43.svg'
import tooth44_img from '../../images/oralCavity/standart/44.svg'
import tooth45_img from '../../images/oralCavity/standart/45.svg'
import tooth46_img from '../../images/oralCavity/standart/46.svg'
import tooth47_img from '../../images/oralCavity/standart/47.svg'
import tooth48_img from '../../images/oralCavity/standart/48.svg'
//зуби після адгезивної реставрації
```

```
import fill_tooth11_img from '../../images/oralCavity/fillings/п11.svg'
import fill_tooth12_img from '../../images/oralCavity/fillings/п12.svg'
import fill_tooth13_img from '../../images/oralCavity/fillings/п13.svg'
import fill_tooth14_img from '../../images/oralCavity/fillings/п14.svg'
import fill_tooth15_img from '../../images/oralCavity/fillings/п15.svg'
import fill_tooth16_img from '../../images/oralCavity/fillings/п16.svg'
import fill_tooth17_img from '../../images/oralCavity/fillings/п17.svg'
import fill_tooth18_img from '../../images/oralCavity/fillings/п18.svg'
import fill_tooth21_img from '../../images/oralCavity/fillings/п21.svg'
import fill_tooth22_img from '../../images/oralCavity/fillings/п22.svg'
import fill_tooth23_img from '../../images/oralCavity/fillings/п23.svg'
import fill_tooth24_img from '../../images/oralCavity/fillings/п24.svg'
import fill_tooth25_img from '../../images/oralCavity/fillings/п25.svg'
import fill_tooth26_img from '../../images/oralCavity/fillings/п26.svg'
import fill_tooth27_img from '../../images/oralCavity/fillings/п27.svg'
import fill_tooth28_img from '../../images/oralCavity/fillings/п28.svg'
import fill_tooth31_img from '../../images/oralCavity/fillings/п31.svg'
import fill_tooth32_img from '../../images/oralCavity/fillings/п32.svg'
import fill_tooth33_img from '../../images/oralCavity/fillings/п33.svg'
import fill_tooth34_img from '../../images/oralCavity/fillings/п34.svg'
import fill_tooth35_img from '../../images/oralCavity/fillings/п35.svg'
import fill_tooth36_img from '../../images/oralCavity/fillings/п36.svg'
import fill_tooth37_img from '../../images/oralCavity/fillings/п37.svg'
import fill_tooth38_img from '../../images/oralCavity/fillings/п38.svg'
import fill_tooth41_img from '../../images/oralCavity/fillings/п41.svg'
import fill_tooth42_img from '../../images/oralCavity/fillings/п42.svg'
import fill_tooth43_img from '../../images/oralCavity/fillings/п43.svg'
import fill_tooth44_img from '../../images/oralCavity/fillings/п44.svg'
import fill_tooth45_img from '../../images/oralCavity/fillings/п45.svg'
import fill_tooth46_img from '../../images/oralCavity/fillings/п46.svg'
import fill_tooth47_img from '../../images/oralCavity/fillings/п47.svg'
import fill_tooth48_img from '../../images/oralCavity/fillings/п48.svg'
// зуби після екстирпації пульпи або хірургічної обробки кореневого каналу
import nerv_tooth11_img from '../../images/oralCavity/nerves/бн11.svg'
import nerv_tooth12_img from '../../images/oralCavity/nerves/бн12.svg'
import nerv_tooth13_img from '../../images/oralCavity/nerves/бн13.svg'
import nerv_tooth14_img from '../../images/oralCavity/nerves/бн14.svg'
import nerv_tooth15_img from '../../images/oralCavity/nerves/бн15.svg'
import nerv_tooth16_img from '../../images/oralCavity/nerves/бн16.svg'
import nerv_tooth17_img from '../../images/oralCavity/nerves/бн17.svg'
import nerv_tooth18_img from '../../images/oralCavity/nerves/бн18.svg'
import nerv_tooth21_img from '../../images/oralCavity/nerves/бн21.svg'
import nerv_tooth22_img from '../../images/oralCavity/nerves/бн22.svg'
import nerv_tooth23_img from '../../images/oralCavity/nerves/бн23.svg'
import nerv_tooth24_img from '../../images/oralCavity/nerves/бн24.svg'
import nerv_tooth25_img from '../../images/oralCavity/nerves/бн25.svg'
import nerv_tooth26_img from '../../images/oralCavity/nerves/бн26.svg'
import nerv_tooth27_img from '../../images/oralCavity/nerves/бн27.svg'
import nerv_tooth28_img from '../../images/oralCavity/nerves/бн28.svg'
import nerv_tooth31_img from '../../images/oralCavity/nerves/бн31.svg'
import nerv_tooth32_img from '../../images/oralCavity/nerves/бн32.svg'
import nerv_tooth33_img from '../../images/oralCavity/nerves/бн33.svg'
import nerv_tooth34_img from '../../images/oralCavity/nerves/бн34.svg'
import nerv_tooth35_img from '../../images/oralCavity/nerves/бн35.svg'
import nerv_tooth36_img from '../../images/oralCavity/nerves/бн36.svg'
import nerv_tooth37_img from '../../images/oralCavity/nerves/бн37.svg'
import nerv_tooth38_img from '../../images/oralCavity/nerves/бн38.svg'
import nerv_tooth41_img from '../../images/oralCavity/nerves/бн41.svg'
import nerv_tooth42_img from '../../images/oralCavity/nerves/бн42.svg'
import nerv_tooth43_img from '../../images/oralCavity/nerves/бн43.svg'
import nerv_tooth44_img from '../../images/oralCavity/nerves/бн44.svg'
import nerv_tooth45_img from '../../images/oralCavity/nerves/бн45.svg'
```

```
import nerv_tooth46_img from '../../images/oralCavity/nerves/бн46.svg'
import nerv_tooth47_img from '../../images/oralCavity/nerves/бн47.svg'
import nerv_tooth48_img from '../../images/oralCavity/nerves/бн48.svg'
//зуби покриті нальотом
import plag_tooth11_img from '../../images/oralCavity/plaque/н11.svg'
import plag_tooth12_img from '../../images/oralCavity/plaque/н12.svg'
import plag_tooth13_img from '../../images/oralCavity/plaque/н13.svg'
import plag_tooth14_img from '../../images/oralCavity/plaque/н14.svg'
import plag_tooth15_img from '../../images/oralCavity/plaque/н15.svg'
import plag_tooth16_img from '../../images/oralCavity/plaque/н16.svg'
import plag_tooth17_img from '../../images/oralCavity/plaque/н17.svg'
import plag_tooth18_img from '../../images/oralCavity/plaque/н18.svg'
import plag_tooth21_img from '../../images/oralCavity/plaque/н21.svg'
import plag_tooth22_img from '../../images/oralCavity/plaque/н22.svg'
import plag_tooth23_img from '../../images/oralCavity/plaque/н23.svg'
import plag_tooth24_img from '../../images/oralCavity/plaque/н24.svg'
import plag_tooth25_img from '../../images/oralCavity/plaque/н25.svg'
import plag_tooth26_img from '../../images/oralCavity/plaque/н26.svg'
import plag_tooth27_img from '../../images/oralCavity/plaque/н27.svg'
import plag_tooth28_img from '../../images/oralCavity/plaque/н28.svg'
import plag_tooth31_img from '../../images/oralCavity/plaque/н31.svg'
import plag_tooth32_img from '../../images/oralCavity/plaque/н32.svg'
import plag_tooth33_img from '../../images/oralCavity/plaque/н33.svg'
import plag_tooth34_img from '../../images/oralCavity/plaque/н34.svg'
import plag_tooth35_img from '../../images/oralCavity/plaque/н35.svg'
import plag_tooth36_img from '../../images/oralCavity/plaque/н36.svg'
import plag_tooth37_img from '../../images/oralCavity/plaque/н37.svg'
import plag_tooth38_img from '../../images/oralCavity/plaque/н38.svg'
import plag_tooth41_img from '../../images/oralCavity/plaque/н41.svg'
import plag_tooth42_img from '../../images/oralCavity/plaque/н42.svg'
import plag_tooth43_img from '../../images/oralCavity/plaque/н43.svg'
import plag_tooth44_img from '../../images/oralCavity/plaque/н44.svg'
import plag_tooth45_img from '../../images/oralCavity/plaque/н45.svg'
import plag_tooth46_img from '../../images/oralCavity/plaque/н46.svg'
import plag_tooth47_img from '../../images/oralCavity/plaque/н47.svg'
import plag_tooth48_img from '../../images/oralCavity/plaque/н48.svg'
//модуль Щоденика пацієнта
const PatientDiary = () => {
let userId;
let patientId;
if(JSON.parse(localStorage.getItem("patient"))){
 patientId=JSON.parse(localStorage.getItem("patient")).idPatient;
 if(patientId>=0){
  userId=patientId;
 }
}else{
  userId= JSON.parse(localStorage.getItem("user")).iduser;
}
const [selected, setSelected] = useState([]);
const [selected2, setSelected2] = useState([]);
const [dateStFilter, setDateStFilter] =useState([]);
const [dateEnFilter, setDateEnFilter] =useState([]);
const [inputs, setInputs] = useState({
  username:"",
  phone:""
});
const handleChange = (e) =>{
  setInputs((prev) => ({ ...prev, [e.target.name]: e.target.value }));
};
let teethNumber =32;
localStorage.setItem("treatment", JSON.stringify({"":""}))
localStorage.setItem("record", JSON.stringify({"":""}))
```

```
localStorage.setItem("procedure", JSON.stringify({"":""}))
localStorage.setItem("doctor",JSON.stringify({doctorId:"",username:""}));
//отримання даних з серверу
//лікування
const {isLoading:treatmentLoading, error:treatmentError, data:treatmentData} =useQuery(['treatments'], ()=>
makeRequest.get("/treatments/view?userId="+userId).then(res=>{
  res.data.forEach((el)=>{
    if(el.start_date_trtment!==null && el.start_date_trtment!==""){
    let stDate = new Date(el.start_date_trtment);
    el.start_date_trtment= `${stDate.getFullYear()}-${stDate.getMonth()+1}-${stDate.getDate()}`
    }
    if(el.end_date_trtment!==null && el.end_date_trtment!==""){
    let enDate = new Date(el.end_date_trtment);
    el.end_date_trtment = `${enDate.getFullYear()}-${enDate.getMonth()+1}-${enDate.getDate()}`
    }
  })
  return res.data;
}));
//медичні записи
const {isLoading:recordsLoading, error: recordsError, data: recordsData} =useQuery(['records'], ()=>
makeRequest.get("/records/view?userId="+userId).then(res=>{
  res.data.forEach((el)=>{
    if(el.record_date!==null && el.record_date!==""){
      let recDate = new Date(el.record_date);
      el.record_date = `${recDate.getFullYear()}-${recDate.getMonth()+1}-${recDate.getDate()}`
    }
  })
  return res.data;
}));
//процедури
const {isLoading: proceduresLoading, error: proceduresError, data: proceduresData} =useQuery(['procedures'], ()=>
makeRequest.get("/procedures/view?userId="+userId).then(res=>{
  return res.data;
}));
//зубні щітки
const {isLoading: brushesLoading, error: brushesError, data: brushesData} =useQuery(['brushes'], ()=>
makeRequest.get("/brushes?userId="+userId).then(res=>{
  return res.data;
}));
//дані про чистки зубів
const {isLoading: brushingLoading, error: brushingError, data: brushingData} =useQuery(['brushing'], ()=>
makeRequest.get("/brushing?userId="+userId).then(res=>{
  return res.data;
}));
//частини ротової попрожнини
const {isLoading: ocpLoading, error: ocpError, data: ocpData} =useQuery(['oral_cavity_parts'], ()=>
makeRequest.get("/oralCavityParts").then(res=>{
  return res.data;
}));
//системні процедури
const {isLoading: dirProceduresLoading, error: dirProceduresError, data: dirProceduresData}
=useQuery(['dir_procedures'], ()=>
makeRequest.get("/directoryProcedures").then(res=>{
  return res.data;
}));
let dateNow =new Date();
let diffDays =0;
let lastProBrushing=0;
let diffInProBrushing =0;
let brushing=0;
let brushChange =0;
let oral_cavity_parts=[];
```

```
let procedures=[];
// посилання на елементи моделі зубної порожнини та інші
const tooth_11 = useRef(null);
const tooth_12 = useRef(null);
const tooth_13 = useRef(null);
const tooth_14  = useRef(null);
const tooth_15  = useRef(null);
const tooth_16  = useRef(null);
const tooth_17  = useRef(null);
const tooth_18  = useRef(null);
const tooth_21 = useRef(null);
const tooth_22  = useRef(null);
const tooth_23  = useRef(null);
const tooth_24 = useRef(null);
const tooth_25 = useRef(null);
const tooth_26 = useRef(null);
const tooth_27 = useRef(null);
const tooth_28 = useRef(null);
const tooth_31 = useRef(null);
const tooth_32 = useRef(null);
const tooth_33  = useRef(null);
const tooth_34  = useRef(null);
const tooth_35  = useRef(null);
const tooth_36 = useRef(null);
const tooth_37 = useRef(null);
const tooth_38 = useRef(null);
const tooth_41 = useRef(null);
const tooth_42 = useRef(null);
const tooth_43 = useRef(null);
const tooth_44 = useRef(null);
const tooth_45 = useRef(null);
const tooth_46 = useRef(null);
const tooth_47 = useRef(null);
const tooth_48 = useRef(null);
const filter_box = useRef(null);
const checkbox = useRef();
//обробка отриманих даних
if(recordsData!==undefined && proceduresData!==undefined && brushingData!==undefined &&
brushesData!==undefined){
  //розрахунок часу з останнього відвідування стоматолога
if(recordsData.length>0){
    const dateLastRecord = new Date(recordsData[recordsData.length-1].record_date);
    const diffTime = Math.abs(dateNow-dateLastRecord);
    diffDays = Math.ceil(diffTime/(1000*60*60*24));
    }
    //рахунок загальну кількість зубів
    let count =0;
    proceduresData.forEach(el => {
      if(el.proc_name.includes("Видалення зуба або його частин(-и)")|| el.proc_name.includes("Секційне
видалення зуба або його частин(-и)") || el.proc_name.includes("видалення зуба")){
        count++;
      }else if (el.proc_name.includes("Видалення всіх зубів")){
        teethNumber=0;
      }
      })
  teethNumber=teethNumber-count;
  //розрахунок часу з останньої професійної чистки
  recordsData.forEach((el)=> {
    if(el.proc_names!==null){
    if(el.proc_names.includes("Відбілювання, внутрішнє") ||el.proc_names.includes("Відбілювання, в домашніх
умовах")|| el.proc_names.includes("Відбілювання, зовнішнє") || el.proc_names.includes("Видалення нальоту або
плям на зубах")){
```

```
    lastProBrushing= el;
  }
  }
})
let lastProBrushingDate = new Date(lastProBrushing.record_date);
diffInProBrushing=Math.ceil(Math.abs(dateNow-lastProBrushingDate)/(1000*60*60*24))
if(isNaN(diffInProBrushing)){
  diffInProBrushing=0;
}
brushingData.forEach((el)=>{
  let dateBr = new Date(el.brushing_date);
  let diffBr = Math.ceil(Math.abs(dateNow-dateBr)/(1000*60*60*24));
  if(diffBr<=1){
    brushing=el.number_of_times;
  }
})
//розрахунок часу з останьої заміни зубної щітки
brushesData.forEach((el)=>{
  let dateBr = new Date(el.start_date);
  brushChange = Math.ceil(Math.abs(dateNow-dateBr)/(1000*60*60*24));
})
// реалізація реагування моделі ротової порожнини на проведені процедури
if(tooth_11.current!==undefined && tooth_11.current!==null){
  if(diffInProBrushing<365){
    tooth_11.current.src = tooth11_img;
    tooth_12.current.src = tooth12_img;
    tooth_13.current.src = tooth13_img;
    tooth_14.current.src = tooth14_img;
    tooth_15.current.src = tooth15_img;
    tooth_16.current.src = tooth16_img;
    tooth_17.current.src = tooth17_img;
    tooth_18.current.src = tooth18_img;
    tooth_21.current.src = tooth21_img;
    tooth_22.current.src = tooth22_img;
    tooth_23.current.src = tooth23_img;
    tooth_24.current.src = tooth24_img;
    tooth_25.current.src = tooth25_img;
    tooth_26.current.src = tooth26_img;
    tooth_27.current.src = tooth27_img;
    tooth_28.current.src = tooth28_img;
    tooth_31.current.src = tooth31_img;
    tooth_32.current.src = tooth32_img;
    tooth_33.current.src = tooth33_img;
    tooth_34.current.src = tooth34_img;
    tooth_35.current.src = tooth35_img;
    tooth_36.current.src = tooth36_img;
    tooth_37.current.src = tooth37_img;
    tooth_38.current.src = tooth38_img;
    tooth_41.current.src = tooth41_img;
    tooth_42.current.src = tooth42_img;
    tooth_43.current.src = tooth43_img;
    tooth_44.current.src = tooth44_img;
    tooth_45.current.src = tooth45_img;
    tooth_46.current.src = tooth46_img;
    tooth_47.current.src = tooth47_img;
    tooth_48.current.src = tooth48_img;
  }else{
    tooth_11.current.src = plag_tooth11_img;
    tooth_12.current.src = plag_tooth12_img;
    tooth_13.current.src = plag_tooth13_img;
    tooth_14.current.src = plag_tooth14_img;
    tooth_15.current.src = plag_tooth15_img;
```

```
    tooth_16.current.src = plag_tooth16_img;
    tooth_17.current.src = plag_tooth17_img;
    tooth_18.current.src = plag_tooth18_img;
    tooth_21.current.src = plag_tooth21_img;
    tooth_22.current.src = plag_tooth22_img;
    tooth_23.current.src = plag_tooth23_img;
    tooth_24.current.src = plag_tooth24_img;
    tooth_25.current.src = plag_tooth25_img;
    tooth_26.current.src = plag_tooth26_img;
    tooth_27.current.src = plag_tooth27_img;
    tooth_28.current.src = plag_tooth28_img;
    tooth_31.current.src = plag_tooth31_img;
    tooth_32.current.src = plag_tooth32_img;
    tooth_33.current.src = plag_tooth33_img;
    tooth_34.current.src = plag_tooth34_img;
    tooth_35.current.src = plag_tooth35_img;
    tooth_36.current.src = plag_tooth36_img;
    tooth_37.current.src = plag_tooth37_img;
    tooth_38.current.src = plag_tooth38_img;
    tooth_41.current.src = plag_tooth41_img;
    tooth_42.current.src = plag_tooth42_img;
    tooth_43.current.src = plag_tooth43_img;
    tooth_44.current.src = plag_tooth44_img;
    tooth_45.current.src = plag_tooth45_img;
    tooth_46.current.src = plag_tooth46_img;
    tooth_47.current.src = plag_tooth47_img;
    tooth_48.current.src = plag_tooth48_img;
   }
   if(proceduresData!==undefined){
  proceduresData.forEach((el)=>{
   if(el.proc_name.includes("Адгезивна реставрація переднього зуба") || el.proc_name.includes("Адгезивна
реставрація бічного зуба")){
    if(el.partname!==null){
    el.partname.includes("Різець 11") ? tooth_11.current.src = fill_tooth11_img : tooth_11.current.src =
tooth11_img;
    el.partname.includes("Різець 12") ? tooth_12.current.src = fill_tooth12_img : tooth_12.current.src =
tooth12_img;
    el.partname.includes("Клик 13") ? tooth_13.current.src = fill_tooth13_img : tooth_13.current.src =
tooth13_img;
    el.partname.includes("Премоляр 14") ? tooth_14.current.src = fill_tooth14_img : tooth_14.current.src =
tooth14_img;
    el.partname.includes("Премоляр 15") ? tooth_15.current.src = fill_tooth15_img : tooth_15.current.src =
tooth15_img;
    el.partname.includes("Моляр 16") ? tooth_16.current.src = fill_tooth16_img : tooth_16.current.src =
tooth16_img;
    el.partname.includes("Моляр 17") ? tooth_17.current.src = fill_tooth17_img : tooth_17.current.src =
tooth17_img;
    el.partname.includes("Моляр  18") ? tooth_18.current.src = fill_tooth18_img : tooth_18.current.src =
tooth18_img;
    el.partname.includes("Різець 21") ? tooth_21.current.src = fill_tooth21_img : tooth_21.current.src =
tooth21_img;
    el.partname.includes("Різець 22") ? tooth_22.current.src = fill_tooth22_img : tooth_22.current.src =
tooth22_img;
    el.partname.includes("Клик 23") ? tooth_23.current.src = fill_tooth23_img : tooth_23.current.src =
tooth23_img;
    el.partname.includes("Премоляр 24") ? tooth_24.current.src = fill_tooth24_img : tooth_24.current.src =
tooth24_img;
    el.partname.includes("Премоляр 25") ? tooth_25.current.src = fill_tooth25_img : tooth_25.current.src =
tooth25_img;
    el.partname.includes("Моляр 26") ? tooth_26.current.src = fill_tooth26_img : tooth_26.current.src =
tooth26_img;
```

```
    el.partname.includes("Моляр 27") ? tooth_27.current.src = fill_tooth27_img : tooth_27.current.src =
tooth27_img;
    el.partname.includes("Моляр 28") ? tooth_28.current.src = fill_tooth28_img : tooth_28.current.src =
tooth28_img;
    el.partname.includes("Різець 31") ? tooth_31.current.src = fill_tooth31_img : tooth_31.current.src =
tooth31_img;
    el.partname.includes("Різець 32") ? tooth_32.current.src = fill_tooth32_img : tooth_32.current.src =
tooth32_img;
    el.partname.includes("Клик 33") ? tooth_33.current.src = fill_tooth33_img : tooth_33.current.src =
tooth33_img;
    el.partname.includes("Премоляр 34") ? tooth_34.current.src = fill_tooth34_img : tooth_34.current.src =
tooth34_img;
    el.partname.includes("Премоляр 35") ? tooth_35.current.src = fill_tooth35_img : tooth_35.current.src =
tooth35_img;
    el.partname.includes("Моляр 36") ? tooth_36.current.src = fill_tooth36_img : tooth_36.current.src =
tooth36_img;
    el.partname.includes("Моляр 37") ? tooth_37.current.src = fill_tooth37_img : tooth_37.current.src =
tooth37_img;
    el.partname.includes("Моляр 38") ? tooth_38.current.src = fill_tooth38_img : tooth_38.current.src =
tooth38_img;
    el.partname.includes("Різець 41") ? tooth_41.current.src = fill_tooth41_img : tooth_41.current.src =
tooth41_img;
    el.partname.includes("Різець 42") ? tooth_42.current.src = fill_tooth42_img : tooth_42.current.src =
tooth42_img;
    el.partname.includes("Клик 43") ? tooth_43.current.src = fill_tooth43_img : tooth_43.current.src =
tooth43_img;
    el.partname.includes("Премоляр 44") ? tooth_44.current.src = fill_tooth44_img : tooth_44.current.src =
tooth44_img;
    el.partname.includes("Премоляр 45") ? tooth_45.current.src = fill_tooth45_img : tooth_45.current.src =
tooth45_img;
    el.partname.includes("Моляр 46") ? tooth_46.current.src = fill_tooth46_img : tooth_46.current.src =
tooth46_img;
    el.partname.includes("Моляр 47") ? tooth_47.current.src = fill_tooth47_img : tooth_47.current.src =
tooth47_img;
    el.partname.includes("Моляр 48") ? tooth_48.current.src = fill_tooth48_img : tooth_48.current.src =
tooth48_img;
    }
   }
   if(el.proc_name.includes("Екстірпація пульпи або хірургічнаобробка кореневого каналу")){
    if(el.partname!==null){
    el.partname.includes("Різець 11") ? tooth_11.current.src = nerv_tooth11_img : tooth_11.current.src =
tooth11_img;
    el.partname.includes("Різець 12") ? tooth_12.current.src = nerv_tooth12_img : tooth_12.current.src =
tooth12_img;
    el.partname.includes("Клик 13") ? tooth_13.current.src = nerv_tooth13_img : tooth_13.current.src =
tooth13_img;
    el.partname.includes("Премоляр 14") ? tooth_14.current.src = nerv_tooth14_img : tooth_14.current.src =
tooth14_img;
    el.partname.includes("Премоляр 15") ? tooth_15.current.src = nerv_tooth15_img : tooth_15.current.src =
tooth15_img;
    el.partname.includes("Моляр 16") ? tooth_16.current.src = nerv_tooth16_img : tooth_16.current.src =
tooth16_img;
    el.partname.includes("Моляр 17") ? tooth_17.current.src = nerv_tooth17_img : tooth_17.current.src =
tooth17_img;
    el.partname.includes("Моляр 18") ? tooth_18.current.src = nerv_tooth18_img : tooth_18.current.src =
tooth18_img;
    el.partname.includes("Різець 21") ? tooth_21.current.src = nerv_tooth21_img : tooth_21.current.src =
tooth21_img;
    el.partname.includes("Різець 22") ? tooth_22.current.src = nerv_tooth22_img : tooth_22.current.src =
tooth22_img;
    el.partname.includes("Клик 23") ? tooth_23.current.src = nerv_tooth23_img : tooth_23.current.src =
tooth23_img;
```

```
    el.partname.includes("Премоляр 24") ? tooth_24.current.src = nerv_tooth24_img : tooth_24.current.src =
tooth24_img;
    el.partname.includes("Премоляр 25") ? tooth_25.current.src = nerv_tooth25_img : tooth_25.current.src =
tooth25_img;
    el.partname.includes("Моляр 26") ? tooth_26.current.src = nerv_tooth26_img : tooth_26.current.src =
tooth26_img;
    el.partname.includes("Моляр 27") ? tooth_27.current.src = nerv_tooth27_img : tooth_27.current.src =
tooth27_img;
    el.partname.includes("Моляр 28") ? tooth_28.current.src = nerv_tooth28_img : tooth_28.current.src =
tooth28_img;
    el.partname.includes("Різець 31") ? tooth_31.current.src = nerv_tooth31_img : tooth_31.current.src =
tooth31_img;
    el.partname.includes("Різець 32") ? tooth_32.current.src = nerv_tooth32_img : tooth_32.current.src =
tooth32_img;
    el.partname.includes("Клик 33") ? tooth_33.current.src = nerv_tooth33_img : tooth_33.current.src =
tooth33_img;
    el.partname.includes("Премоляр 34") ? tooth_34.current.src = nerv_tooth34_img : tooth_34.current.src =
tooth34_img;
    el.partname.includes("Премоляр 35") ? tooth_35.current.src = nerv_tooth35_img : tooth_35.current.src =
tooth35_img;
    el.partname.includes("Моляр 36") ? tooth_36.current.src = nerv_tooth36_img : tooth_36.current.src =
tooth36_img;
    el.partname.includes("Моляр 37") ? tooth_37.current.src = nerv_tooth37_img : tooth_37.current.src =
tooth37_img;
    el.partname.includes("Моляр 38") ? tooth_38.current.src = nerv_tooth38_img : tooth_38.current.src =
tooth38_img;
    el.partname.includes("Різець 41") ? tooth_41.current.src = nerv_tooth41_img : tooth_41.current.src =
tooth41_img;
    el.partname.includes("Різець 42") ? tooth_42.current.src = nerv_tooth42_img : tooth_42.current.src =
tooth42_img;
    el.partname.includes("Клик 43") ? tooth_43.current.src = nerv_tooth43_img : tooth_43.current.src =
tooth43_img;
    el.partname.includes("Премоляр 44") ? tooth_44.current.src = nerv_tooth44_img : tooth_44.current.src =
tooth44_img;
    el.partname.includes("Премоляр 45") ? tooth_45.current.src = nerv_tooth45_img : tooth_45.current.src =
tooth45_img;
    el.partname.includes("Моляр 46") ? tooth_46.current.src = nerv_tooth46_img : tooth_46.current.src =
tooth46_img;
    el.partname.includes("Моляр 47") ? tooth_47.current.src = nerv_tooth47_img : tooth_47.current.src =
tooth47_img;
    el.partname.includes("Моляр 48") ? tooth_48.current.src = nerv_tooth48_img : tooth_48.current.src =
tooth48_img;
    }}
  if(el.proc_name.includes("Видалення зуба або його частин") || el.proc_name.includes("Секційне видалення
зуба або його частин(-и)") || el.proc_name.includes("видалення зуба")){
    if(el.partname!==null){
    el.partname.includes("Різець 11") ? tooth_11.current.style.visibility = "hidden" : tooth_11.current.style.visibility
= "visible";
    el.partname.includes("Різець 12") ? tooth_12.current.style.visibility = "hidden" : tooth_12.current.style.visibility
= "visible";
    el.partname.includes("Клик 13") ? tooth_13.current.style.visibility = "hidden" : tooth_13.current.style.visibility
= "visible";
    el.partname.includes("Премоляр 14") ? tooth_14.current.style.visibility = "hidden" :
tooth_14.current.style.visibility = "visible";
    el.partname.includes("Премоляр 15") ? tooth_15.current.style.visibility = "hidden" :
tooth_15.current.style.visibility = "visible";
    el.partname.includes("Моляр 16") ? tooth_16.current.style.visibility = "hidden" :
tooth_16.current.style.visibility = "visible";
    el.partname.includes("Моляр 17") ? tooth_17.current.style.visibility = "hidden" :
tooth_17.current.style.visibility = "visible";
    el.partname.includes("Моляр  18") ? tooth_18.current.style.visibility = "hidden" :
tooth_18.current.style.visibility = "visible";
```

```
    el.partname.includes("Різець 21") ? tooth_21.current.style.visibility = "hidden" : tooth_21.current.style.visibility
= "visible";
    el.partname.includes("Різець 22") ? tooth_22.current.style.visibility = "hidden" : tooth_22.current.style.visibility
= "visible";
    el.partname.includes("Клик 23") ? tooth_23.current.style.visibility = "hidden" : tooth_23.current.style.visibility
= "visible";
    el.partname.includes("Премоляр 24") ? tooth_24.current.style.visibility = "hidden" :
tooth_24.current.style.visibility = "visible";
    el.partname.includes("Премоляр 25") ? tooth_25.current.style.visibility = "hidden" :
tooth_25.current.style.visibility = "visible";
    el.partname.includes("Моляр 26") ? tooth_26.current.style.visibility = "hidden" :
tooth_26.current.style.visibility = "visible";
    el.partname.includes("Моляр 27") ? tooth_27.current.style.visibility = "hidden" :
tooth_27.current.style.visibility = "visible";
    el.partname.includes("Моляр 28") ? tooth_28.current.style.visibility = "hidden" :
tooth_28.current.style.visibility = "visible";
    el.partname.includes("Різець 31") ? tooth_31.current.style.visibility = "hidden" : tooth_31.current.style.visibility
= "visible";
    el.partname.includes("Різець 32") ? tooth_32.current.style.visibility = "hidden" : tooth_32.current.style.visibility
= "visible";
    el.partname.includes("Клик 33") ? tooth_33.current.style.visibility = "hidden" : tooth_33.current.style.visibility
= "visible";
    el.partname.includes("Премоляр 34") ? tooth_34.current.style.visibility = "hidden" :
tooth_34.current.style.visibility = "visible";
    el.partname.includes("Премоляр 35") ? tooth_35.current.style.visibility = "hidden" :
tooth_35.current.style.visibility = "visible";
    el.partname.includes("Моляр 36") ? tooth_36.current.style.visibility = "hidden" :
tooth_36.current.style.visibility = "visible";
    el.partname.includes("Моляр 37") ? tooth_37.current.style.visibility = "hidden" :
tooth_37.current.style.visibility = "visible";
    el.partname.includes("Моляр 38") ? tooth_38.current.style.visibility = "hidden" :
tooth_38.current.style.visibility = "visible";
    el.partname.includes("Різець 41") ? tooth_41.current.style.visibility = "hidden" : tooth_41.current.style.visibility
= "visible";
    el.partname.includes("Різець 42") ? tooth_42.current.style.visibility = "hidden" : tooth_42.current.style.visibility
= "visible";
    el.partname.includes("Клик 43") ? tooth_43.current.style.visibility = "hidden" : tooth_43.current.style.visibility
= "visible";
    el.partname.includes("Премоляр 44") ? tooth_44.current.style.visibility = "hidden" :
tooth_44.current.style.visibility = "visible";
    el.partname.includes("Премоляр 45") ? tooth_45.current.style.visibility = "hidden" :
tooth_45.current.style.visibility = "visible";
    el.partname.includes("Моляр 46") ? tooth_46.current.style.visibility = "hidden" :
tooth_46.current.style.visibility = "visible";
    el.partname.includes("Моляр 47") ? tooth_47.current.style.visibility = "hidden" :
tooth_47.current.style.visibility = "visible";
    el.partname.includes("Моляр 48") ? tooth_48.current.style.visibility = "hidden" :
tooth_48.current.style.visibility = "visible";
   }
  }
   })
  }}
oral_cavity_parts = ocpData.map(({ partname }) => ({ ['label']: partname , ['value']: partname }));
procedures= dirProceduresData.map(({ proc_name }) => ({ ['label']: proc_name , ['value']: proc_name }));
}
const [show, setShow] = useState(false);
const [showShareBoard, setShowShareBoard] = useState(false);
const [filterNumber, setFilter] = useState(1);
const [filterOn, setFilterOn] = useState(0);
const [procedureAfterFilter, setprocedureAfterFilter]= useState([]);
const showSharing = () =>{
  setShowShareBoard(current => !current);
```

```
}
const showFilter = () =>{
  setShow(current => !current);
 }
const showTreatments = async()=>{
  setFilter(1);
}
const showProcedures = async()=>{
  setFilter(3);
}
const showRecords = async()=>{
  setFilter(2);
}
//реалізація пошук даних за фільтром
const acceptFilter = async(e) =>{
  procedureAfterFilter.splice(0,procedureAfterFilter.length);
  e.preventDefault();
  let procedF = JSON.parse(JSON.stringify(selected));
  let ocpF=JSON.parse(JSON.stringify(selected2));
  let dateStr = new Date(JSON.stringify(dateStFilter));
  let dateEnd = new Date(JSON.stringify(dateEnFilter));
  if( checkbox.current.checked){
     recordsData.forEach((el)=>{
       let dateRec = new Date(el.record_date);
       if(dateRec>dateStr && dateRec <dateEnd){
         procedureAfterFilter.push(el);
       }
     })
     setFilter(2);
     setFilterOn(1);
  }else{
    proceduresData.forEach((el)=>{
      if((Object.keys(procedF)).length>0){
        procedF.forEach((element)=>{
          if(element.value==el.proc_name){
            if((Object.keys(ocpF)).length>0){
              ocpF.forEach((ocp)=>{
                if(el.partname.includes(ocp.value)){
                  procedureAfterFilter.push(el);
                }
              })
            }else{
              procedureAfterFilter.push(el);
            }
          }else{
            if((Object.keys(ocpF)).length>0){
              ocpF.forEach((ocp)=>{
                if(el.partname.includes(ocp.value)){
                  procedureAfterFilter.push(el);
                }
              })
            }
          }
        });
      }else if((Object.keys(ocpF)).length>0){
        ocpF.forEach((ocp)=>{
          if(el.partname.includes(ocp.value)){
            if((Object.keys(procedF)).length>0){
              procedF.forEach((element)=>{
                if(element.value==el.proc_name){
                  procedureAfterFilter.push(el);
                }
```

```
            })
          }else{
            procedureAfterFilter.push(el);
          }
        }else{
          if((Object.keys(procedF)).length>0){
            procedF.forEach((element)=>{
              if(element.value==el.proc_name){
                procedureAfterFilter.push(el);
              }
            })
          }
        }
      })
    }
   });
   setFilter(3);
   setFilterOn(1);
 }}
//перемикання між типами записів
const  rightFilter= async()=>{
 if(filterNumber==1){
   setFilter(2);
 }else if(filterNumber==2){
   setFilter(3);
 }else if(filterNumber==3){
   setFilter(1);
 }}
const leftFilter = async()=>{
 if(filterNumber==1){
   setFilter(3);
 }else if(filterNumber==2){
   setFilter(1);
 }else if(filterNumber==3){
   setFilter(2);
 }}
const [doctorsData, setDoctors]= useState([]);
const [doctorsLoading, setdoctorsLoading]= useState(false);
const [doctorsErr, setDoctorsErr] = useState('');
//реалізація пошуку лікаря
const GetDoctors = async()=>{
 setdoctorsLoading(true);
   try {
     const response = await
fetch("http://localhost:8800/users/doctors?username="+inputs.username+"&phone="+inputs.phone, {
       method: 'GET',
     });
     if (!response.ok) {
       throw new Error(`Error! status: ${response.status}`);
     }
     const result = await response.json();
     setDoctors(result);
   } catch (err) {
     setDoctorsErr(err.message);
   } finally {
     setdoctorsLoading(false);
   }
}
 return (
   <>
   <div className='mouth-board'>
     <div className='mouth-board__left-wrap'>
```

```
  <img className='mouth-board__mouth'src={mouth_img}/>
  <img className='mouth__tooth_11 mouth__tooth' ref={tooth_11} src={tooth11_img}/>
  <img className='mouth__tooth_12 mouth__tooth' ref={tooth_12} src={tooth12_img}/>
  <img className='mouth__tooth_13 mouth__tooth' ref={tooth_13} src={tooth13_img}/>
  <img className='mouth__tooth_14 mouth__tooth' ref={tooth_14} src={tooth14_img}/>
  <img className='mouth__tooth_15 mouth__tooth' ref={tooth_15} src={tooth15_img}/>
  <img className='mouth__tooth_16 mouth__tooth' ref={tooth_16} src={tooth16_img}/>
  <img className='mouth__tooth_17 mouth__tooth' ref={tooth_17} src={tooth17_img}/>
  <img className='mouth__tooth_18 mouth__tooth' ref={tooth_18} src={tooth18_img}/>
  <img className='mouth__tooth_21 mouth__tooth' ref={tooth_21} src={tooth21_img}/>
  <img className='mouth__tooth_22 mouth__tooth' ref={tooth_22} src={tooth22_img}/>
  <img className='mouth__tooth_23 mouth__tooth' ref={tooth_23} src={tooth23_img}/>
  <img className='mouth__tooth_24 mouth__tooth' ref={tooth_24} src={tooth24_img}/>
  <img className='mouth__tooth_25 mouth__tooth' ref={tooth_25} src={tooth25_img}/>
  <img className='mouth__tooth_26 mouth__tooth' ref={tooth_26} src={tooth26_img}/>
  <img className='mouth__tooth_27 mouth__tooth' ref={tooth_27} src={tooth27_img}/>
  <img className='mouth__tooth_28 mouth__tooth' ref={tooth_28} src={tooth28_img}/>
  <img className='mouth__tooth_31 mouth__tooth' ref={tooth_31} src={tooth31_img}/>
  <img className='mouth__tooth_32 mouth__tooth' ref={tooth_32} src={tooth32_img}/>
  <img className='mouth__tooth_33 mouth__tooth' ref={tooth_33} src={tooth33_img}/>
  <img className='mouth__tooth_34 mouth__tooth' ref={tooth_34} src={tooth34_img}/>
  <img className='mouth__tooth_35 mouth__tooth' ref={tooth_35} src={tooth35_img}/>
  <img className='mouth__tooth_36 mouth__tooth' ref={tooth_36} src={tooth36_img}/>
  <img className='mouth__tooth_37 mouth__tooth' ref={tooth_37} src={tooth37_img}/>
  <img className='mouth__tooth_38 mouth__tooth' ref={tooth_38} src={tooth38_img}/>
  <img className='mouth__tooth_41 mouth__tooth' ref={tooth_41} src={tooth41_img}/>
  <img className='mouth__tooth_42 mouth__tooth' ref={tooth_42} src={tooth42_img}/>
  <img className='mouth__tooth_43 mouth__tooth' ref={tooth_43} src={tooth43_img}/>
  <img className='mouth__tooth_44 mouth__tooth' ref={tooth_44} src={tooth44_img}/>
  <img className='mouth__tooth_45 mouth__tooth' ref={tooth_45} src={tooth45_img}/>
  <img className='mouth__tooth_46 mouth__tooth' ref={tooth_46} src={tooth46_img}/>
  <img className='mouth__tooth_47 mouth__tooth' ref={tooth_47} src={tooth47_img}/>
  <img className='mouth__tooth_48 mouth__tooth' ref={tooth_48} src={tooth48_img}/>
</div>
<div className='mouth-board__right-wrap'>
  <div className='mouth-board__card'>
    <div className='mouth-board__text-wrap_left'>
      <p className='mouth-board__text'>Кількість зубів:</p>
    </div>
    <div className='mouth-board__text-wrap_right'>
      <p className='mouth-board__text-days_data number_teeth'>{teethNumber}</p>
    </div>
  </div>
  <div className='mouth-board__card'>
    <div className='mouth-board__text-wrap_left'>
      <p className='mouth-board__text'>Час з останнього відвідування стоматолога:</p>
    </div>
    <div className='mouth-board__text-wrap_right'>
      <p className='mouth-board__text-days_data last_appointment'>{diffDays}</p>
      <p className='mouth-board__text-days'>днів</p>
    </div>
  </div>
  <div className='mouth-board__card'>
    <div className='mouth-board__text-wrap_left'>
      <p className='mouth-board__text'>Час з останньої професійної чистки:</p>
    </div>
    <div className='mouth-board__text-wrap_right'>
      <p className='mouth-board__text-days_data last_cleaning'>{diffInProBrushing}</p>
      <p className='mouth-board__text-days'>днів</p>
    </div>
  </div>
  <div className='mouth-board__card'>
```

```
      <div className='mouth-board__text-wrap_left'>
        <p className='mouth-board__text'>Час з останньої заміни зубної щітки:</p>
      </div>
      <div className='mouth-board__text-wrap_right'>
        <p className='mouth-board__text-days_data last_brushchange'>{brushChange}</p>
        <p className='mouth-board__text-days'>днів</p>
      </div>
    </div>
    <div className='mouth-board__card'>
      <div className='mouth-board__text-wrap_left'>
        <p className='mouth-board__text'>Сьогоднішня кількість разів чистки зубів:</p>
      </div>
      <div className='mouth-board__text-wrap_right'>
        <p className='mouth-board__text-days_data count_today_brushing'>{brushing}</p>
      </div>
    </div>
  </div>
</div>
<div className='diary'>
  <div className='diary__filters-wrap'>
    <button className='filters__buttn_treatment filters__buttn' onClick={showTreatments} >
      <p className='filters__buttn-text' >ЛІКУВАННЯ</p>
    </button>
    <button className='filters__buttn_records filters__buttn' onClick={showRecords}>
      <p className='filters__buttn-text' >ЗАПИСИ</p>
    </button>
    <button className='filters__buttn_procedures filters__buttn' onClick={showProcedures}>
      <p className='filters__buttn-text' >ПРОЦЕДУРИ</p>
    </button>
  </div>
  <div className='diary__nav-panel'>
    <div className='nav-panel__arrows'>
      <button className='buttons__left' onClick={leftFilter}></button>
      <button className='buttons__right'onClick={rightFilter}></button>
    </div>
    <div className='nav-panel__button-wrap'>
      <Link to="/treatment">
        <button className=' nav-panel__button nav-panel__button_primary button__add-treatment'>
          <p className='nav-panel__button_text'>Нове лікування</p>
        </button>
      </Link>
      <button className='nav-panel__button nav-panel__button_secondary button__share'>
        <p className='nav-panel__button_text' onClick={showSharing}>Поділитися</p>
      </button>
      <button className='button__filter' onClick={showFilter}></button>
    </div>
  </div>
  <div className='diary__records'>
    <div className='diary__records-wrap'>
    {treatmentError
    ? "Щось пішло не так!"
    : treatmentLoading
    ? "завантаження..."
    : filterNumber===1
    ? treatmentData.map((treatments)=> <TreatmentCard treatment={treatments} key={treatments.idtreatment}/>)
    : ""
    }
    {recordsError
    ? "Щось пішло не так!"
    : recordsLoading
    ? "завантаження..."
    : filterNumber===2
```

```
      ? filterOn===1
      ? procedureAfterFilter.map((records) => <RecordCard teeth_record={records} key={records.idrecord}/>)
      : recordsData.map((records) => <RecordCard teeth_record={records} key={records.idrecord}/>)
      : ""
      }
      {proceduresError
      ? "Щось пішло не так!"
      : proceduresLoading
      ? "завантаження..."
      : filterNumber===3
      ? filterOn===1
      ? procedureAfterFilter.map((procedures) => <ProcedureCard procedure={procedures}
key={procedures.idprocedure}/>)
      : proceduresData.map((procedures) => <ProcedureCard procedure={procedures}
key={procedures.idprocedure}/>)
      : ""
      }
      </div>
    </div>
  </div>
  <div className='sharing-board' style={{display: showShareBoard ? 'block' : 'none'}}>
    <div className='sharing-board__wrap'>
      <p className='sharing-board__text_header'>Введіть дані для пошуку</p>
      <div className='sharing-board__input-wrap'>
        <p className='sharing-board__text_title'>Прізвище Ім'я Побатькові:</p>
        <input type="text" className='sharing-board__input_name' name="username"
onChange={handleChange}/>
      </div>
      <div className='sharing-board__input-wrap'>
        <p className='sharing-board__text_title'>Номер телефону:</p>
        <input type="text" className='sharing-board__input_name' name="phone" onChange={handleChange}/>
      </div>
      <div className='sharing-board__input-wrap_buttons'>
        <button className='sharing-board__buttn_search' onClick={GetDoctors}>Пошук</button>
      </div>
      <div className='result-box'>
        <p className='result-box__text_header'>Результат пошуку:</p>
      {doctorsData.length>0
      ? doctorsData.map((results)=> <ResultBox result={results} key={results.iduser}/>)
      : "Не знайдено відповідностей"
      }
      </div>
    </div>
  </div>
  <div className='filter-board' rex={filter_box} style={{display: show ? 'block' : 'none'}} >
    <div className='filter-board__wrap'>
      <div className='filter-board__header-wrap'>
        <input type='checkbox' className='filter-board__check-date' ref={checkbox}/>
        <p className='filter-board__text_title'>Дата:</p>
        <div className='filter-board__date-wrap'>
        <p className='filter-board__text_title'>з</p>
        <input type="date" className='filter-board__date_start filter-board__input-date' name="date"
value={dateStFilter} onChange={(e)=>setDateStFilter(e.target.value)}/>
        <p className='filter-board__text_title'>до</p>
        <input type="date" className='filter-board__date_end filter-board__input-date' value={dateEnFilter}
onChange={(e)=>setDateEnFilter(e.target.value)} />
      </div>
      <div className='filter-board__select-wrap'>
        <p className='filter-board__text_title'>Процедура:</p>
          <MultiSelect
            className='filter-board__select'
```

```
            options={procedures}
            value = {selected}
            onChange={setSelected}
            labelledBy='Select'
          />
        </div>
        <div className='filter-board__select-wrap'>
          <p className='filter-board__text_title'>Область застосування:</p>
          <MultiSelect
            className='filter-board__select'
            options={oral_cavity_parts}
            value = {selected2}
            onChange={setSelected2}
            labelledBy='Select'
          />
        </div>
        <div className='filter-board__button-wrap'>
          <button className='filter-board__button' onClick={acceptFilter}>
            <p className='filter-board__buttn-text'>Застосувати</p>
          </button>
        </div>
      </div>
    </div>
    </>
  )
}
export default PatientDiary;
```

_variables.scss// файл scss змінних

```
//змінні характкристик розмірів екрану
$width-main: 1788;
$width-medium: 1081;
$width-mini: 760;
$height-main: calc(($width-main*9)/16);
$height-medium: calc(($width-medium*9)/16);
$height-mini: calc(($width-mini*9)/16);
//змінні всіх кольорів
$main_color: #53c8ed;
$form_back_color: #b7b9b9;
$form_input_color: #D9D9D9;
$pressed_buttton-color: #cdfcff;
$font_color: #475ead;
$trtmnt_bckgr: #ceffcd;
$record_bcrgr: #feffcd;
$procedure_bcrgr: #ffcddf;
$appointment_color: #82ef80;
$black_color: #000000;
$background: #FFFFFF;
$error_color: #F11320;
```

_function.scss // файл scss функцій

```
@import "variables";
//функція для визначення розмірів
@function find-size($val1, $val2, $val3){
   @if $val1==0{
      @return 0;
   }@else{
      @return calc(($val1*$val2)/$val3)
   }
};
//функція для зазначення радіуса рамки
@function border_rad($pxl){
```

```
    @return calc(($pxl*1rem)/16);
};
// mixin для задання висоти і ширини елементу
@mixin size($w, $h, $w_res: $width-main, $h_res: $height-main){
    width: find-size($w,100vw, $w_res);
    height: find-size($h,100vh,$h_res);
}
// mixin для задання висоти елементу
@mixin height($h, $h_res: $height-main ){
    height: find-size($h,100vh,$h_res);
}
// mixin для задання ширини елементу
@mixin width($w, $w_res: $width-main){
    width: find-size($w,100vw, $w_res);
}
// mixin для задання властивості margin
@mixin margin($t, $r, $b, $l, $w_res: $width-main, $h_res: $height-main){
    margin-top: find-size($t,100vh,$h_res);
    margin-right: find-size($r,100vw, $w_res);
    margin-bottom: find-size($b,100vh,$h_res);
    margin-left: find-size($l,100vw, $w_res);
}
// mixin для задання властивості padding
@mixin padding($t, $r, $b, $l, $w_res: $width-main, $h_res: $height-main){
    padding-top: find-size($t,100vh,$h_res);
    padding-right: find-size($r,100vw, $w_res);
    padding-bottom: find-size($b,100vh,$h_res);
    padding-left: find-size($l,100vw, $w_res);
}
// mixin для задання властивості box-shadow
@mixin shadow($t, $r, $b, $l,$color, $w_res: $width-main, $h_res: $height-main){
    box-shadow: find-size($t,100vh,$h_res) find-size($r,100vw, $w_res) find-size($b,100vh,$h_res) find-
size($l,100vw, $w_res) $color;
}
// mixin для визначення позиції елемента відосно верхнього і лівого країв
@mixin position_t_l($t, $l, $w_res: $width-main, $h_res: $height-main){
    top: find-size($t,100vh,$h_res);
    left: find-size($l,100vw, $w_res);
}
// mixin для визначення позиції елемента відосно нижнього і лівого країв
@mixin position_l_b( $l, $b, $w_res: $width-main, $h_res: $height-main){
    left: find-size($l,100vw, $w_res);
    bottom: find-size($b,100vh,$h_res);
}
// mixin для визначення позиції елемента відосно нижнього і правого країв
@mixin position_r_b( $r, $b, $w_res: $width-main, $h_res: $height-main){
    right: find-size($r,100vw, $w_res);
    bottom: find-size($b,100vh,$h_res);
}
// mixin для визначення позиції елемента відосно верхнього і правого країв
@mixin position_t_r($t, $r, $w_res: $width-main, $h_res: $height-main){
    top: find-size($t,100vh,$h_res);
    right: find-size($r,100vw, $w_res);
}
// mixin для зазначення розміру шрифтів
@mixin font_size($size, $line_height, $h_resolution: $height-main){
    font-size: find-size($size,100vh,$h_resolution);
    line-height: find-size($line_height,100vh,$h_resolution);
}
```

procedure.scss // файл стилів картки процедури

```
@import "../../style/fonts";
```

```
@import "../../style/variables";
@import "../../style/function";
.procedure-card{
   position: relative;
   @include margin(44,0,44,0);
   overflow: hidden;
   @include width(1510);
   background-color: $procedure_bcrgr;
   border-radius: border_rad(34);
   display: grid;
   grid-template-columns: 45% 45% 10%;
   grid-template-rows: 60% 40%;
   @include padding(46,0,27,0);
   &__left-wrap{z-index: 12; }
   &__buttns-wrap{ display:flex;
                  flex-direction: column;
                  align-items: flex-end;
                  z-index: 12;}
   .buttn{ &_delete{
         background-size: find-size(50,100vw, $width-main) find-size(50,100vh, $height-main);
         cursor: pointer;}
         &_edit{ background-size: find-size(50,100vw, $width-main) find-size(50,100vh, $height-main);
         cursor: pointer;}}
   &__text{ &-title{ @include font_size(37,50);
                     color: $black_color;
                     font-weight: 400;
                     font-family: OpenSans;
                     @include margin(0,0,0,113);
                     z-index: 12;}
      &-data{@include margin(0,0,14,0);
            @include font_size(48,72);
            color: $main_color;
            font-weight: 700;
            font-family: OpenSans;
            z-index: 12;}}
   &__feather_first{@include size(490,490);
      position: absolute;
      z-index: 10;
      @include position_l_b(-138,-48);}
   &__feather_second{ @include size(490,490);
      position: absolute;
      z-index: 10;
      transform: rotate(110.28deg);
      @include position_t_l(-113, 1090);
   }
   &__file-img{@include size(119,119);
      background-image: url("../../images/icons/svges/drive_file_insert_icon.svg");
      background-repeat: no-repeat;
      background-size: find-size(119,100vw, $width-main) find-size(119,100vh, $height-main);
      background-color: transparent;
      cursor: pointer;
      z-index: 12;
      display: none;}}
```

procedureCard.jsx // файл модуля картки процедури

```
//підключення модулів
import React ,{ useRef }from 'react'
import file_img from "../../images/icons/svges/drive_file_insert_icon.svg"
import  feather_img from "../../images/icons/svges/wing_feather_angel_heaven_bird_icon.svg"
import { useMutation, queryClient } from 'react-query';
import { makeRequest } from '../../axios';
```

```
import { useNavigate } from "react-router-dom";
//модуль картки процедури
const ProcedureCard = ({procedure}) => {
   const navigate=useNavigate();
   //обробка видалення
   const deleteMutation = useMutation(
      (procId) => {return makeRequest.delete("/procedures/api/" + procId); },
      {onSuccess: () => {queryClient.invalidateQueries(["procedures"]);},});
   const deleteProcedure = ()=>{
      let answer = window.confirm('Ви впевнені, що хочете видалити цей запис і що з ним пов\'язано?')
      if (answer){
         deleteMutation.mutate(procedure.idprocedure);
         window.location.reload(false);
      }};
      // реалізація відкриття файлу
   const openProcFile =(event)=>{
    localStorage.setItem("procedure",JSON.stringify({
      idprocedure: procedure.idprocedure,
      partname: procedure.partname,
      proc_name: procedure.proc_name,
      id_or_cav_part: procedure.id_or_cav_part
    }) );
    navigate("/procedure_file");}
    //обробка оновлення
   const updateProcedure=()=>{
    localStorage.setItem("procedure",JSON.stringify({
      idprocedure: procedure.idprocedure,
      partname: procedure.partname,
      proc_name: procedure.proc_name,
      id_or_cav_part: procedure.id_or_cav_part
    }) );
    navigate("/procedure")}
   const buttFileRef = useRef(null);
   if(buttFileRef.current!==null){
    if(procedure.file_name!==null){
      buttFileRef.current.style.display="block"
    }else{
      buttFileRef.current.style.display="none"
    } }
   return (
      <div className='procedure-card'>
         <img className='procedure-card__feather_first' src={feather_img}/>
         <img className='procedure-card__feather_second' src={feather_img}/>
         <div className="procedure-card__left-wrap">
            <p className='procedure-card__text-title'>Назва:</p>
         </div>
         <p className='procedure-card__text-data_name procedure-card__text-data'>{procedure.proc_name}</p>

         <div className='procedure-card__buttns-wrap'>
            <button className='buttn_delete' onClick={deleteProcedure}></button>
            <button className='buttn_edit' onClick={updateProcedure}></button>
         </div>
         <div className="procedure-card__left-wrap">
            <p className='procedure-card__text-title'>Область застосування</p>
         </div>
         <p className='procedure-card__text-data_name procedure-card__text-data'>{procedure.partname}</p>
         <button className='procedure-card__file-img' ref={buttFileRef} onClick={openProcFile} src={file_img}
></button>
    </div>)}
export default ProcedureCard
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему:

«Розробка веб-орієнтованого застосунку ведення дентальної хроніки для запису і аналізу стану ротової порожнини»

студентки групи 122-19-4 Михайленко Марії Олександрівни

Керівник економічного розділу

зав.каф. проф. д.е.н. ПЕП та ПУ **О.Г.Вагонова**

ДОДАТОК В

СИСТЕМНІ ДАНІ ДЛЯ БАЗИ ДАНИХ

Перелік системних процедур:

1. Підготовка тканин протезного ложа перед зняттям зліпків, одна обробка.

2. Встановлення шини з композиту, непряма процедура .

3. Встановлення шини з металу, непряма процедура .

4. Встановлення обтуратора.

5. Характеризація базису зубного протеза.

6. Зняття зліпка для відновлення зубного протеза.

7. Ідентифікація зубного протезу .

8. Металева вкладка для протезного зуба .

9. Надання хірургічного шаблона для імедіат-протеза .

10. Видалення нальоту або плям на зубах.

11. Відновлення раніше зробленої реставрації .

12. Видалення контрементів з поверхні зубів.

13. Мікроабразія емалі, один зуб .

14. Відбілювання, внутрішнє.

15. Відбілювання, зовнішнє .

16. Відбілювання, в домашніх умовах.

17. Стоматологічне бактеріологічне дослідження.

18. Стоматологічне дослідження і ідентифікація виділених культур .

19. Стоматологічний тест на чутливість.

20. Стоматологічне взяття зразків матеріалу неінвазивне для виявлення патологічних процесів.

21. Скринінг-тест слини.

22. Бактеріологічний скринінг-тест .

23. Стоматологічне гістопатологічне дослідження тканин .

24. Цитологічне (стоматологічне) дослідження.

25. Скринінг слизової оболонки.

26. Взяття зразка крові, для діагностики стоматологічного статусу.

27. Стоматологічне гематологічне обстеження .

28. Тестування пульпи.

29. Побудова діагностичної моделі .

30. Фотографічні знімки, внутрішньоротові.

31. Фотографічні знімки, позаротові .

32. Цефалометричний аналіз і інтерпретація результатів .

33. Прогностичний аналіз співвідношення розмірів зубів і щелеп .

34. Томограічний аналіз .

35. Електроміографічний аналіз.

36. Паліативні стоматологічні послуги .

37. Послуги екстреної стоматологічної допомоги у позаробочий час.

38. Виготовлення за індивідуальними параметрами пацієнта капи на нижню або верхню щелепу для можливості самостійного введення лікарських засобів.

39. Застосування лікарських препаратів/засобів у зв'язку із стоматологічними процедурами.

40. Встановлення внутрішньовенних канюль і проведення інфузії у зв'язку із стоматологічними процедурами.

41. Альвеолектомія, один сегмент.

42. Корекція (редукція), пов'язана з надлишково рухомим альвеолярним гребенем, один сегмент.

43. Видалення гіперплазованих тканин .

44. Перенесення місць прикріплення м'язів рота .

45. Лікування гострої періодонтальної інфекції.

46. Клінічний періодонтальний аналіз і фіксація результатів .

47. Вирівнювання поверхні коренів зубо з під'ясневим кюретажем.

48. Гінгівектомія.

49. Періодонтальна клаптева операція.

50. Операція з формування альвеолярного відростка .

51. Трансплантація альвеолярної кісткової тканини .

52. Трансплантація ясневої тканини.

53. Направлена тканинна регенерація .

54. Напралена тканинна регенерація, видалення мембрани .

55. Періодонтальна клаптева операція для подовження коронки зуба .

56. Резекція кореня зуба .

57. Трансплантація альвеолярних кісткових тканин, блок .

58. Хірургічна періодонтальна процедура, не класифікована в інших рубриках.

59. Нехірургічне періодонтальне лікування, не класифіковане в інших рубриках.

60. Реставрація зуба (із застосуванням металу), 1 поверхня, пряма .

61. Реставрація із застосуванням металу, 2 поверхні, пряма .

62. Реставрація із застосуванням металу, 3 поверхні, пряма .

63. Реставрація із застосуванням металу, 4 поверхні, пряма .

64. Реставрація із застосуванням металу, 5 поверхонь, пряма.

65. Реставрація із застосуванням металу, 1 повернхі, непряма .

66. Реставрація із застосуванням металу, 2 поверхні, непряма .

67. Реставрація із застосуванням металу, 3 поверхні, непряма .

68. Реставрація із застосуванням металу, 4 поверхні, непряма .

69. Реставрація із застосуванням металу, 5 поверхні, непряма .

70. Тимчасова реставрація зуба .

71. Фіксація металевого (ортодонтичного) кільця до зуба цементом .

72. Встановлення стоматологічного штифта .

73. Встановлення металевої коронки .

74. Покриття вістря зуба.

75. Реставрація кута різця переднього зуба .

76. Фіксація фрагмента зубаNULL Інші реставраційні стоматологічні послуги.

77. Фіксація вініра до поверхні зуба, непряме.

78. Видалення непрямого відновлення .

79. Повторне цементування непрямого відновлення .

80. Встановлення зубного штифта (відлитого, сформованого з металу).

81. Стоматологічні дієтичні поради .

82. Навчання гігієни порожнини рота .

83. Встановлення капи, непряма фіксація .

84. Бімаксилярна капа, непряма фіксація.

85. Фісура емалі і/або обтурація поверхні зуба через зуб.

86. Процедура для зниження чутливості зуба.

87. Одонтопластика.

88. Припасовка імпланта-абатмента.

89. Хірургічне видалення зубного імплантата та/або ретенційного пристрою.

90. Припасовка дуги зубного протеза .

91. Протез зі знімною копмпозитною основою, прикріпленою до імплантатів, за дугу.

92. Протез з фіксованим металевим каркасом, прикріпленим до імплантатів, за дугу.

93. Протез зі змінним металевим каркасом, прикріпленим до імплантатів, за дугу.

94. Видалення і заміна гвинта конструкції або абатменту.

95. Видалення і повторне приєднання протезу, зафіксованого до імплантата.

96. Повна коронка, приєднана до остеоінтегрованого імплантата, неметалева, непряма реставрація.

97. Повна коронка, приєднана до остеоінтегрованого імплатата, з фисеткою, непряма реставрація.

98. Повна коронка, приєднана до остеоінтегрованого імплантата, металева, непряма реставрація.

99. Застосування діагностичних шаблонів.

100. Застосування хірургічного шаблона для імплантатів.

101. Встановлення тимчасового імплатата.

102. Встановлення тимчасового фіксувального пристрою.

103. Корекція (налаштування), пов'язана із зубним протезом.

104. Перебазування повного знімного зубного протеза, обробленого.

105. Перебазування частково знімного зубного протеза, обробленого.

106. Ремоделювання повного знімного зубного протеза .

107. Ремоделювання часткового знімного зубного протеза .

108. Перебазування часткового знімного зубного протеза, пряме.

109. Перебазування повного знімного зубного протеза, пряме .

110. Чистка і полірування існуючого зубного протеза .

111. Модифікація базису зубного протеза.

112мПеріапікальний кюретаж.

113. Апікектомія.

114. Експлоративне хірургічне втручання у прикореневій зоні .

115. Пломбування апікальної частини кореневого каналу.

116. Хірургічна герматизація перфорації кореня зуба.

117. Хірургічне лікування і корекція у разі зовнішньої резорбції кореня зуба.

118. Гемісекція багатокореневого зуба.

119. Адгезивна реставрація переднього зуба, 1 поверхня, пряма.

120. Адгезивна реставрація переднього зуба, 2 поверхні, пряма .

121. Адгезивна реставрація переднього зуба, 3 поверхні, пряма .

122. Адгезивна реставрація переднього зуба, 4 поверхні, пряма .

123. Адгезивна реставрація бічного зуба, 1 поверхня, пряма.

124. Адгезивна реставрація бічного зуба, 2 поверхні, пряма .

125. Адгезивна реставрація бічного зуба, 3 поверхні, пряма .

126. Адгезивна реставрація бічного зуба, 4 поверхні, пряма .

127. Адгезивна реставрація бічного зуба, 5 поверхонь, пряма .

128. Реставрація, що імітує колір зуба, 1 поверхня, непряма.

129. Реставрація, що імітує колір зуба, 2 поверхні, непряма.

130. Реставрація, що імітує колір зуба, 3 поверхні, непряма.

131. Реставрація, що імітує колір зуба, 4 поверхні, непряма.

132. Місцеве застосування ремінералістичного агента.

133. Місцеве застосування каріостатичного агента.

134. Застосування концентрованого препарату ремінералізуючої дії .

135. Застосування концентрованого препарату каріостатичної дії .

136. Експлорація або проходження кальцифікованого кореневого каналу.

137. Видалення кореневої пломбиNULL Інші ендодонтичні послуги.

138. Видалення зацементованого штифту дл япломбування каналу зуба або коронки Річмонда .

139. Видалення або обхід уламків ендодонтичного інструменту.

140. Промивання і пломбування системи кореневого каналу .

141. Обтурація дефекту резорбітації або перфорації.

142. Тимчасове пломбування кореневого каналу .

143. Повторне цементування коронки або вініру .

144. Повторне цементування моста або шини.

145. Ребондінг моста або шини.

146. Видалення коронки.

147. Видалення моста або шини.

148. Відновлення коронки, моста або шини, непряме.

149. Відновлення коронки, моста або шини, пряме.

150. Стоматологічне лікування включно з видаленням або відновленням м'яких тканин, не класифіковане в інших рубриках .

151. Марсупіалізація кісти в ротовій порожнині.

152. Установка фіксаторів (ортодонтичних кілець) на частину зубного ряду.

153. Установка фіксаторів (ортодонтичних кілець) для кріплення апарату на весь зубний ряд.

154. Введення фіксованої піднебінної або лінгвальної (ортодонтичної) дуги.

155. Установка фіксаторів (ортодонтичних кілець) на частину зубного ряду для міжщелепних еластичних тяг.

156. Встановлення апарату для збільшення верхньої щелепи.

157. Встановлення закріпленого на місці пасивного ортодонтичного апарату.

158. Забезпечення фіксації напрямку прорізування зубів.

159. Повна коронка, акрилова смола, непряма реставрація.

160. Повна коронка, немателава, непряма реставрація.

161. Повна коронка, з фасеткою, непряма реставрація.

162. Повна коронка, металева, непряма реставрація.

163. Основна для коронки разом зі штифтом, непряма реставрація.

164. Попередня реставрація при встановленні зубної коронки, пряма.

165. Фіксація штифта і кореневого ковпачка, непряма.

166. Тимчасова коронка.

167. Пряме захисне покриття пульпи.

168. Пульпотомія.

169. Повна підготовча хемо-механічна обробка кореневого каналу.

170. Повна підготовча хемо-механічна обробка кореневого каналу, кожний додатковий канал.

171. Обтурація кореневих каналів.

172. Обтурація кореневих каналів, кожен додатковий канал.

173. Екстірпація пульпи або хірургічнаобробка кореневого каналу, проведення екстреної або паліативної процедури.

174. Пломбування кореневого каналу із застосуванням матеріалів, що резорбуються.

175. Видалення зуба або його частин(-и).

176. Секційне видалення зуба або його частин(-и).

177. Рефіксація існуючого кламера до зубного протезу.

178. Заміна або додавання нового кламера до зубного протезу .

179. Відновлення зламаного базису повного знімного протеза.

180. Відновлення зламаного базису часткового знімного протеза .

181. Заміна або додавання нового зуба на протезі.

182. Повторне встановлення існуючого зуба на зубному протезі.

183. Додавання зуба до частового знімного протеза для заміни видалення зуба або зуба з видаленою коронкою.

184. Відновлення або внесення додаткових змін до конструкції литого металевого каркаса.

185. Хірургічне оголення ретинованого зуба, з забезпеченням стимуляції і тампонування.

186. Хірургічне оголення ретинованого зуба з приєднанням пристрою ортодонтичного витягування.

187. Корекція положення дистопованих зубів.

188. Хірургічна корекція положення ретинованого зуба .

189. Шинування дистопованого зуба .

190. Реплантація і шинування зуба.

191. Трансплантація зуба або зубного зачатка.

192. Хірургічна операція для ізоляції та збереження нервових і судинних тканин.

193. Накладення швів на нервовий стовбур у поєднанні з проведенням стоматологічної процедури.

194. Контроль післяопераційної кровотечі після проведення стоматологічної процедури.

195. Повне обстеження ротової порожнини .

196. Періодичне обстеження ротової порожнини.

197. Обмежене обстеження ротової порожнини.

198. Консультація стоматолога, розширена.

199. Надання письмового висновку від стоматолога.

200. Інтраоральна періапікальна або прикусна рентгенографія.

201. Інтраоральна оклюзійна рентгенографія .

202. Томографія черепа, або частини черепа.

203. Тимчасовий міст, тіло состоподібного протеза.

204. Тимчасовий імплант-абатмент .

205. Відновлення тимчасового імпланта-абатмента.

206. Тіло мостоподібного протеза, пряма процедура.

207. Тіло мостоподібного протеза, непряма процедура .

208. Замкове кріплення.

209. Презиційне або магнітне (замкове кріплення), мостовподібний зубний протез.

210. Фіксація ретейнера для незнімної ортодонтичної конструкції, непряма.

211. Прилаштування незмінного (фіксованого) або знімного ортодонтичного апарата.

212. Відновлення змінного апарата, на основі композиту.

213. Відновлення кламера, пружини або зуба на знімному апараті.

214. Додавання кламера, пружини або зуба до знімного апарату .

215. Перебазуваня знімного апарату, обробленого.

216. инування і стабілізація зуба, пряма процедура.

217. Зішліфовування зубної емалі.

218. Встановлення внутрішньоротового пристрою у діагностованих випадках, пов'язаних з хропінням і синдромом обструктивного апное під час сну.

219. Післяопераційна стоматологічна допомога, не класифікована в інших рубриках.

220. Встановлення пасивного знімного ортодонтичного апарату, верхньощелепна дуга.

221. Встановлення пасивного знімного ортодонтичного апарату, нижньощелепна дуга.

222. Встановлення активного знімного ортодонтичного апарату, верхньощелепна дуга.

223. Встановлення активного знімного ортодонтичного апарату, нижньощелепна дуга.

224. Встановлення функціонального ортопедичного апарату .

225. Встановлення пластикових елайнерів для послідовного преміщення зубів.

226. Незначна оклюзійна корекція.

227. Клінічний аналіз оклюзії включно з пальпацією м'язів і суглобів.

228. Реєстрація і установка моделі для проведення аналізу оклюзії.

229. Установка оклюзійної шини.

230. Корекція положення існуючої оклюзійної шини.

231. Реєстрація рухів нижньої щелепи за допомогою пантографа.

232. Оклюзійна корекція після проведення аналізу оклюзії.

233. Допоміжна фізіотерапія стоматологічної спрямованості для скронево-нижньощелепного суглоба і пов'язаних структур.

234. Відновлення/додавання оклюзійної шини .

235. Фіксація замку (берега) до зуба для створення ортодонтичного зусилля.

236. Хірургічне видалення зуба, яке не потребує видалення кістки або поділу зуба на частини.

237. Видалення всіх зубів.

238. Хірургічне видалення зуба, яке потребує видалення кістки .

239. Хірургічне видалення зуба, яке потребує як видалення кістки, так і поділу зуба на частини.

240. Встановлення позаротового пристрою .

241. Встановлення повного знімного протезу верхньої щелепи.

242. Встановлення повного знімного протезу нижньої щелепи .

243. Встановлення армуючої металевої сітки або пластини.

244. Встановлення повних знімних протезів нижньої і верхної щелеп .

245. Встановлення часткового знімного протезу верхньої щелепи, на основі композита.

246. Встановлення часткового знімного протезу нижньої щелепи, на основі композита.

247. Встановлення часткового знімного протезу верхньої щелепи, литий металевий каркас.

248. Встановлення часткового знімного протезу нижньої щелепи, литий металевий каркас.

249. Встановлення ретейнера до часткового знімного протезу.

250. Встановлення кламера з оклюзійною накладкою до часткового знімного протезу.

251. Встановлення зуба до часткового знімного протезу .

252. Вкладка оверлей.

253. Презиційне або магнітне замкове кріплення, внутрішньопротезна частина.

254. Негайне протезування (заміна зуба).

255. Встановлення пружньої прокладки.

257. Встановлення металевого каркаса фасетки для протезного зуба.

Перелік частин ротової порожнини:

1. Різець 11.
2. Різець 12.
3. Клик 13.
4. Премоляр 14.
5. Премоляр 15.
6. Моляр 16.
7. Моляр 17.
8. Моляр 18.

9. Різець 21.
10. Різець 22.
11. Клик 23.
12. Премоляр 24.
13. Премоляр 25.
14. Моляр 26.
15. Моляр 27.
16. Моляр 28.
17. Різець 31.
18. Різець 32.
19. Клик 33.
20. Премоляр 34.
21. Премоляр 35.
22. Моляр 36.
23. Моляр 37.
24. Моляр 38.
25. Різець 41.
26. Різець 42.
27. Клик 43.
28. Премоляр 44.
29. Премоляр 45.
30. Моляр 46.
31. Моляр 47.
32. Моляр 48.
33. Губи.
34. Ясна.
35. Піднебіння.
36. Зубний ряд верхньої щелепи.
37. Зубний ряд нижньої щелепи.

ДОДАТОК Г

ПРОЕКТ ТАБЛИЦЬ ФІЗИЧНОЇ МОДЕЛІ БАЗИ ДАНИХ

Таблиця. Г.1

Таблиця сутності «Доступ»

access (Доступ)				1
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	id_access	Унікальний ключ доступу	int	
2	user_id	Унікальний ключ користувача	int	
3	diary_id	Унікальний ключ щоденника	int	

Таблиця. Г.2

Таблиця сутності «Щоденник»

diary (Щоденник)				2
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	iddiary	Унікальний ключ щоденника	int	
2	user_id	Унікальний ключ користувача	int	
3	diary_type	Тип щоденника	varchar	45

Таблиця. Г.3

Таблиця сутності «Лікар»

doctor_info (Лікар)				3
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	id_doctor	Унікальний ключ лікаря	int	
2	doctor_phone	Номер телефону	varchar	45
3	doctor_adress	Робоча адреса	varchar	100
4	doctor_name	Ім’я лікаря	varchar	45
5	doctor_speciality	Спеціалізація	varchar	45
6	user_id	Унікальний ключ користувача	int	

Таблиця. Г.4

Таблиця сутності «Файл»

file_ph (Файл)				4
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idfile	Унікальний ключ файлу	int	
2	file_name	Ім’я файлу	varchar	100
3	file_date	Дата	date	
4	file_adress	Адреса	varchar	100
5	procedure_id	Унікальний ключ процедури	int	

Таблиця. Г.5

Таблиця сутності «Частина ротової порожнини»

oral_cavity_parts (Частина ротової порожнини)				5
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	id_or_cav_part	Унікальний ключ частини ротової порожнини	int	
2	partname	Назва частини ротової порожнини	int	

Таблиця. Г.6

Таблиця сутності «Процедура»

procedure (Процедура)				6
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idprocedure	Унікальний ключ процедури	int	
2	procedure_id	Унікальний ключ системної процедури	int	
3	reord_id	Унікальний ключ запису	int	
4	comment	Коментар	varchar	450

Таблиця. Г.7

Таблиця сутності «Область застосування»

procedure_area (Область застосування)				7
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	id_area	Унікальний ключ області застосування	int	
2	procedure_id	Унікальний ключ процедури	int	
3	or_cav_part_id	Унікальний ключ частини ротової порожнини	int	

Таблиця. Г.8

Таблиця сутності «Системна процедура»

procedures (Системна процедура)				8
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idprocedure	Унікальний ключ процедури	int	
2	proc_name	Назва процедури	varchar	200
3	proc_type	Тип процедури	varchar	100

Таблиця. Г.9

Таблиця сутності «Запис чистки»

record_brushing (Запис чистки)				9
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	id_brushing	Унікальний ключ запису чистки	int	
2	brushing_date	Дата чистки	date	
3	number_of_times	Кількість разів	int	
4	morning_check	Чистка зранку	int	
5	after_meals	Чистка після прийому їжі	int	
6	before_sleep	Чистка перед сном	int	
7	diary_id	Унікальний ключ щоденника	int	

Таблиця. Г.10

Таблиця сутності «Симптом»

symptom (Симптом)				10
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idsymptom	Унікальний ключ симптому	int	
2	symptom_name	Назва симптому	varchar	65
3	level_pain	Рівень болю	float	
4	symptom_date	Дата створення запису про симптом	date	
5	symptom_datestart	Дата початку симптому	datetime	
6	diary_id	Унікальний ключ щоденника	int	

Таблиця. Г.11

Таблиця сутності «Запис»

teeth_record (Запис)				11
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idrecord	Унікальний ключ запису	int	
2	doctor_id	Унікальний ключ лікаря	int	
3	record_date	Дата запису	date	
4	treatment_id	Унікальний ключ лікування	int	
5	rating	Оцінка	varchar	100
6	record_adress	Адреса	varchar	100
7	prescription	Рецепт від лікаря	varchar	450

Таблиця. Г.12

Таблиця сутності «Зубна щітка»

toothbrush (Зубна щітка)				12
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idtoothbrush	Унікальний ключ зубної щітки	int	
2	diary_id	Унікальний ключ щоденника	int	

Продовж. табл. Г.12

№ п/п	Найменування стовбців	Примітка	Тип	Розмір
3	hardness	Жорсткість	varchar	100
4	color	Колір	varchar	45
5	start_date	Дата початку користування	date	
6	end_date	Дата припинення користування	date	
7	status	Статус	int	

Таблиця. Г.13

Таблиця сутності «Лікування»

treatment (Лікування)				13
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	idtreatment	Унікальний ключ лікування	int	
2	treatment_name	Назва	varchar	100
3	diary_id	Унікальний ключ щоденника	int	
4	start_date_trtment	Дата початку	date	
5	end_date_trtment	Дата кінця	date	

Таблиця. Г.14

Таблиця сутності «Користувач»

user (Користувач)				14
№ п/п	Найменування стовбців	Примітка	Тип	Розмір
1	iduser	Унікальний ключ користувача	int	
2	username	Ім’я користувача	varchar	45
3	email	Адреса електронної пошти	varchar	45
4	password	Пароль	varchar	100
5	usertype	Тип користувача	varchar	45
6	birthday	Дата народження	date	
7	sex	Стать	varchar	10

Продовж. табл. Г.14

№ п/п	Найменування стовбців	Примітка	Тип	Розмір
8	workadress	Робоча адреса	varchar	100
9	specialization	Спеціалізація	varchar	65
10	phone	Телефон	varchar	45

ДОДАТОК Ґ

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	**Опис**
Пояснювальні документи	
КваліфікаційнаРобота_Михайленко.doxc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
КваліфікаційнаРобота _Михайленко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
diplom.zip	Архів. Містить коди програми.
Презентація	
Презентація_Михайленко.ppt	Презентація кваліфікаційної роботи