МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут автоматики, кібернетики та обчислювальної техніки

"До захисту допущений"
Зав. кафедри комп'ютерних наук та прикладної математики

_____

«__» _____20___р.

КВАЛІФІКАЦІЙНА РОБОТА

Розробка інтелектуального користувацького інтерфейсу з використанням фреймворку AngularJS

Виконав: **Дворак Максим Петрович** _____
(прізвище, ім'я, по батькові) (підпис)

група ІПЗ-41

Керівник: **доцент, к.т.н. Жуковський В.В.** _____
(науковий ступінь, вчене звання, посада, прізвище, ініціали) (підпис)

Рівне – 2024

Зміст

РЕФЕРАТ

Кваліфікаційна робота: 60 сторінок, 51 рисунків, 5 таблиць, 10 джерел.

Метою кваліфікаційної роботи є розробка інтелектуального інтерфейсу користувача з використанням фреймворку AngularJS.

Об'єкт дослідження - процес створення ефективного та зручного інтелектуального інтерфейсу, який безперешкодно взаємодіє з користувачами.

Предмет дослідження - реалізація інтелектуального інтерфейсу користувача з використанням AngularJS, що включає в себе модулі, які покращують взаємодію з користувачем та покращують загальний користувацький досвід.

Методи дослідження - використання фреймворку AngularJS, сучасних методів веб-розробки, технології розпізнавання голосу та інтеграцію інтерактивних компонентів для створення динамічного користувацького інтерфейсу.

Новизна та практична цінність отриманих результатів. Визначення потреб користувачів та забезпечення чуйного та інтуїтивно зрозумілого інтерфейсу завжди користувалося високим попитом. Ця вимога є ще більш вираженою в епоху стрімкого технологічного прогресу та зростаючих очікувань користувачів. Основною перевагою цього проекту є створення голосового асистента, який значно покращує взаємодію з користувачем, дозволяючи керувати без допомоги рук і забезпечуючи більш доступний користувацький досвід.

Ключові слова: інтелектуальний інтерфейс користувача, AngularJS, голосовий асистент, веб-розробка, користувацький досвід, інтерактивні компоненти.

ВСТУП

Сучасний стан розвитку інформаційних технологій дозволяє значно підвищити ефективність та зручність роботи у різних галузях, зокрема і в медицині. Застосування інтелектуальних користувацьких інтерфейсів сприяє підвищенню продуктивності та точності виконання професійних завдань, зменшуючи навантаження на фахівців і забезпечуючи швидкий доступ до необхідної інформації. В умовах швидкого розвитку цифрових технологій і зростаючих потреб в інтерактивних системах, особливо актуальним стає створення спеціалізованих веб-додатків для медичних працівників.

Однією з актуальних проблем в стоматології є ведення медичної документації та інтерактивне взаємодія з пацієнтами. Зокрема, існує потреба в ефективних засобах для створення, зберігання та управління персональними картками пацієнтів, а також їх зубними картами. Це сприяє не тільки зручності роботи стоматолога, але й підвищує точність діагностики та лікування.

Дана кваліфікаційна робота присвячена розробці інтелектуального користувацького інтерфейсу з використанням фреймворку AngularJS, який буде застосований у веб-додатку для стоматологів. Цей додаток дозволяє реєструвати пацієнтів, створювати їх особисті картки та інтерактивні зубні карти, де лікар може взаємодіяти з зубами, видаляти їх, виділяти частини зубів із карієсом тощо. Важливою інноваційною складовою цього інтерфейсу є вбудований голосовий асистент, що забезпечує більш зручну та швидку взаємодію лікаря з системою. Наприклад, за допомогою голосових команд лікар може швидко виділити частини зуба з карієсом, не користуючись мишею або клавіатурою.

Наукова новизна роботи полягає у застосуванні сучасних веб-технологій для вирішення конкретних практичних завдань у стоматології, а саме – інтерактивного управління зубними картами пацієнтів з використанням голосового асистента. Практична значущість роботи визначається можливістю підвищення ефективності роботи стоматологів,

покращення якості наданих послуг і зменшення часу, необхідного для ведення документації.

Метою даної кваліфікаційної роботи є розробка інтелектуального користувацького інтерфейсу для стоматологів, що забезпечить зручне та ефективне управління медичною документацією та зубними картами пацієнтів. Для досягнення цієї мети були поставлені наступні завдання:

- Дослідити сучасні тенденції та вимоги до користувацьких інтерфейсів у медичних веб-додатках.
- Розробити архітектуру веб-додатку для стоматологів.
- Реалізувати функціональність для реєстрації пацієнтів і створення їх особистих карток.
- Створити інтерактивну зубну карту з можливістю взаємодії з окремими зубами та їх частинами.
- Інтегрувати голосовий асистент для взаємодії з зубною картою.
- Провести тестування та оцінку ефективності розробленого інтерфейсу.

Основні методи та підходи до вирішення поставленої проблеми включають аналіз сучасних веб-технологій, проектування архітектури додатку, програмування, впровадження алгоритмів голосового розпізнавання та інтерактивного управління інтерфейсом.

Розробка такого веб-додатку забезпечить новий рівень зручності та ефективності в роботі стоматологів, сприяючи підвищенню якості наданих медичних послуг та оптимізації робочого процесу.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ДЕТАЛЬНИЙ ОПИС СКЛАДОВИХ КОМПОНЕНТІВ СИСТЕМИ

1.1 Генезис розвитку наукових поглядів на проблему

Розробка інтелектуальних користувальницьких інтерфейсів для медичних систем, особливо для стоматологів, є сучасним напрямком, що поєднує досягнення в галузі інформаційних технологій і медицини.

Метою сучасних наукових досліджень є підвищення зручності та ефективності роботи медичного персоналу за допомогою інтерактивних систем, які зменшують ручне введення даних та автоматизують щоденні завдання. Однією з головних проблем у стоматології є ведення медичних записів та створення інтерактивних записів пацієнтів. Традиційні методи зберігання та обробки даних пацієнтів часто непрактичні та неефективні.

Сучасні інформаційні системи на основі мережевих технологій значно підвищують ефективність роботи та забезпечують швидкий доступ до даних і можливість інтерактивної обробки.

«Жодне медичне рішення не обходиться без електронної медичної картки. Це наріжний камінь медичного веб-дизайну, що надає лікарям доступ до медичних записів, лікування, діагнозів та інших аспектів історії хвороби в режимі реального часу. Маючи цю інформацію, лікарі можуть приймати кращі рішення і заощаджувати пацієнтам багато часу на повторні перевірки [1].» Це уривок з американської статті який демонструє нам актуальність нашого проекту. Звичайно в тексті йдеться не конкретно про стоматологічний напрямок, але концепція залишається такою ж важливою.

Відмова від паперових медичних записів і перехід на електронні системи охорони здоров’я дозволяє стоматологам швидко й точно отримувати доступ до важливої інформації про пацієнтів, допомагаючи покращити якість медичних послуг. Інтерактивні стоматологічні карти дозволяють лікарям ефективніше взаємодіяти з пацієнтами та проводити точну діагностику та плани лікування.

Крім того, запровадження голосових помічників додатково оптимізує процес лікування, зменшує концентрацію на ручних завданнях і покращує загальну ефективність роботи.

Цей підхід має велике значення в сучасній медичній практиці, оскільки він не лише забезпечує зручність для медичних працівників, але й покращує задоволеність пацієнтів, оскільки вони можуть довіряти точності та прозорості своїх медичних записів. Використання веб-технологій та інтерактивних інтерфейсів відкриває нові можливості для вдосконалення стоматологічної практики та підвищення ефективності.

1.2 Актуальність роботи

Веб-сайти стали невід'ємною частиною сучасної стоматологічної практики, відіграючи вирішальну роль у підвищенні ефективності та результативності стоматологічної допомоги. Вони є основним засобом, за допомогою якого стоматологи ведуть записи пацієнтів, повідомляють про плани лікування, призначають зустрічі та багато іншого. Завдяки зручному доступу до інформації через Інтернет, наш сайт дозволяє стоматологам знаходити та оновлювати дані пацієнтів за лічені секунди, значно спрощуючи робочі процеси та покращуючи загальну якість обслуговування.

Рис. 1.1 Посилання на веб-сайт

Однією з ключових інновацій мого проекту повинна була бути розробка інтерактивної зубної карти, яка має важливе значення для сучасної стоматологічної практики. Ця карта, повинна була бути десь розроблена, а потім реалізована за допомогою якоїсь мови програмування. Фінальний продукт повинен допомогти стоматологам візуально взаємодіяти зі стоматологічною картою кожного пацієнта. Ось ключові причини, чому ця можливість має вирішальне значення:

Традиційні паперові стоматологічні карти забирають багато часу на оновлення та схильні до помилок. Інтерактивна цифрова карта спрощує цей процес, забезпечуючи швидке оновлення та доступ до даних пацієнта в режимі реального часу. Така ефективність робочого процесу означає, що стоматологи можуть витрачати більше часу на лікування пацієнтів, а не на адміністративні завдання, тим самим підвищуючи загальну продуктивність стоматологічної практики.

Інтеграція голосового асистента в систему додає ще один рівень зручності та ефективності. Голосові команди дозволяють стоматологам взаємодіяти з картою зуба без необхідності вручну обирати ту чи іншу функцію на зубній мапі, що особливо корисно під час процедур. Наприклад, стоматолог може просто сказати: "Позначте карієс на третьому зубі знизу", і система повинна автоматично оновити карту. Така система без необхідності вручну виділяти всілякі хвороби пацієнта на екрані підвищує стерильність і прискорює процес, оскільки стоматологам не потрібно знімати рукавички або відриватися від роботи, щоб взаємодіяти з комп'ютером.

Голосові асистенти та інтерактивні цифрові карти вже доводять свою ефективність у різних сферах медицини. Наприклад, системи електронних медичних карток (ЕМК) з голосовим управлінням стають все більш популярними в лікарнях і клініках. Ці системи дозволяють лікарям оновлювати записи пацієнтів, призначати зустрічі та отримувати інформацію за допомогою голосових команд, заощаджуючи час і зменшуючи

адміністративний тягар. Застосування подібних технологій у стоматологічній практиці гарантує, що стоматологи зможуть отримати такі ж переваги.

У 2024 році web-розробка залишається стратегічним напрямком у розвитку цифрового світу. З її допомогою створюються інноваційні веб-проекти, які визначають майбутнє Інтернету. Зростаючі вимоги до мобільності, безпеки та новітніх технологій роблять web-розробку ключовим компонентом глобальної технологічної еволюції [2].

1.3 Основні складові компоненти системи

Розроблена система складається з наступних основних компонентів:

- Інтерфейс реєстрації пацієнтів – забезпечує можливість створення та зберігання особистих даних пацієнтів.
- Персональна карта пацієнта – включає в себе деяку інформацію про пацієнта та його дані.
- Зубна карта пацієнта – інтерактивний інструмент, що дозволяє стоматологу взаємодіяти з окремими зубами та їх частинами. Зубна карта надає можливість відмічати проблемні зони, такі як карієс, видаляти зуби, виділяти пломбу та додавати коментарі. Інтерфейс також підтримує кольорове кодування для позначення різних станів зубів та необхідних процедур, що допомагає стоматологу швидко орієнтуватися в інформації.
- Голосовий асистент – інноваційна складова системи, що дозволяє стоматологу керувати інтерфейсом за допомогою голосових команд. Це значно спрощує роботу, дозволяючи лікарю зосередитися на пацієнті без необхідності відволікатися на комп'ютер. Голосовий асистент підтримує різні команди, такі як "карієс третій зуб знизу", що автоматично виділяє відповідну ділянку на зубній карті.
- Форма реєстрації лікаря – дозволяє лікарям зареєструватися в системі, проте на даному етапі реалізована лише фронтенд-частина. Форма включає поля для введення логіна та пароля з різними валідаціями, які

забезпечують мінімальні вимоги до складності пароля та правильності введених даних. Це допомагає захистити облікові записи лікарів від несанкціонованого доступу.

Використання AngularJS, який я обрав як основу для свого проекту, забезпечує гнучкість та модульність системи, що дозволяє легко додавати нові функції та змінювати існуючі. AngularJS підтримує двостороннє зв'язування даних, що забезпечує автоматичне оновлення інтерфейсу при зміні даних, що покращує користувацький досвід.

Angular Material та FxLayout забезпечують адаптивний дизайн, що дозволяє використовувати систему на різних пристроях, включаючи планшети та смартфони [3] [4]. Angular Material надає набір готових UI-компонентів, таких як кнопки, форми, модальні вікна та інші, що забезпечують єдиний стиль та високу функціональність інтерфейсу. FxLayout спрощує створення адаптивних макетів, дозволяючи легко налаштовувати вигляд інтерфейсу на різних екранах.

1.4 Аналіз новітніх наукових матеріалів і порівняння з конкурентами

Аналіз сучасної наукової літератури показує, що використання веб-технологій у медичних системах стає все більш поширеним. Відкриті стандарти, такі як HTML5, CSS3, та JavaScript, дозволяють створювати складні інтерфейси, що можуть адаптуватися до потреб користувачів. Зокрема, AngularJS є одним з найбільш популярних фреймворків для розробки односторінкових додатків, завдяки своїй модульності та можливості швидкого створення складних інтерактивних інтерфейсів.

Проаналізувавши ринок, я знайшов 4 сайти, які використовуються стоматологами, для взаємодії зі своїми пацієнтами та їх зубними хворобами.

Denta Pro та його переваги та недоліки:

Таблиця 1.1

Переваги	Недоліки
Комплексна CRM, спеціально розроблена для стоматологічних практик, що оптимізує управління пацієнтами та комунікацію.	Складність: Через широкі можливості системи новим користувачам може здатися, що вона складна і в ній важко орієнтуватися на початковому етапі.
Автоматизована комунікація з пацієнтами та планування зустрічей зменшує адміністративне навантаження.	Вартість: Широкі функціональні можливості мають вищу ціну, що може бути бар'єром для невеликих стоматологічних практик.
Інструменти для детального планування лікування та відстеження історії пацієнта.	Кастомізація: Хоча Denta Pro пропонує багато функцій, широкі можливості кастомізації можуть бути обмежені порівняно з більш гнучкими платформами.
Бізнес-аналітика для визначення зон зростання та покращення якості послуг.	
Онлайн-бронювання та автоматичні сповіщення підвищують зручність для пацієнтів.	

Baza Znanii та його переваги та недоліки:

Таблиця 1.2

Переваги	Недоліки
Зосереджується на веденні детальних стоматологічних карт і записів клієнтів.	Обмежена інтеграція: Може не так легко інтегруватися з іншим програмним забезпеченням або

	системами, що використовуються в клініці, що може спричинити потенційні перерви в робочому процесі.
Сприяє комплексному документуванню та відстеженню лікування пацієнтів.	Інтерфейс користувача: Інтерфейс може бути менш інтуїтивно зрозумілим у порівнянні з більш сучасними, зручними дизайнами.
Підтримує співпрацю та обмін даними між стоматологами.	Масштабованість: Малі масштаби та менше розширених функцій можуть не відповідати потребам великих або більш спеціалізованих стоматологічних практик.

Clinic Cards та його переваги та недоліки:

Таблиця 1.3

Переваги	Недоліки
Підкреслює простоту використання і доступність в управлінні записами пацієнтів.	Набір функцій: Може не вистачати деяких розширених функцій, які пропонують більш комплексні CRM-системи.
Функції включають планування зустрічей, управління історією хвороби та планами лікування.	Підтримка та оновлення: Потенційно обмежена підтримка клієнтів і менша кількість регулярних оновлень порівняно з великими конкурентами.
Зручний для мобільних пристроїв інтерфейс для доступу на ходу.	Кастомізація: Менша гнучкість у налаштуванні під унікальні потреби практики.

Zubok CRM та його переваги та недоліки:

Таблиця 1.4

Переваги	Недоліки
Універсальне рішення для управління стоматологічною клінікою, що інтегрує управління пацієнтами, планування та фінансовий облік.	Крива навчання: Новим користувачам може знадобитися значне навчання для ефективного використання всіх функцій.
Включає маркетингові інструменти, такі як SMS-повідомлення та повідомлення електронною поштою, для покращення комунікації з пацієнтами та їх утримання.	Користувацький досвід: Користувацький досвід може бути не таким спрощеним, як на нових, більш сучасних платформах.
	Інтеграція: Потенційні проблеми з інтеграцією з іншими сторонніми інструментами та програмним забезпеченням.

Унікальність саме мого проекту повинна полягати в інтеграції голосового асистента. Хоча вищезгадані платформи пропонують надійні рішення для управління стоматологічною практикою, інтеграція голосового асистента дає кілька унікальних переваг:

Підвищена ефективність: Стоматологи можуть використовувати голосові команди для взаємодії з картою зубів, що дозволяє їм оновлювати записи пацієнтів і плани лікування без допомоги рук. Це скорочує час, витрачений на адміністративні завдання, і дозволяє більше зосередитися на лікуванні пацієнтів.

Покращена доступність: Голосове управління може бути особливо корисним під час стоматологічних процедур, дозволяючи стоматологам

отримувати доступ до інформації про пацієнта і оновлювати записи без необхідності знімати рукавички або переривати свою роботу.

Інновації та модернізація: Включення голосового асистента демонструє сучасний, технологічний підхід, який може привабити технічно підкованих пацієнтів і професіоналів, які шукають найсучасніші інструменти.

Індивідуальна карта зубів: Створюючи індивідуальну карту зуба у Figma та переводячи її в код, інструмент пристосовується до конкретних потреб практики, пропонуючи рівень кастомізації, який не можуть забезпечити типові рішення.

1.5 Голосові асистенти

Голосові асистенти є потужними інструментами, що значно полегшують взаємодію користувача з комп'ютерними системами та мобільними додатками. Вони використовують технології розпізнавання мови та штучного інтелекту, щоб розуміти та виконувати команди користувачів.

Ось основні переваги та зручності використання голосових асистентів:

1. Зручність використання: Голосові асистенти дозволяють виконувати завдання, не використовуючи клавіатуру або мишу. Це особливо корисно в ситуаціях, коли руки зайняті або коли користувачеві зручніше використати голосові команди. Наприклад, лікар може швидко внести інформацію про пацієнта, не відволікаючись від медичних процедур.

2. Швидкість доступу: Голосові команди можуть бути швидшим способом виконання завдань у порівнянні з традиційними методами. Наприклад, пошук інформації в інтернеті, встановлення будильника або управління музикою можна виконати за кілька секунд, просто вимовивши команду.

3. Інклюзивність: Голосові асистенти роблять технології доступнішими для людей з обмеженими можливостями. Люди з проблемами зору або моторики можуть використовувати голосові команди для взаємодії з комп'ютерами та смартфонами, що значно підвищує їхню незалежність та якість життя.

4. Мультитаскінг: Голосові асистенти дозволяють виконувати декілька завдань одночасно. Наприклад, під час приготування їжі можна попросити асистента прочитати рецепт або відправити повідомлення. Це значно підвищує ефективність і продуктивність.

5. Персоналізація: Голосові асистенти можуть запам'ятовувати переваги користувачів та адаптуватися до їхніх потреб. Вони можуть надавати персоналізовані рекомендації, запам'ятовувати розклад і нагадувати про важливі події.

Приклади існуючих голосових асистентів:

Siri (Apple): Siri є голосовим асистентом, вбудованим у пристрої Apple, такі як iPhone, iPad, Mac, та Apple Watch. Siri може відповідати на запитання, надсилати повідомлення, встановлювати нагадування, і навіть керувати смарт-пристроями вдома.

Рис. 1.2 Голосовий асистент Siri

Google Assistant (Google): Google Assistant доступний на пристроях Android і iOS, а також у розумних колонках, таких як Google Nest. Він може виконувати широкий спектр завдань, від пошуку в інтернеті до управління домашньою автоматизацією.

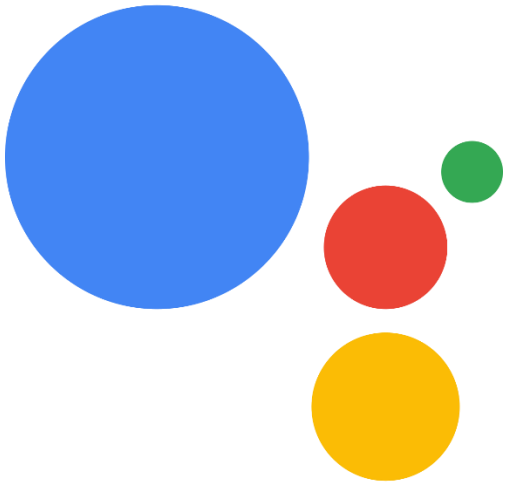

Рис. 1.3 Голосовий асистент Google Assistant

Alexa (Amazon): Alexa є голосовим асистентом, інтегрованим у пристрої Amazon Echo та інші розумні пристрої. Alexa може відтворювати музику, відповідати на запитання, керувати смарт-пристроями, робити покупки онлайн і багато іншого.

Рис 1.4 Amazon Echo з вбудованим голосовим асистентом Alexa

Cortana (Microsoft): Cortana інтегрована в операційну систему Windows і доступна на пристроях Windows, Xbox, і через додатки на iOS та Android. Вона може допомагати з управлінням календарем, надсилати електронні листи, знаходити файли і надавати відповіді на запитання.

Рис. 1.5 Голосовий асистент Cortana

Голосові асистенти продовжують еволюціонувати, стаючи все більш точними та універсальними. Вони не тільки полегшують повсякденні завдання, але й відкривають нові можливості для взаємодії з технологіями, роблячи їх більш інтуїтивно зрозумілими та доступними для широкого кола користувачів.

Саме тому мені прийшла ідея розробити в певній мірі прототип вище наведених голосових асистентів, проте асистента який буде допомагати лікарям краще та простіше позначати всілякі захворювання на зубах своїх пацієнтів.

РОЗДІЛ 2. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ТА ПЛАТФОРМИ РОЗРОБКИ ПРОЕКТУ

2.1 Вибір середовища розробки проекту

При розробці проекту я обрав Visual Studio Code (VS Code) як основне середовище розробки. Цей вибір був обумовлений кількома факторами, які роблять VS Code чудовим інструментом для сучасних веб-розробників.

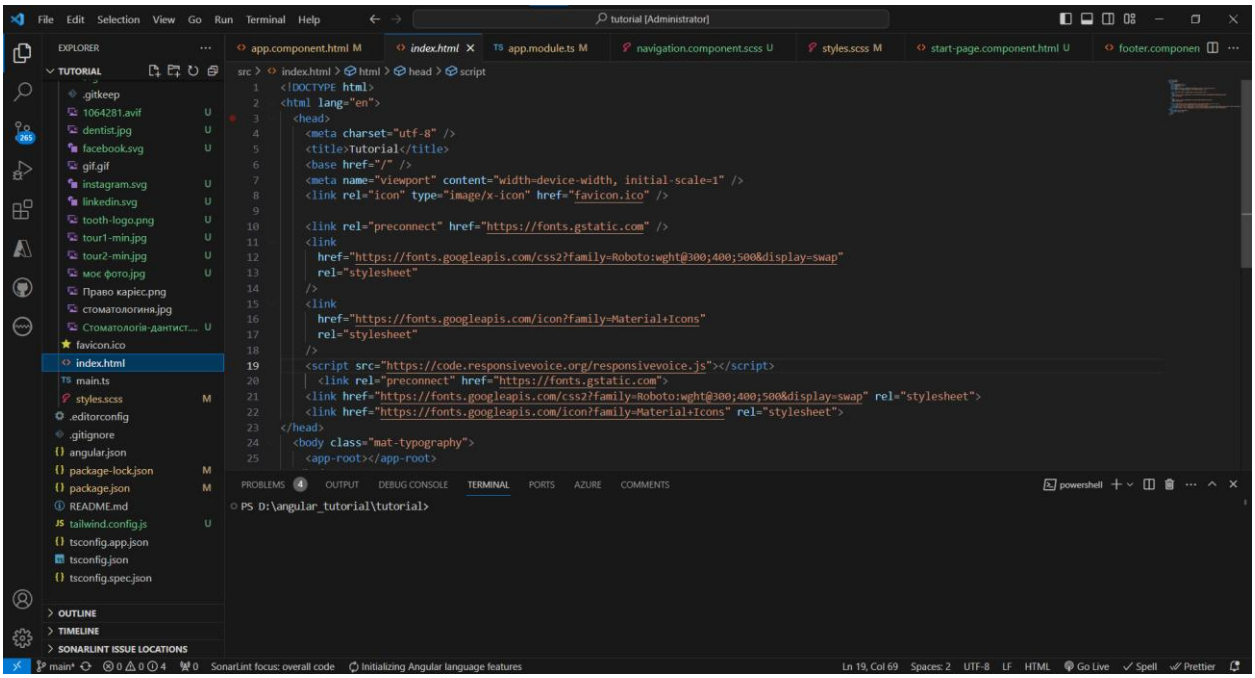

Рис. 2.1 Дизайн VS Code

Переваги VS Code:

- Легка вага та швидкість: VS Code є легким редактором коду, який запускається швидко і не навантажує системні ресурси. Це важливо для розробників, які працюють на різних типах апаратного забезпечення, включаючи менш потужні ноутбуки та комп’ютери.
- Інтеграція з Git: Вбудована підтримка Git дозволяє легко керувати версіями коду, виконувати коміти, стежити за змінами та працювати з гілками безпосередньо з редактора. Це значно полегшує роботу в команді та забезпечує ефективне управління проєктом.
- Розширюваність: VS Code має величезну кількість розширень, які можна завантажити з Visual Studio Code Marketplace. Ці розширення додають нові функції, такі як підтримка додаткових мов програмування,

лінтери, автозаповнення коду, інтеграція з різними сервісами та інструментами.

- Підтримка багатьох мов програмування: VS Code підтримує велику кількість мов програмування з коробки, а завдяки розширенням можна додати підтримку практично будь-якої мови. Це робить його універсальним інструментом для розробки проектів різної складності та спрямованості.
- Інтеграція з інструментами розробки: VS Code легко інтегрується з іншими інструментами розробки, такими як дебагери, тестові фреймворки, системи управління базами даних тощо. Це дозволяє створювати повний цикл розробки та тестування без необхідності перемикатися між різними додатками.
- Активна спільнота та підтримка: VS Code має активну спільноту розробників та підтримку від Microsoft. Регулярні оновлення, нові функції та виправлення помилок роблять цей редактор надійним та сучасним інструментом для розробки.

Недоліки VS Code:

- Висока залежність від розширень: Основна сила VS Code - його розширюваність, але це також може бути недоліком. Для отримання повного функціоналу часто доводиться встановлювати велику кількість розширень, що може призвести до конфліктів між ними та вплинути на продуктивність.
- Обмежені можливості з коробки: Без встановлення розширень VS Code є відносно базовим редактором коду. Деякі розробники можуть знайти цей підхід менш зручним, ніж інші редактори, які мають більше вбудованих функцій.
- Використання ресурсів при великій кількості розширень: Хоча VS Code сам по собі є легким редактором, велика кількість встановлених розширень може значно збільшити його використання ресурсів. Це

може призвести до зниження продуктивності, особливо на менш потужних машинах.

Порівняння з конкурентами:

- Sublime Text: Sublime Text є іншим популярним редактором коду, відомим своєю швидкістю та легкістю. Однак, він не має такої ж глибокої інтеграції з Git та меншої кількості розширень у порівнянні з VS Code. Крім того, Sublime Text є платним, тоді як VS Code є безкоштовним.
- Atom: Atom, розроблений GitHub, також є потужним редактором коду з великою кількістю розширень. Проте, Atom відомий своєю повільною роботою у порівнянні з VS Code, особливо при роботі з великими проектами.
- IntelliJ IDEA: IntelliJ IDEA є одним з найкращих IDE для розробки на Java та інших мовах. Хоча він пропонує багатий функціонал з коробки, його висока вартість та велике використання ресурсів роблять його менш привабливим для тих, хто шукає легке та безкоштовне рішення.

Таким чином, VS Code є чудовим вибором для розробки завдяки своїй легкості, гнучкості та широкій підтримці інструментів та мов програмування. Його безкоштовність та активна підтримка спільноти роблять його одним з найпопулярніших середовищ розробки в світі.

2.2 Angular

Angular – це потужний фреймворк для розробки веб-додатків, який розробляється та підтримується компанією Google.

Ось декілька основних переваг Angular про які я прочитав в книзі “AngularJS Up I Running” [5]:

- AngularJS є мета-фреймворком для створення односторінкових додатків (SPA). З клієнтським шаблонуванням та інтенсивним використанням JavaScript, створення та підтримка додатка можуть стати нудними та обтяжливими. AngularJS прибирає зайве і виконує важку роботу, щоб ми могли зосередитися виключно на найважливішому.

- Додаток на AngularJS вимагатиме менше рядків коду для виконання завдання, ніж чисте JavaScript рішення з використанням jQuery. У порівнянні з іншими фреймворками, ви все одно будете писати менше шаблонного коду і чистіший код, оскільки він переносить вашу логіку в багаторазові компоненти і знімає її з виду.
- Більшість коду, який ви пишете в додатку на AngularJS, буде зосереджена на бізнес-логіці або основній функціональності вашого додатка, а не на непотрібному рутинному коді. Це результат того, що AngularJS бере на себе весь шаблонний код, який ви б інакше зазвичай писали, а також архітектурної схеми MVC.
- Декларативна природа AngularJS робить легшим написання та розуміння додатків. Легко зрозуміти намір додатка, просто поглянувши на HTML та контролери. У певному сенсі, AngularJS дозволяє створювати HTMLX (замість того, щоб покладатися на HTML5 або чекати на HTML6 тощо), який є підмножиною HTML, що відповідає вашим потребам і вимогам.
- Додатки AngularJS можна стилізувати за допомогою CSS і HTML незалежно від їхньої бізнес-логіки та функціональності. Тобто, цілком можливо змінити весь макет і дизайн додатка, не торкаючись жодного рядка JavaScript.
- Шаблони додатків AngularJS написані на чистому HTML, тому дизайнерам буде легше працювати з ними та стилізувати їх.
- Надзвичайно просто здійснювати модульне тестування додатків AngularJS, що також робить додаток стабільнішим і легшим у підтримці протягом тривалого часу. Нові функції? Потрібно внести зміни до існуючої логіки? Все це легко з такою надійною базою тестів.
- Нам не потрібно відмовлятися від улюблених компонентів jQueryUI чи BootStrap. AngularJS добре працює з бібліотеками сторонніх компонентів і надає нам гачки для їх інтеграції на власний розсуд.

Проте, нажаль Angular має і деякі недоліки, ось деякі з них:

- Крута крива навчання: Angular має багатий набір функцій і складну архітектуру, що може бути важким для початківців. Вивчення всіх аспектів Angular вимагає значних зусиль та часу.
- Великий розмір додатків: Додатки, створені за допомогою Angular, можуть бути досить великими за розміром, що впливає на час завантаження, особливо на мобільних пристроях з повільним інтернетом. Проте, існують методи оптимізації, такі як lazy loading та мінімізація коду, які можуть зменшити цей недолік.
- Складність налагодження: Через використання механізму ін'єкції залежностей та інших абстракцій, налагодження Angular-додатків може бути складнішим порівняно з іншими фреймворками.
- Часті оновлення: Angular активно розвивається, що призводить до частих оновлень. Хоча це дозволяє використовувати нові функції та покращення, іноді це може спричинити проблеми з сумісністю та вимагати додаткових зусиль для оновлення існуючих проектів.
- Висока залежність від екосистеми: Angular сильно залежить від своєї екосистеми та специфічних інструментів, таких як Angular CLI. Це може обмежити гнучкість розробників та вимагати додаткових зусиль для інтеграції з іншими інструментами та технологіями.

Також для загального розвитку, перед початком роботи над цим проектом я переглянув курс по AngularJS на платформі YouTube [6].

Загалом, Angular є потужним інструментом для створення сучасних веб-додатків, проте його використання потребує значних знань та зусиль для ефективного освоєння та застосування.

2.3 Порівняння Angular з конкурентами

React і Angular є двома найпопулярнішими фреймворками для розробки сучасних веб-додатків. Кожен має свої особливості, переваги та недоліки та підходить для різних типів проектів.

React — це бібліотека інтерфейсу користувача, розроблена Facebook. Це основа для створення інтерактивних компонентів інтерфейсу користувача.

Саме через те, що React та Angular є найбільшими конкурентами на ринку фреймворків, я вирішив їх порівняти.

Таблиця 2.1

Параметр	React	Angular
Тип	Бібліотека для створення користувацьких інтерфейсів	Повноцінний фреймворк для розробки односторінкових додатків
Розробник	Facebook	Google
Основний підхід	Компонентний підхід	Компонентний підхід, модульність
Декларативні шаблони	Використовує JSX	Використовує HTML-шаблони
Двостороннє зв'язування даних	Ні, використовує одностороннє зв'язування даних	Так, підтримує двостороннє зв'язування даних
DOM	Віртуальний DOM для оптимізації рендерингу	Реальний DOM
Гнучкість	Висока, легко інтегрується з іншими бібліотеками	Менш гнучкий, забезпечує комплексне рішення
Крива навчання	Плоска, простіший для початківців	Крута, потребує більше зусиль для освоєння
Інструменти для розробників	Багато сторонніх інструментів	Angular CLI, вбудовані інструменти
Розмір додатків	Зазвичай менший	Зазвичай більший

Продуктивність	Висока завдяки віртуальному DOM	Може бути повільнішим через двостороннє зв'язування даних
Частота оновлень	Регулярні, але менш часті	Часті оновлення
Екосистема	Велика спільнота, багато сторонніх бібліотек	Велика спільнота, комплексне рішення "все-в-одному"
Можливість тестування	Підтримка unit-тестування	Потужні інструменти для тестування
Сумісність з іншими інструментами	Висока, добре інтегрується з jQueryUI та Bootstrap	Висока, але менша гнучкість у виборі сторонніх інструментів

React підходить для проектів, які потребують високої гнучкості та продуктивності. Підходить для розробників, які цінують легкість інтеграції з іншими бібліотеками та легкість навчання.

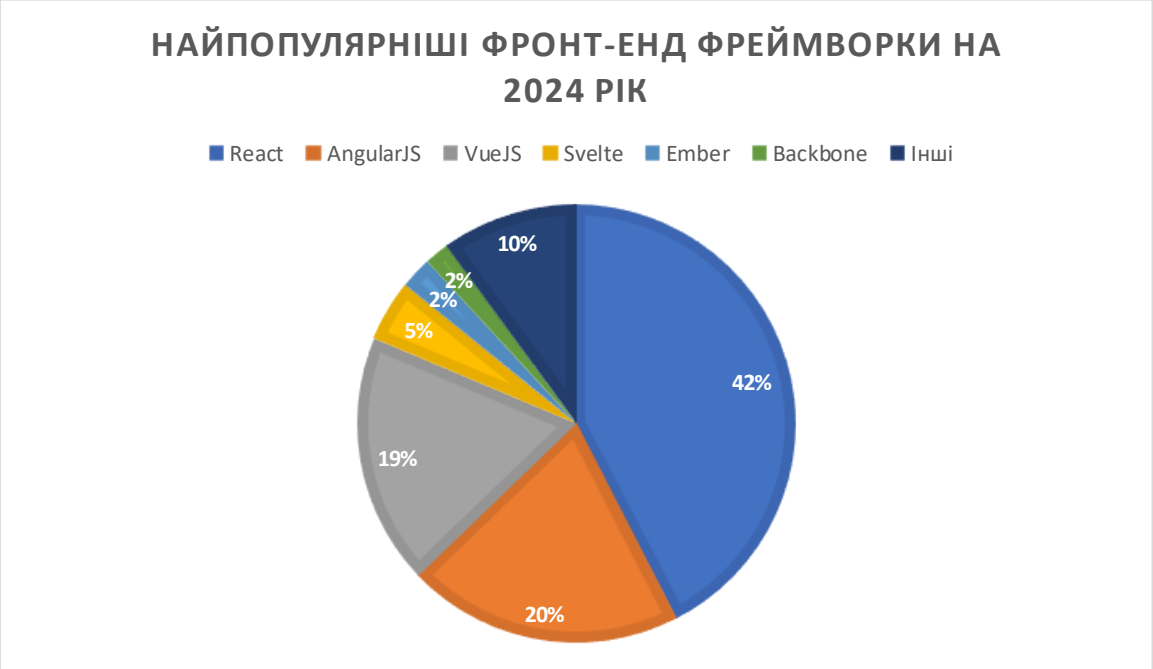

Рис. 2.3 Діаграма популярності фреймворків

Я ж обрав Angular тому, що раніше вже мав з ним досвід роботи і мені відповідно було набагато простіше писати проект на базі саме цього фреймворку, тому я його і обрав. Також, ось декілька однозначних переваг цього фреймворку над іншими.

Готове рішення: AngularJS - це повноцінний фреймворк, який надає все необхідне "з коробки", від маршрутизації та обробки форм до управління HTTP-клієнтами та станами. Це особливо корисно для розробки багатофункціональних додатків без значної залежності від сторонніх бібліотек.

Синхронізовані дані: Двостороння прив'язка даних в AngularJS гарантує, що будь-які зміни в моделі автоматично відображаються у поданні і навпаки. Це має вирішальне значення для інтерактивних стоматологічних схем, де потрібні оновлення в реальному часі та негайний зворотній зв'язок.

Архітектура на основі компонентів: AngularJS дозволяє розробникам створювати додатки за модульним принципом. Ця модульність полегшує повторне використання коду, легше обслуговування та паралельну розробку, що може пришвидшити процес розробки та покращити якість коду.

Безпека типів: AngularJS використовує TypeScript, статично типізовану підмножину JavaScript, яка допомагає виявляти помилки на ранніх стадіях розробки, покращує читабельність коду, а також надає потужні інструменти та функції автозавершення.

Ефективний робочий процес розробки: Angular CLI (інтерфейс командного рядка) спрощує процес розробки завдяки потужним інструментам для планування, створення, тестування та розгортання додатків. Це може значно підвищити продуктивність та забезпечити дотримання найкращих практик.

Керовані залежності: Вбудована в AngularJS система ін'єкції залежностей полегшує управління сервісами, компонентами та іншими залежностями, сприяючи кращій організації та тестуванню додатку.

Простота тестування: AngularJS розроблений з думкою про тестування. Фреймворк включає в себе інструменти та практики, які спрощують модульне та наскрізне тестування, гарантуючи, що додаток залишається стабільним та підтримуваним протягом тривалого часу.

Сильна підтримка: AngularJS підтримується Google і має велику спільноту розробників. Це означає велику кількість документації, безліч навчальних посібників, багату екосистему сторонніх бібліотек, а також постійні оновлення та вдосконалення.

2.4 Figma

Figma — це потужний інструмент для розробки інтерфейсів, який надає широкі можливості створення прототипів і співпраці між дизайнерами та розробниками. Його простота використання, функціональність і функції співпраці в режимі реального часу роблять його однією з найпопулярніших платформ у світі дизайну. Однією з головних переваг Figma є те, що вона доступна через браузер, що дозволяє користувачам працювати з будь-якого пристрою, підключеного до Інтернету, без встановлення додаткового програмного забезпечення.

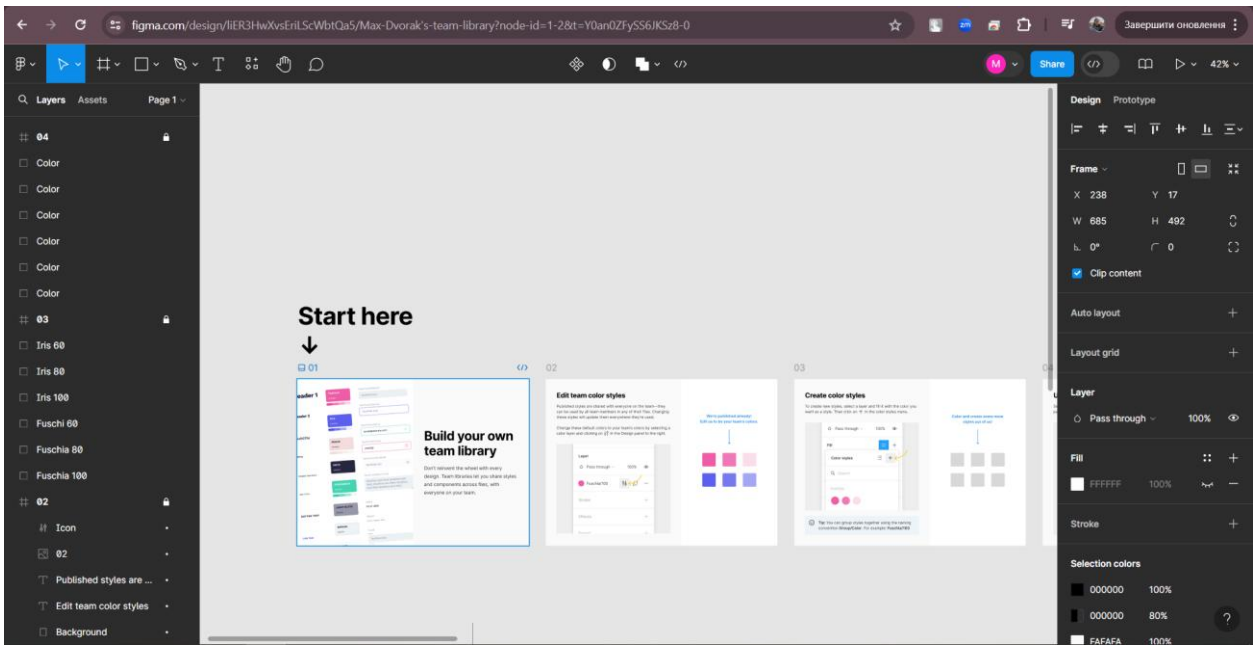

Рис. 2.4 Інтерфейс Figma

Підтримка Figma векторної графіки робить її ідеальною для створення складних дизайнів і високоточних моделей. Цей інструмент надає повну

функціональність для малювання, редагування та організації графічних елементів, включаючи можливість створювати багаторазові компоненти, що значно полегшує роботу з великими проектами.

Завдяки цьому дизайнерам легко створювати та підтримувати узгоджені стилі у своїх програмах. Однією з важливих особливостей Figma є функція співпраці. Дизайнери, розробники та інші зацікавлені сторони можуть одночасно працювати над одним проектом і залишати коментарі та пропозиції в режимі реального часу. Це значно прискорює процес розробки та затвердження дизайну, зменшує потребу в кількох ітераціях і покращує комунікацію команди.

Figma також пропонує інтеграцію з іншими інструментами та платформами, такими як Slack, Zeplin і Trello, що робить її ще більш гнучкою та зручною для користувача. Крім того, Figma має потужну систему плагінів, яка дозволяє вам розширити її функціональність відповідно до потреб вашого конкретного проекту чи команди.

Ще однією важливою перевагою Figma є можливість створювати інтерактивні прототипи. Це дозволяє розробникам швидко тестувати та демонструвати функціональність майбутніх програм без написання коду. Прототипи Figma можуть містити складні переходи та анімацію, що дає вам більш реалістичне уявлення про ваш кінцевий продукт.

Що стосується продуктивності, Figma має хорошу продуктивність завдяки своїй хмарній архітектурі, що дозволяє їй ефективно обробляти великі проекти, не навантажуючи локальні ресурси користувачів. Крім того, постійні оновлення та вдосконалення інструменту забезпечують придатність і відповідність останнім вимогам ринку дизайну.

Figma зуміла об'єднати весь набір інструментів для дизайну, щоб забезпечити комплексне рішення. Figma охоплює майже все, що вам потрібно для створення складного інтерфейсу, від мозкового штурму та створення каркасів до прототипування та спільного використання активів. Крім цього,

Figma виходить за межі дизайну продукту та генерує код CSS, iOS і Android для використання розробниками [7].

Ось як виглядає готовий варіант вималюваних тридцяти двох зубів в Figma[8].(рис. 2.4)

Рис. 2.5 Фінальний вигляд зубної карти в Figma

Незважаючи на всі переваги, у Figma є і недоліки. По-перше, робота вимагає постійного підключення до Інтернету, що може бути проблематичним, якщо доступ до мережі обмежений. Хоча Figma пропонує безкоштовну версію, деякі розширені функції доступні лише за платною підпискою, що може обмежувати роботу невеликих команд або окремих дизайнерів.

Загалом, Figma — це потужний і гнучкий інструмент розробки інтерфейсу, який пропонує високий рівень продуктивності та простоти використання. Можливості співпраці та інтеграції з іншими платформами роблять його ідеальним вибором для команд, які працюють над складними проектами, які потребують ефективного спілкування та швидкого впровадження змін.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Компоненти в AngularJS

Проект в AngularJS зазвичай складається з компонентів, які значно покращують розуміння структури проекту навіть для користувача який ніколи раніше не працював з цим проектом.

Компоненти є основними будівельними блоками в Angular, які дозволяють створювати та підтримувати складні інтерфейси користувача. Кожен компонент в Angular складається з трьох основних частин: шаблону, логіки та стилів.

Шаблон: Це HTML-код, який визначає візуальну частину компонента. Він містить елементи інтерфейсу користувача, такі як кнопки, форми, таблиці тощо. Шаблон може також містити прив'язки даних та Angular директиви, що дозволяє динамічно змінювати вміст на основі даних компонента.

Логіка: Це TypeScript-клас, який містить всю бізнес-логіку компонента. Клас визначає властивості та методи, які можуть використовуватися в шаблоні для відображення та обробки даних. Логіка компонента також обробляє події, які відбуваються в шаблоні, такі як кліки на кнопки або введення тексту в форми.

Стилі: Це CSS або SCSS-стилі, які визначають зовнішній вигляд компонента. Стилі можуть бути інкапсульовані в компонент, що дозволяє уникнути конфліктів з іншими стилями на сторінці.

Компоненти в Angular є модульними і незалежними, що дозволяє легко їх використовувати повторно в різних частинах додатка. Це зменшує кількість коду, який потрібно писати, та спрощує підтримку додатка.

Кожен компонент в Angular інкапсулює свою логіку, шаблон та стилі, що забезпечує їх незалежність від інших частин додатка. Це допомагає уникнути конфліктів між різними компонентами та сприяє більш організованій структурі коду.

Оскільки компоненти в Angular є незалежними та ізольованими, їх легше тестувати. Можна тестувати кожен компонент окремо, що покращує якість коду та зменшує кількість помилок у додатку.

Використання компонентів дозволяє розробникам використовувати декларативний підхід до створення інтерфейсів. Це означає, що замість написання великої кількості імперативного коду, розробники можуть просто визначити, як виглядає інтерфейс у HTML, а Angular автоматично оновлює його відповідно до змін у даних.

Використання компонентів дозволяє підтримувати консистентність у всьому додатку. Оскільки компоненти можуть бути використані повторно, їхній вигляд і поведінка залишаються однаковими незалежно від того, де вони використовуються.

Компоненти дозволяють легко керувати станом додатка. Оскільки кожен компонент відповідає за свій власний стан, можна уникнути складнощів, пов'язаних з глобальним керуванням станом.

Завдяки цим перевагам, компоненти в Angular значно спрощують процес розробки, роблячи його більш ефективним та організованим.

На рис 3.1 можна побачити як наочно виглядають компоненти безпосередньо в проекті.

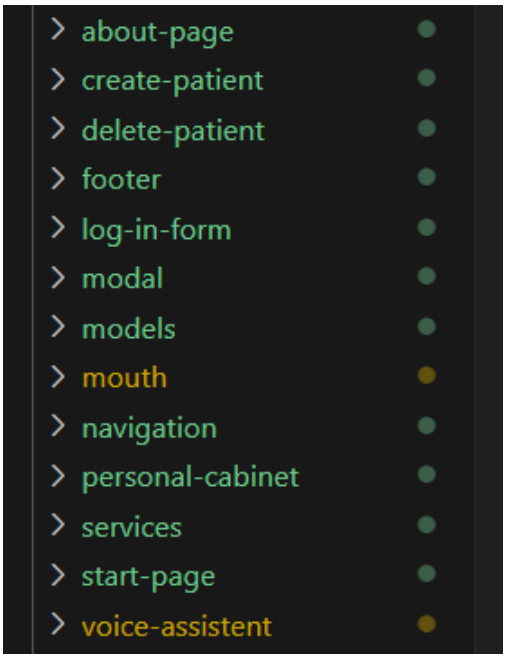

Рис. 3.1 Вигляд компонентів в проекті в редакторі коду VS Code

Важливо зазначити що папки models та services це не компоненти, в них містяться дещо інші файли, про які я напишу згодом.

Як я вже раніше казав, кожна з цих папок включає три файли це html, scss та ts файл з логікою нашого компонента. А викликати, тобто використати наш компонент можна просто викликавши його по назві як звичайний html тег, але варто пам'ятати, що кожен компонент повинен бути видимим для інших, тому його обов'язково потрібно підключити до app.module.ts або до файлу який виконує схожі функції.

```
<app-voice-assistent></app-voice-assistent>
```

Рис. 3.2 Приклад використання компонента в іншому компоненті

```
15  import { CreatePatientComponent } from './create-patient/create-patient.component';
16  import { DeletePatientComponent } from './delete-patient/delete-patient.component';
17  import { MatExpansionModule } from '@angular/material/expansion';
18
19  @NgModule({
20    declarations: [
21      VoiceAssistentComponent,
22      ModalComponent,
23      CreatePatientComponent,
24      DeletePatientComponent,
25    ],
```

Рис. 3.3 Приклад підключення компонентів до app.module.ts

3.2 Структура проекту

Як вже раніше зазначалось, проект складається переважно з компонентів, проте також існують деякі інші файли, такі як сервіси, моделі та директиви. В нашому проекті є такі компоненти:

- start-page, компонент який в основі свого контенту має стартову сторінку яку користувач бачитиме як тільки загрузить наш сайт, на цій сторінці присутні дві кнопки «Увійти» та «Про нас»
- about-page, компонент який в основі свого контенту має сторінку, на якій йдеться про те для кого наш сайт і чому він корисний
- log-in-form, компонент який в основі свого контенту має сторінку на якій користувач може увійти в систему під своїм особистим логіном та паролем, на ці два поля накинуті валідатори, тож ваша електронна

пошта повинна бути валідна, а ваш пароль містити як мінімум одну цифру і бути виключно англійськими літерами.

- navigation, компонент який реалізує навігаційну систему по нашому сайту за допомогою routing модуля. Це ще один файл в якому прописані стандартні шляхи для навігації сторінками, які можуть бути використані в проекті. Тобто якщо наприклад потрібне посилання на якийсь конкретний компонент який мені потрібно буде відкрити через клік на якусь кнопку, то я створюю path з ім'ям шляху і далі підключаю те посилання на компонент який мені потрібен. Тобто всі ці посилання я зможу використовувати будь де в проекті, зокрема і в navigation.component

Рис. 3.4 Частина коду файлу app-routing.module.ts

- footer, компонент який використовується на всьому сайті. Стандартний футер для сайтів, який винесений в окремий компент.
- personal-cabinet, компонент який в основі свого контенту має сторінку де користувач може переглядати своїх пацієнтів та їхні мінімальні дані.

В кожного пацієнта є своя карта клікнувши на яку відкривається інший компонент який відповідає за відображення зубної карти.

- modal, компонент який реалізує мінімальну стилістичну обгортку для компонента create-patient
- create-patient, компонент контент якого можна побачити клікнувши по голубій кнопці з плюсиком на сторінці персонального кабінету лікаря, або ж «Пацієнти». Це вікно в якому можна вписати дані про нового пацієнта: ім’я, мобільний номер та місто проживання та створити нового пацієнта в базі.
- delete-patient, компонент який можна викликати натиснувши на іконку «видалити» на картці клієнта, цей компонент дає можливість видалити непотрібного, або вже не актуального пацієнта з бази.
- mouth, основний компонент нашої програми, це знову ж таки нова сторінка, на яку можна потрапити клікнувши по особистій карті клієнта на сторінці персонального кабінету лікаря, або ж «Пацієнти». В цьому компоненті реалізована вся наша зубна мапа та радіо кнопки, аби перемикати режими редагування цієї зубної карти.
- voice-assistant, компонент який реалізує нашого голосового асистента, цей компонент використовується в компоненті mouth, де він і використовується.
- Сервіси, в моєму проекті є декілька сервісів. Основними цілями сервісів в Angular є організація та модуляризація коду для кращої підтримки, інкапсуляція складної бізнес-логіки подалі від компонентів та ефективне управління залежностями за допомогою Angular's DI-системи. Сервіси мають на меті централізовано керувати станом додатку та робити його доступним для різних частин програми, забезпечуючи послідовне управління даними та збереження стану. Вони також слугують посередниками для міжкомпонентної комунікації, забезпечуючи уніфікований інтерфейс для надсилання та прослуховування подій або змін стану. Крім того, сервіси

централізовано керують взаємодією API, оптимізуючи HTTP-запити і відповіді та забезпечуючи узгоджений інтерфейс для внутрішньої комунікації. Одним з таких сервісів є CommentsService в save-comment.service.ts - це Angular-сервіс, призначений для управління збереженням і завантаженням коментарів пацієнтів у веб-додатку.(рис. 3.5)

Рис. 3.5 Сервіс для збереження та завантаження коментарів

Метод saveComments зберігає масив коментарів до localStorage браузера для конкретного пацієнта. Для зберігання коментарів використовується localStorage.setItem з ключовим форматом patient_<patientId>_comments. Масив коментарів переводиться в рядок JSON за допомогою JSON.stringify() перед збереженням.

На а метод loadComments завантажує масив коментарів зі сховища localStorage для конкретного пацієнта. Він намагається отримати елемент з ключем patient_<patientId>_comments. Якщо коментарі існують, він розбирає рядок JSON назад у масив за допомогою JSON.parse() і повертає його. Якщо коментарів не знайдено, повертається порожній масив.

- Також в нашому проекті є одна модель, в Angular моделі полегшують зв'язування даних між компонентом і шаблоном. Вони слугують мостом, що дозволяє даним плавно перетікати від інтерфейсу користувача до логіки додатку і навпаки. Ось модель яка використовується в нашому проекті(рис. 3.6)

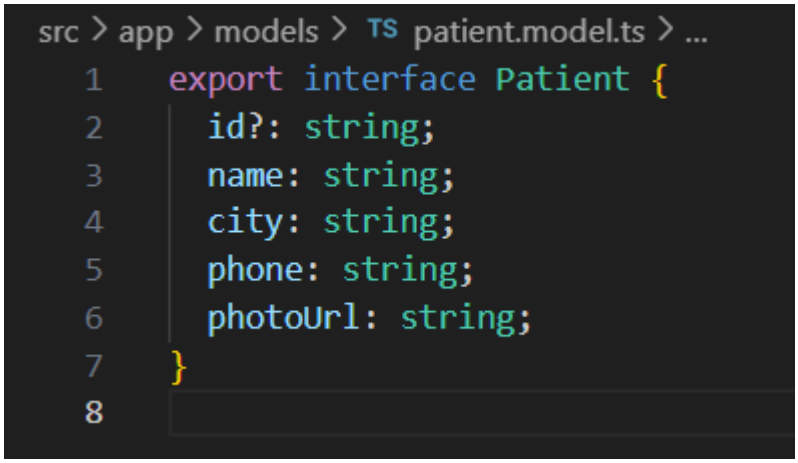

Рис. 3.6 Модель пацієнта

Ця модель використовується для представлення даних пацієнта у всьому додатку, гарантуючи, що всі компоненти та сервіси, які обробляють дані пацієнта, використовують однакову структуру.

При створенні або редагуванні інформації про пацієнта у формі ця модель допомагає прив'язати поля форми до відповідних властивостей в об'єкті пацієнта.

При отриманні даних про пацієнта з API ця модель може бути використана для перевірки типу відповіді та забезпечення відповідності даних очікуваній структурі.

- Також в нашому проекті є одна директива, яка називається ToothVisibilityDirective. Основне призначення директиви ToothVisibilityDirective - динамічно керувати видимістю елементів на основі булевої умови. Це особливо корисно в в нашому додатку, так як ця директива допомагає показувати або приховувати зуб залежно від дій користувача або інших змін стану.

src > app > TS tooth-visibility.directive.ts > ToothVisibilityDirective > updateVisibility

```typescript
import {
  Directive,
  ElementRef,
  Renderer2,
  Input,
  OnChanges,
  SimpleChanges,
} from '@angular/core';

@Directive({
  selector: '[appToothVisibility]',
})
export class ToothVisibilityDirective implements OnChanges {
  @Input('appToothVisibility') isVisible: boolean = true;
  @Input() actionMode: string = '';

  constructor(private el: ElementRef, private renderer: Renderer2) {}

  ngOnChanges(changes: SimpleChanges) {
    if (changes['isVisible']) {
      this.updateVisibility();
    }
  }

  private updateVisibility() {
    if (this.isVisible) {
      this.renderer.setStyle(this.el.nativeElement, 'opacity', '1');
    } else {
      this.renderer.setStyle(this.el.nativeElement, 'opacity', '0');
    }
  }
}

```

Рис. 3.7 ToothVisibilityDirective

В коді видно що до елемента на який буде примінено цю директиву буде накинуто стиль, за умови якщо зуб видно, то opacity:1, тобто прозорості нема, елемент видно. В іншому випадку коли ж зуб перейде в режим невидимий, прозорість стане 0.

```
[appToothVisibility]="!activeParts['tooth16']['removed']"
```

Рис. 3.8 Приклад застосування appToothVisibility

3.3 Реалізація зубної мапи

В компоненті mouth було створено декілька контейнерів в html, для зубів які ми туди мали б вставити. Зуби вставляються в форматі svg. Для двох окремих щелеп було створено два окремі контейнери. Також в кожної зі щелеп є зуб з виглядом зверху та збоку.

Візьмемо для прикладу шістнадцятийтий зуб, і на його прикладі розглянемо як все працює

```
<svg
  class="tooth16"
  width="153"
  height="215"
  viewBox="0 0 153 215"
  fill="none"
  xmlns="http://www.w3.org/2000/svg"
  (click)="
  togglePart('tooth16',actionMode === 'showKaries'? 'karies': actionMode === 'plomba'? 'plomba': 'removed')"
  [appToothVisibility]="!activeParts['tooth16']['removed']"
>
```

Рис. 3.9 svg з click подією та використанням директиви tooth-visibility.directive

Коли на SVG клацають, викликається togglePart з параметрами: 'tooth16': відповідний ідентифікатор зуба.

Рядок, що визначає, яку дію виконати на основі поточного actionMode.

- 'karies', якщо showKaries є істинним.
- 'plomba', якщо plomba істинна.
- 'removed' в іншому випадку.

Директива appToothVisibility керує видимістю зуба на основі того, чи позначений він як видалений в activeParts['tooth16']['removed'].

[class.active]: Додає активний клас до елемента, якщо activeParts['tooth16']['left'] має значення true.

[class.clickable]: Додає клас clickable до елемента, якщо showKaries має значення true, вказуючи на те, що він доступний для взаємодії в цьому режимі.

(click) Подія: При натисканні на елемент шляху викликає togglePart з параметрами 'tooth16' і 'left', впливаючи на ліву частину зуба і якщо режим «Карієс» увімкнено то ліва частина буде зафарбована в інший колір.

[attr.pointer-events]: Встановлює атрибут pointer-events у значення 'auto', якщо showKaries є істинним, дозволяючи елементу бути доступним для кліку. У протилежному випадку встановлює значення 'none', роблячи елемент не взаємодіючим.

[attr.stroke-width]: Встановлює атрибут stroke-width у '0.5', якщо showKaries має значення true, роблячи обведення видимим. В іншому випадку встановлює значення '0', роблячи обведення невидимим.

```
<path
  class="part-of-tooth"
  d="M86.3586 24.1524C85.0953 29.1924 82.9477 39.6504 84.463
  stroke="#606060"
  [class.active]="activeParts['tooth16']['center']"
  [class.clickable]="showKaries"
  (click)="togglePart('tooth16', 'center')"
  [attr.pointer-events]="showKaries ? 'auto' : 'none'"
  [attr.stroke-width]="showKaries ? '0.5' : '0'"
/>
```

Рис. 3.10 Ліва частина зуба та його функціонал

Метод ngOnInit ініціалізує об'єкт activeParts для кожного зуба і завантажує збережені дані зі сховища localStorage, якщо вони доступні.

```
ngOnInit() {
  this.route.params.subscribe((params) => {
    const patientId = params['id'];
    for (let i = 1; i <= 32; i++) {
      this.activeParts[`tooth${i}`] = {
        left: false,
        right: false,
        center: false,
        top: false,
        bottom: false,
        removed: false,
        karies: false,
        plomba: false,
      };
    }
    const savedData = localStorage.getItem(`patient_${patientId}_mouth_data`);
    if (savedData) {
      this.activeParts = JSON.parse(savedData);
    }

    const savedComments = localStorage.getItem(
      `patient_${patientId}_comments`
    );
    if (savedComments) {
      this.comments = JSON.parse(savedComments);
    }
  });

  this.onActionModeChange();
}
```

Рис. 3.11 Метод ngOnInit

Також цей метод генерує об’єкт з 32 зубам, для кожного з якого приписані потрібні параметри: виділення частин зубу в режимі виділення карієсу, видалення зуба, увімкнення режиму карієсу з поділом на частини, та пломба.

Метод onActionModeChange оновлює булеві прапори (showKaries, removeTooth, restoreTooth, plomba) на основі вибраного перемикача (actionMode).

```
57  onActionModeChange() {
58    this.showKaries = this.actionMode === 'showKaries';
59    this.removeTooth = this.actionMode === 'removeTooth';
60    this.restoreTooth = this.actionMode === 'restoreTooth';
61    this.plomba = this.actionMode === 'plomba';
62  }
```

Рис. 3.12 Метод onActionModeChange

ActionMode визначає, яка операція є активною: показати карієс, видалити зуб, реставрувати зуб або додати пломбу. Коли actionMode змінюється, відповідні булеві прапорці оновлюються, щоб відобразити поточну дію.

Метод togglePart обробляє кліки на різних частинах зуба. Він оновлює об'єкт activeParts на основі поточного actionMode.

- Якщо removeTooth має значення true, він позначає зуб як видалений.
- Якщо restoreTooth має значення true, він відновлює видалений зуб.
- Якщо showKaries має значення true, то вмикає видимість карієсу на вказаній ділянці зуба.
- Якщо plomba має значення true, перемикає наявність пломби на зубі.

```
togglePart(toothNumber: string, part: string) {
  if (this.removeTooth) {
    this.activeParts[toothNumber]['removed'] = true;
  } else if (this.restoreTooth) {
    if (this.activeParts[toothNumber]['removed']) {
      this.activeParts[toothNumber]['removed'] = false;
    }
  } else if (this.showKaries) {
    this.activeParts[toothNumber][part] =
      !this.activeParts[toothNumber][part];
  } else if (this.plomba) {
    this.activeParts[toothNumber]['plomba'] =
      !this.activeParts[toothNumber]['plomba'];
  }
  this.saveData();
}
```

Рис. 3.13 Метод togglePart

3.4 Реалізація голосового асистенту

Наш голосовий асистент працює на базі webkitSpeechRecognition [9] [10]. webkitSpeechRecognition - це веб-інтерфейс API, що надається WebKit, рушієм браузера, який використовується в Safari, а раніше в Chrome. Він є частиною Web Speech API, який включає в себе можливості як для розпізнавання, так і для синтезу мовлення. Мета webkitSpeechRecognition - дозволити веб-програмам перетворювати мовленнєвий ввід у текст, що дозволяє створювати голосові інтерфейси.

Ось як він працює: для початку потрібно ініціалізувати webkitSpeechRecognition.

```
ngOnInit() {
  const SpeechRecognition = (window as any).webkitSpeechRecognition;
  const recognition = new SpeechRecognition();
```

Рис. 3.14 Ініціалізація webkitSpeechRecognition

Далі нам потрібно, аби наш голос якось вловлювався, реалізували це ми за допомогою функції onresult. Ця функція перехоплює голосовий ввід і

обробляє його, перетворюючи транскрипт у нижній регістр і передаючи його методу speakThis.

```
16    recognition.onresult = (event: any) => {
17      const current = event.resultIndex;
18      const transcript = event.results[current][0].transcript;
19      this.speakThis(transcript.toLowerCase());
20    };
```

Рис. 3.15 Функція onresult

Налаштовано кнопку, яка запускає процес розпізнавання мови при натисканні

```
const btn = document.querySelector('.talk') as HTMLElement;
btn.addEventListener('click', () => {
  this.startSpeechRecognition(recognition);
});
```

Рис. 3.16 Налаштування кнопки

Метод startSpeechRecognition запускає процес розпізнавання і обробляє результати асинхронно:

```
183  private async startSpeechRecognition(recognition: any) {
184    try {
185      recognition.start();
186      const result = await this.waitForSpeechRecognition(recognition, 10000);
187      if (result) {
188        const transcript = result[0].item(0).transcript.toLowerCase();
189        this.speakThis(transcript);
190      }
191    } catch (error) {
192      console.error('Error starting speech recognition:', error);
193    }
194  }
```

Рис. 3.17 Метод startSpeechRecogni

Метод waitForSpeechRecognition очікує на результати розпізнавання або завершує роботу через певний проміжок часу:

```
196   private waitForSpeechRecognition(
197     recognition: any,
198     timeout: number
199   ): Promise<SpeechRecognitionResult[] | null> {
200     return new Promise((resolve, reject) => {
201       let timerId: any;
202
203       const onResult = (event: any) => {
204         clearTimeout(timerId);
205         recognition.onresult = null;
206         resolve(event.results);
207       };
208
209       recognition.onresult = onResult;
210
211       timerId = setTimeout(() => {
212         recognition.onresult = null;
213         resolve(null);
214       }, timeout);
215     });
216   }
```

Рис. 3.18 Метод waitForSpeechRecognition

Метод speak виставляє параметри нашого асистента: гучність, мову і тому подібне

```
speak(sentence: string) {
  const text_speak = new SpeechSynthesisUtterance(sentence);
  text_speak.lang = 'uk-UA';
  text_speak.rate = 1;
  text_speak.pitch = 1;
  window.speechSynthesis.speak(text_speak);
}
```

Рис. 3.19 Метод speak

В методі speakThis в об’єктах записані всі важливі слова які потрібні для того аби голосовий асистент міг краще розуміти користувача, записані синоніми слів які можуть використовуватись

```
speakThis(message: string) {
  const speech = new SpeechSynthesisUtterance();
  speech.lang = 'uk-UA';
  let finalText = '';

  const parts: { [key: string]: string } = {
    зліва: 'left',
    зправа: 'right',
    'по центрі': 'center',
    зверху: 'top',
    знизу: 'bottom',
    'по центру': 'center',
    середина: 'center',
    посередині: 'center',
    справа: 'right',
  };

  const numbers: { [key: string]: number } = {
    перший: 1,
    першому: 1,
    другий: 2,
    другому: 2,
    третій: 3,
    третьому: 3,
    четвертий: 4,
    четвертому: 4,
    "п'ятий": 5,
```

Рис. 3.20 Метод speakThis та об’єкти в яких записані ключові слова

Далі в методі speakThis перевіряється вже оброблений текст який сказав користувач, якщо те що ви сказали співпадає з тим що очікує код, то виконуються певний функціонал, ось наприклад рис. 3.13 демонструє нам реакцію на слово видалити, і якщо зі словом «видалити» буде номер зуба, то радіо кнопка в компоненті mouth автоматично перемкнеться на ту що видаляє зуб за допомогою наступних рядків коду перших двох рядків коду, а за допомогою наступного, третього, вже зуб відповідно видаляється

```
} else if (message.includes('видалити') && toothNumber !== null) {
  this.mouthComponent.actionMode = 'removeTooth';
  this.mouthComponent.onActionModeChange();
  this.mouthComponent.togglePart(`tooth${toothNumber}`, 'removed');
```

Рис. 3.21 Приклад реагування на слово «видалити» голосового асистента

Метод togglePart це частина компоненту mouth який дозволяє нам відслідкувати на яку частину зуба ми натиснули, чи який зуб ми хочемо видалити, чи відновити, або поставити пломбу

3.5 Інструкція з використання продукту

Як тільки ви запустите проект, ви побачите головну сторінку нашого сайту де зможете натиснути на кнопку «Увійти» та «Про нас», а в навігаційній панелі можна знову ж таки перейти на сторінку «Про нас» та на сторінку з картками пацієнтів.

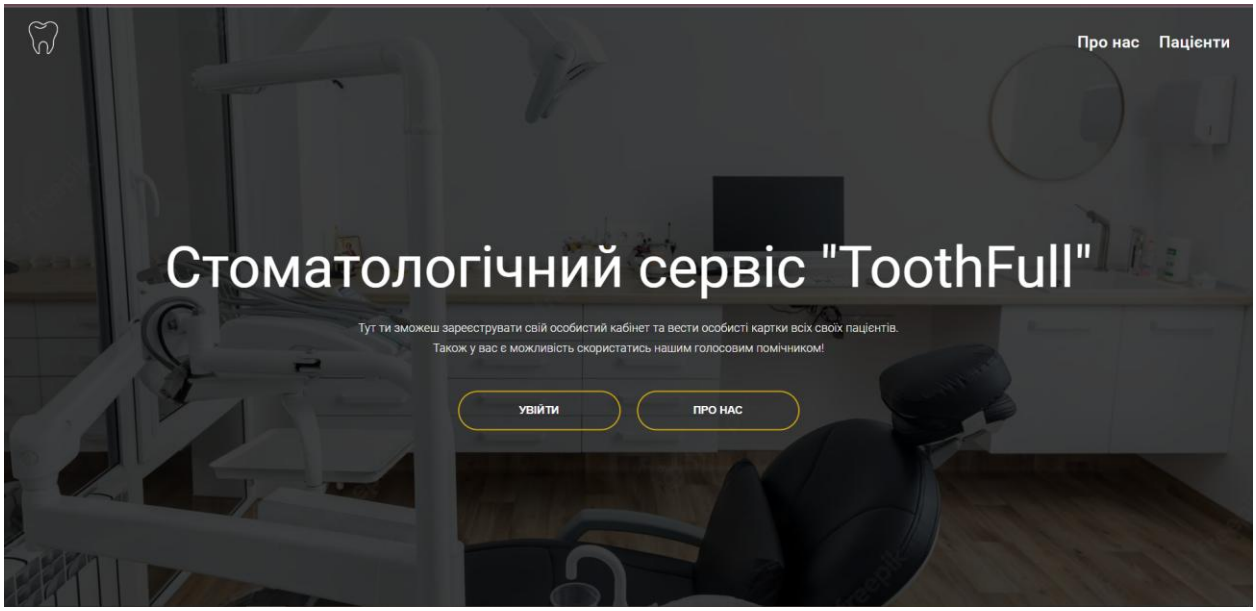

Рис. 3.22 Домашня сторінка

Натиснувши на кнопку увійти, ви потрапите на так звану log-in сторінку, де зможете протестувати цю форму, яка, нагадаю, працює без бекенду.

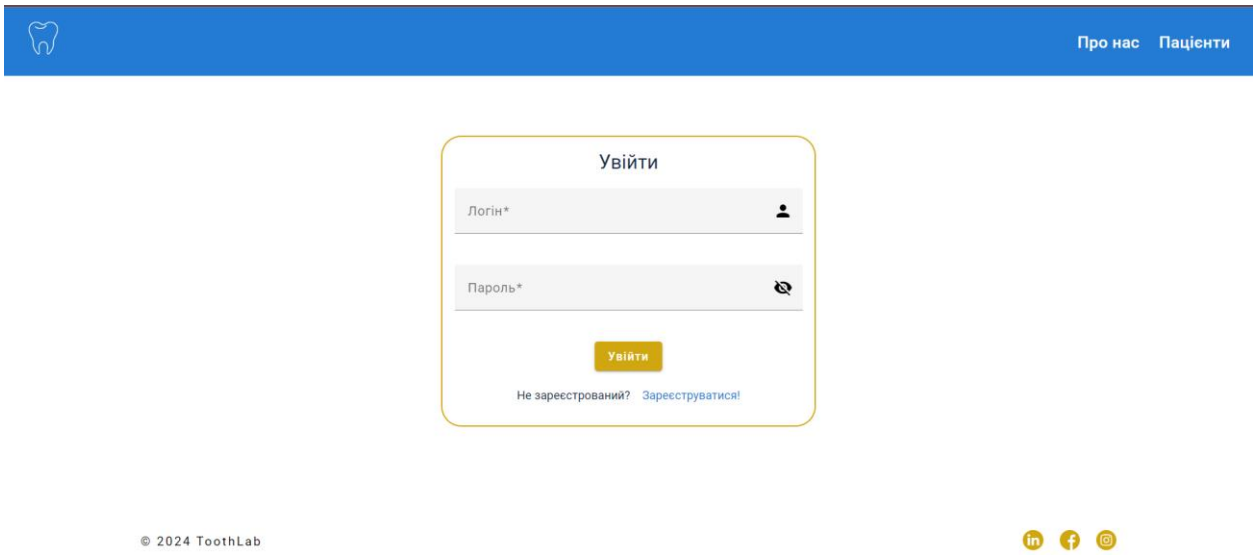

Рис. 3.23 Форма для реєстрації лікаря в системі

Також, форма має валідатори на логін і пароль, які ви можете побачити на рис. 3.17

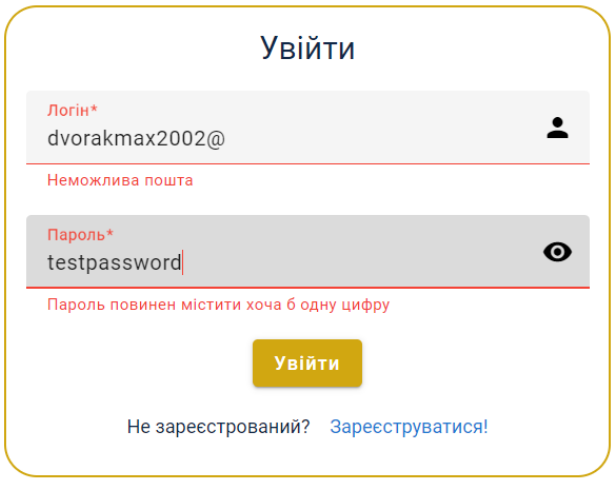

Рис. 3.24 Демонстрація валідацій

При натисканні на кнопку «Про нас» на домашній сторінці ви потрапите на сторінку яка описує сутність та інноваційність нашого сайту.

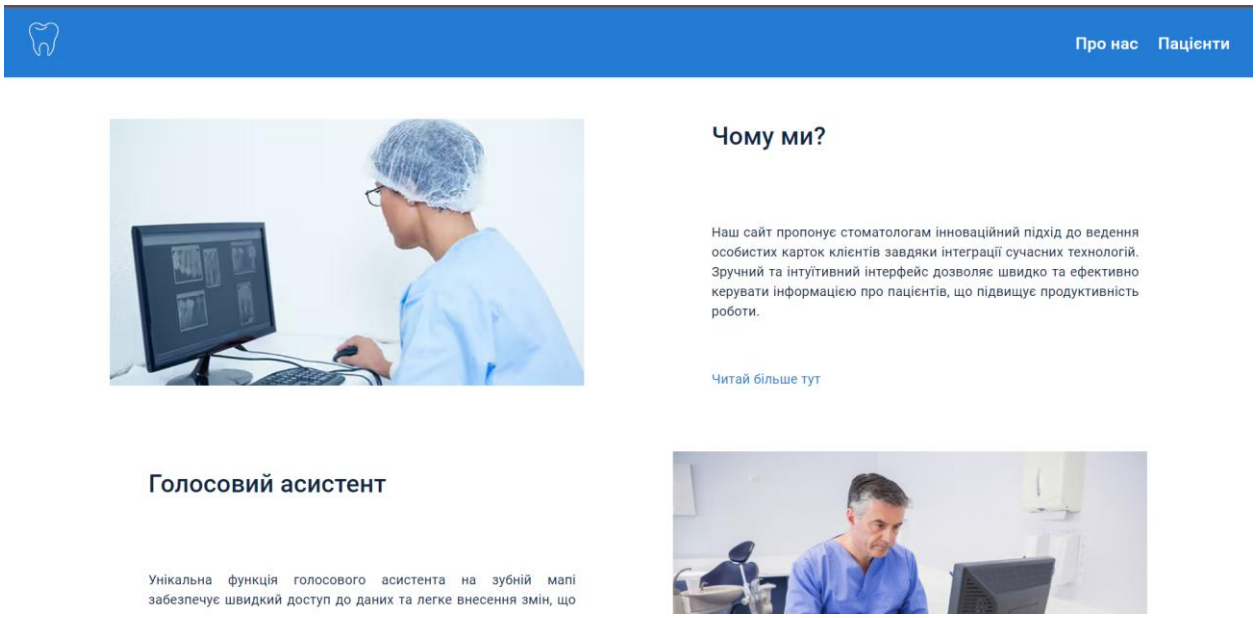

Рис. 3.25 Сторінка «Про нас»

Далі якщо клікнути на вкладку «Пацієнти» на навігаційній панелі, то ви перейдете на сторінку з особистими картками зареєстрованих клієнтів.

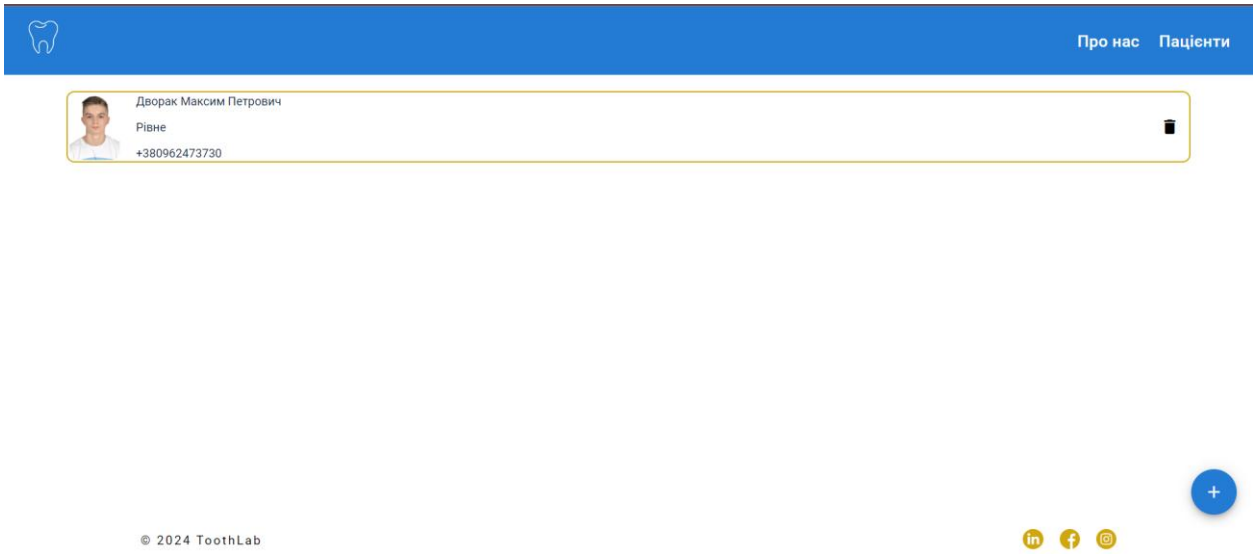

Рис 3.26 Особистий кабінет, або ж сторінка з особистими картками клієнта

Далі ми можемо натиснути на кнопку з плюсом, аби створити нового пацієнта

Рис. 3.27 Кнопка для створення нового пацієнта

Тоді нам відкриється наступне вікно з реєстрацією нового пацієнта. Тут ми може записати деякі дані про нового пацієнта та створити його унікальну зубну мапу.

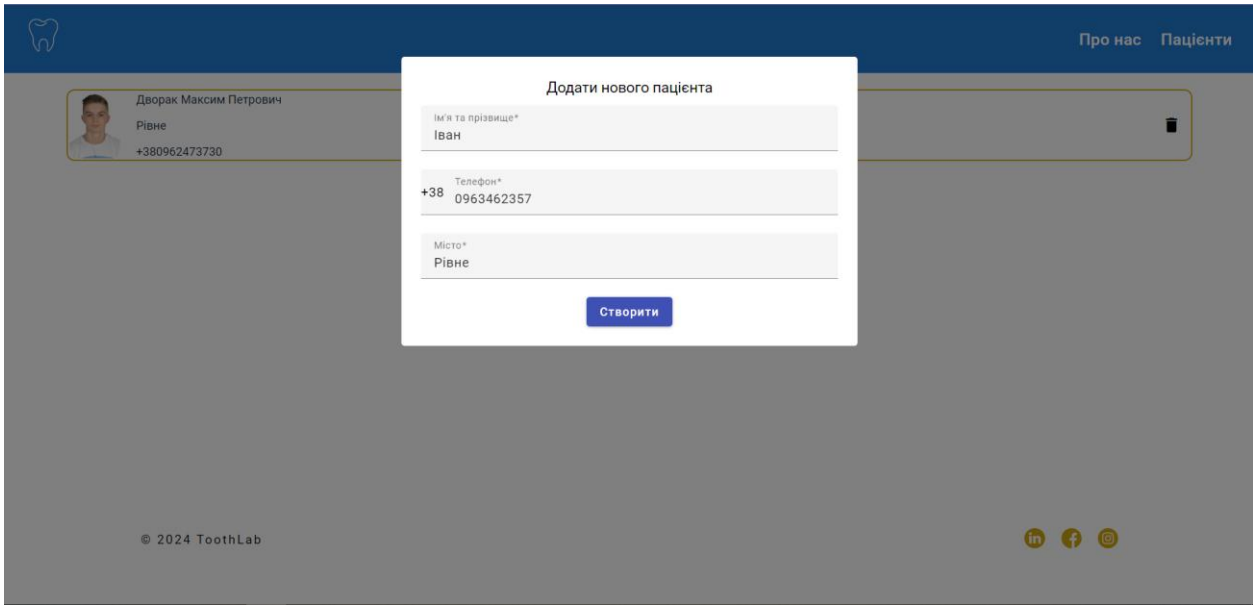

Рис. 3.28 Додавання нового пацієнта

Після цього ми бачимо що пацієнт створився

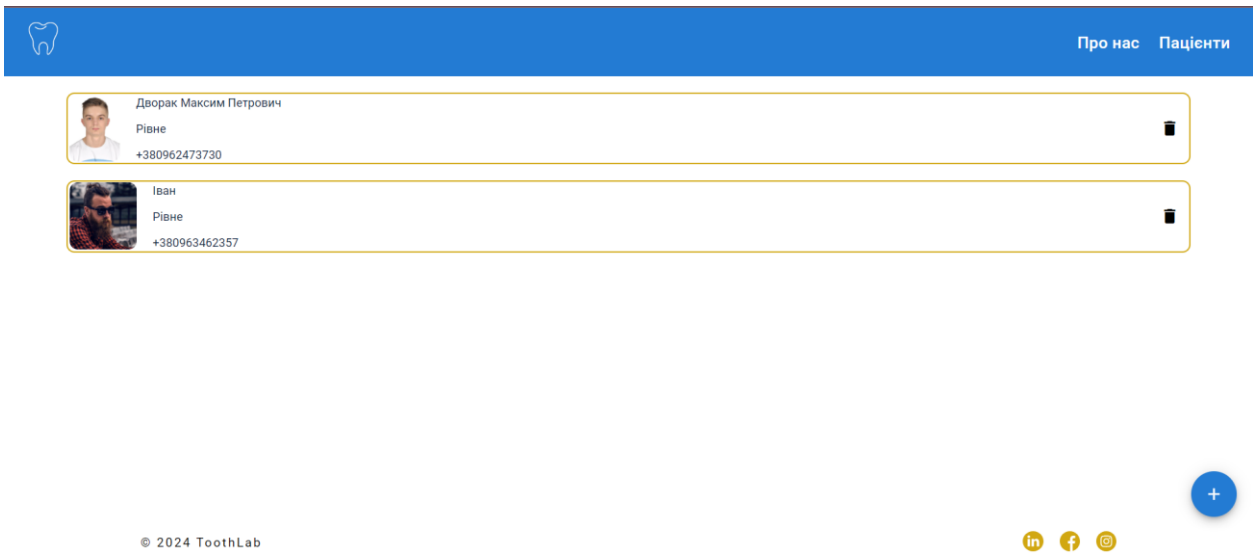

Рис. 3.29 Нового пацієнта додано

Далі ми можемо натиснути на особисту картку клієнта і побачити його зубну мапу та коментарі які поки ще ніхто не залишав

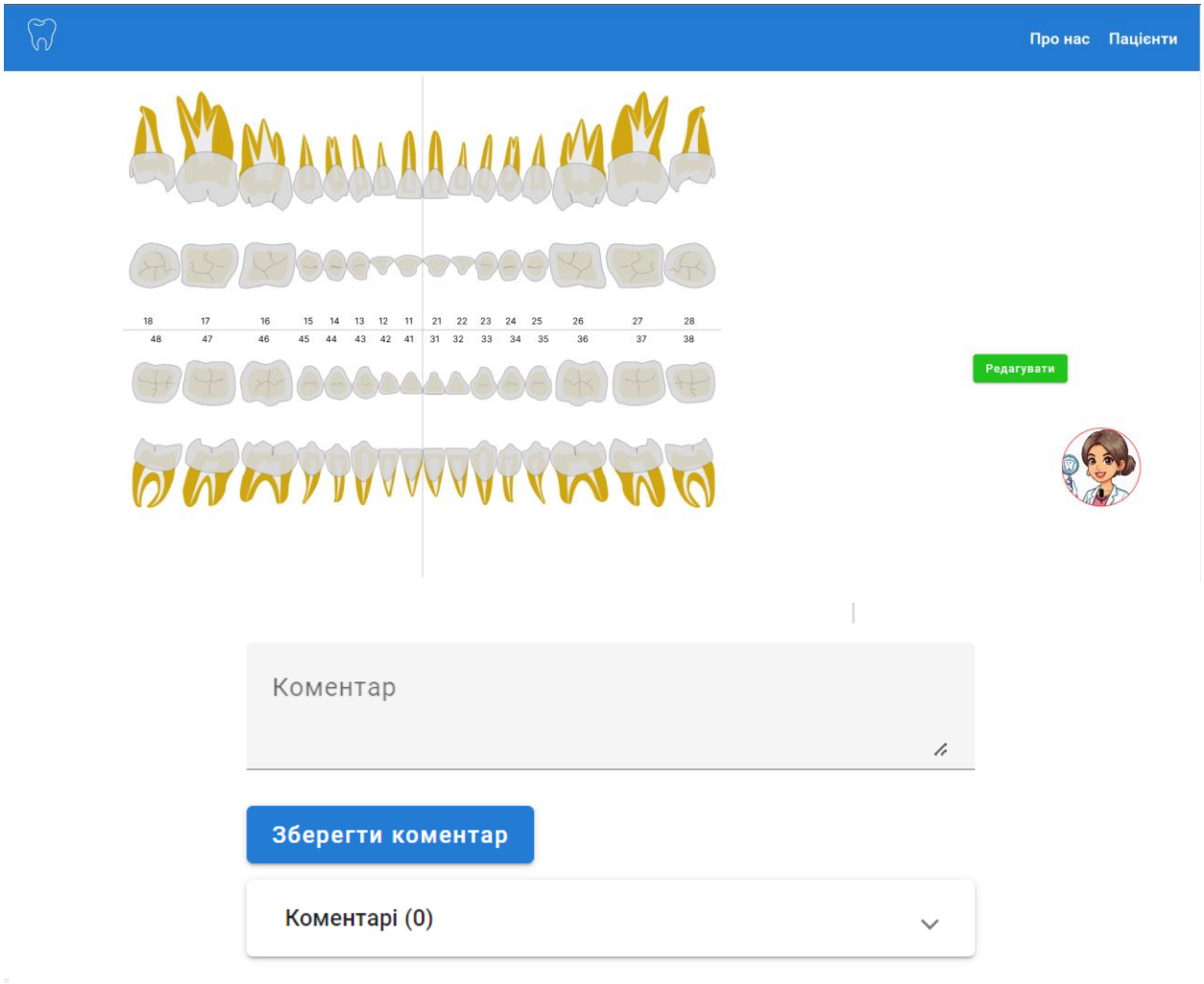

Рис. 3.30-31 Персональна зубна мапа та поле для введення коментарів

Натиснувши на кнопку «Редагувати» ми зможемо побачити 5 радіо кнопок, кожна з яких виконує свою функцію.

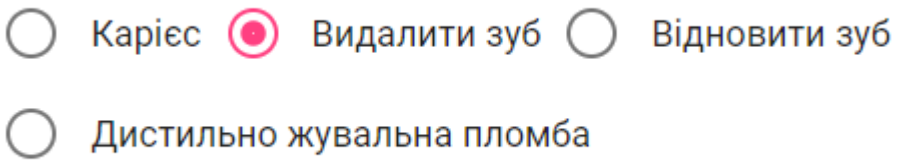

Рис. 3.32 Радіо кнопки

При активній радіо кнопці «Видалити зуб» користувач може видалити певний зуб з мапи.

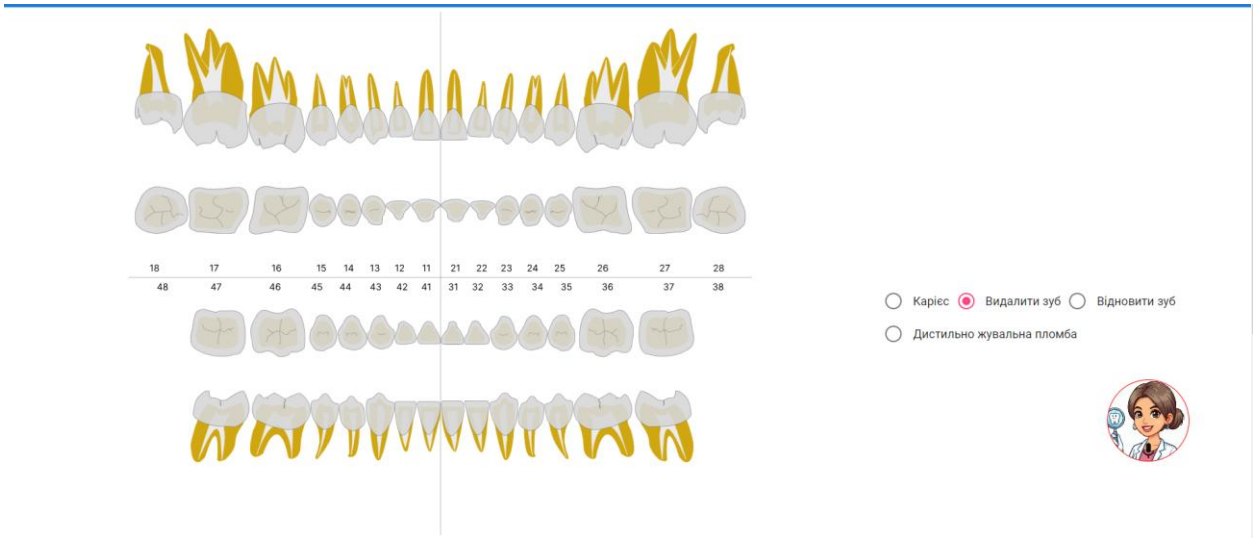

Рис. 3.33 Видалено перший та шістнадцятий зуби

Для того щоб відновити ці зуби потрібно увімкнути кнопку «Відновити зуб» та клацнути по пустому місцю, де не вистачає зуба.

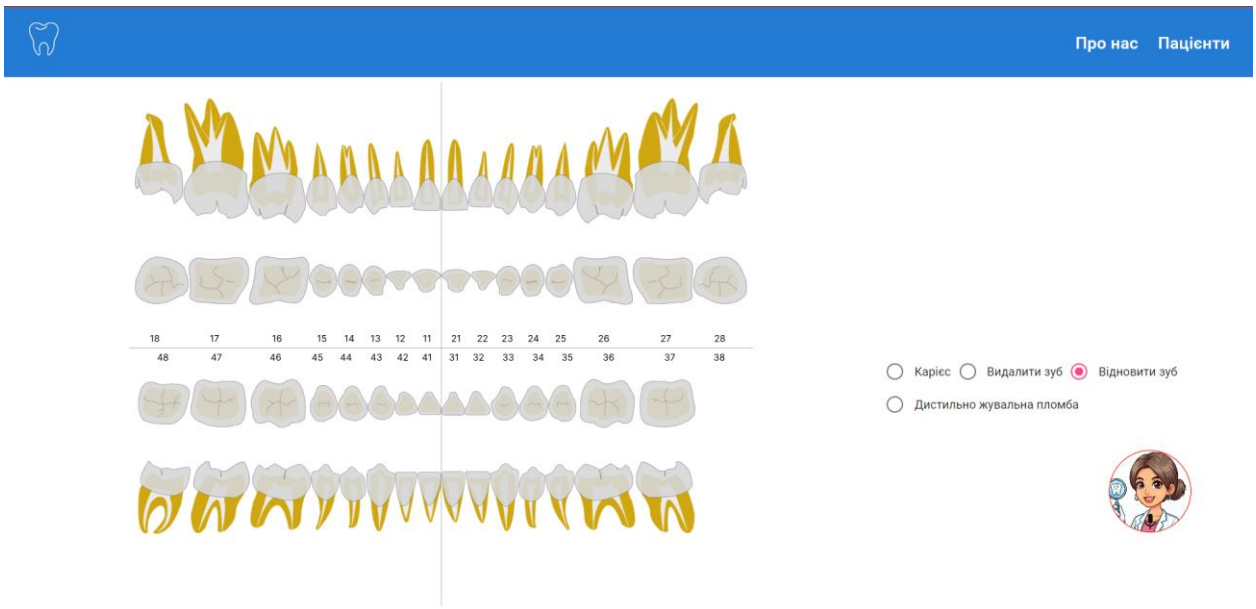

Рис. 3.34 Відновлений перший зуб

Щоб позначити на зубах карієс потрібно перемкнутись на кнопку «Карієс» та виділити потрібні частини потрібно зуба

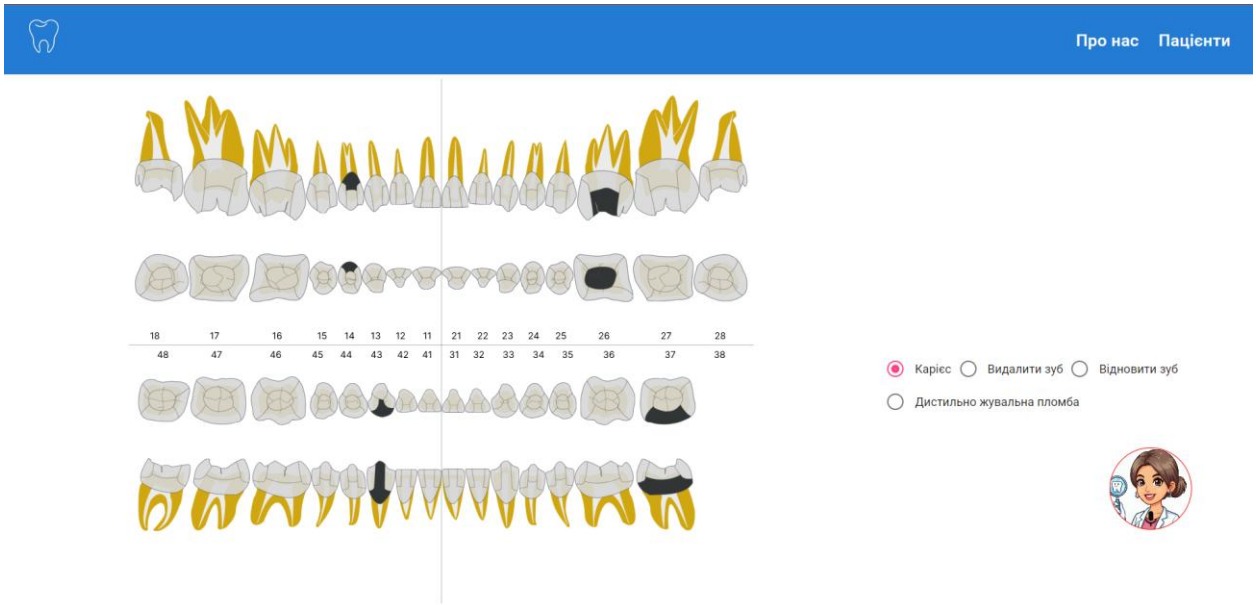

Рис. 3.35 Виділено карієс на декількох зубах

Ну і якщо користувач бажає позначити ще дистильно жувальну пломбу в пацієнта, то йому потрібно активувати однойменну кнопку і обрати потрібний зуб.

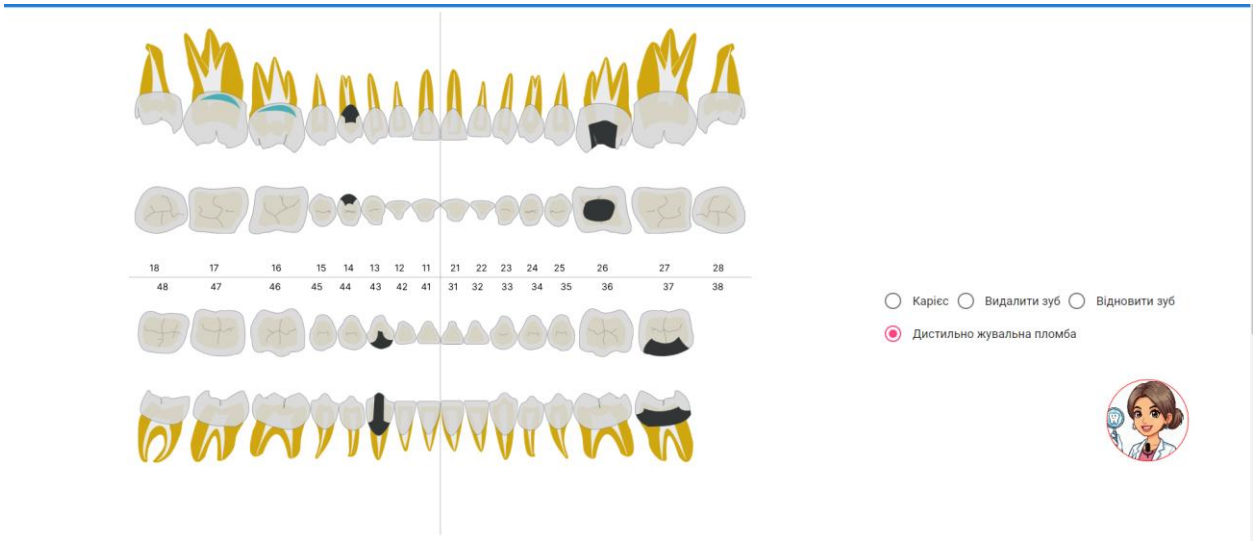

Рис. 3.36 Позначено два зуби в яких є дистильно жувальна пломба

Також лікар може залишити якийсь особливий коментар після візиту пацієнта записавши його у відповідне поле.

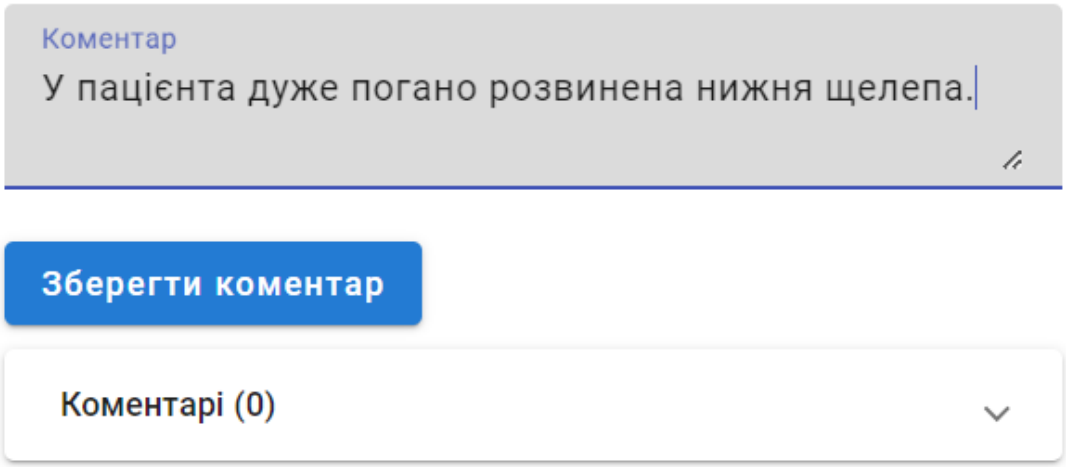

Рис. 3.37 Запис коментаря про пацієнта лікарем

Після того як користувач натисне кнопку «Зберегти коментар» Коментар збережеться до секції про коментарі.

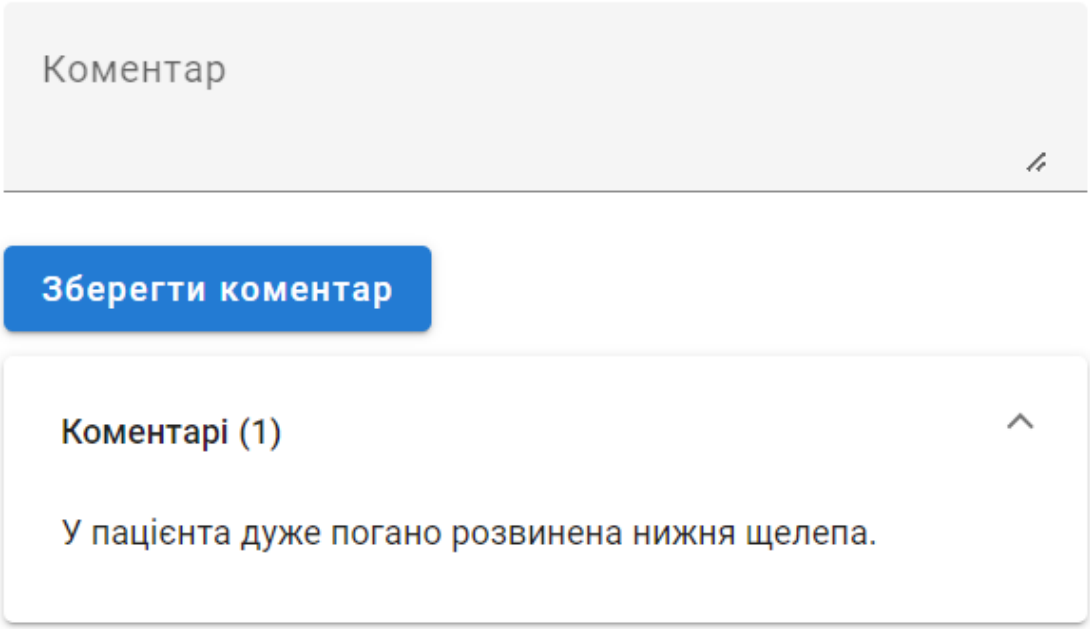

Рис. 3.38 Збережений коментар

І звичайно, все те ж саме, що я щойно демонстрував, робивши все вручну, можна зробити за допомогою голосового асистента. Для того щоб видалити зуб достатньо просто натиснути на кнопку голосового асистента яка знаходиться у нижньому правому куті та сказати «Видалити десятий зуб» і його буде видалено, або ж якщо ви хочете позначити карієс на зубі то потрібно спочатку сказати «Виділити карієс», аби увімкнути сегментацію зубів для

карієсу, а тоді сказати «Карієс п’ятий зуб посередині» і цю частину буде виділено.

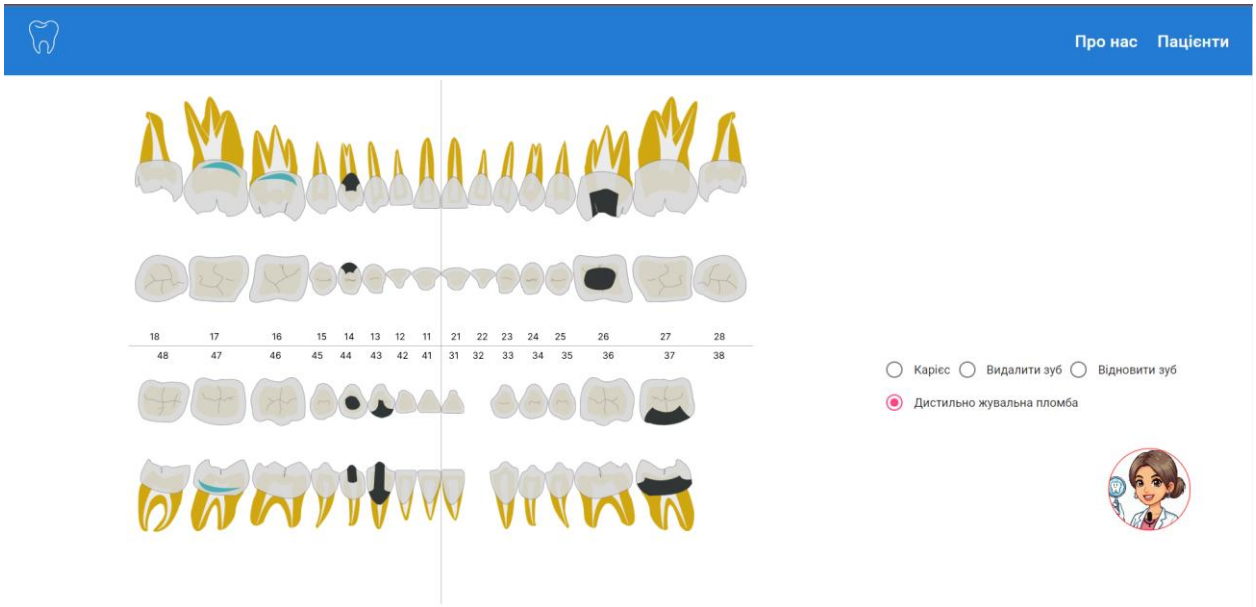

Рис. 3.39 Відредагована зубна мапа за допомогою голосового асистента

Також за допомогою голосу можна залишати і коментарі

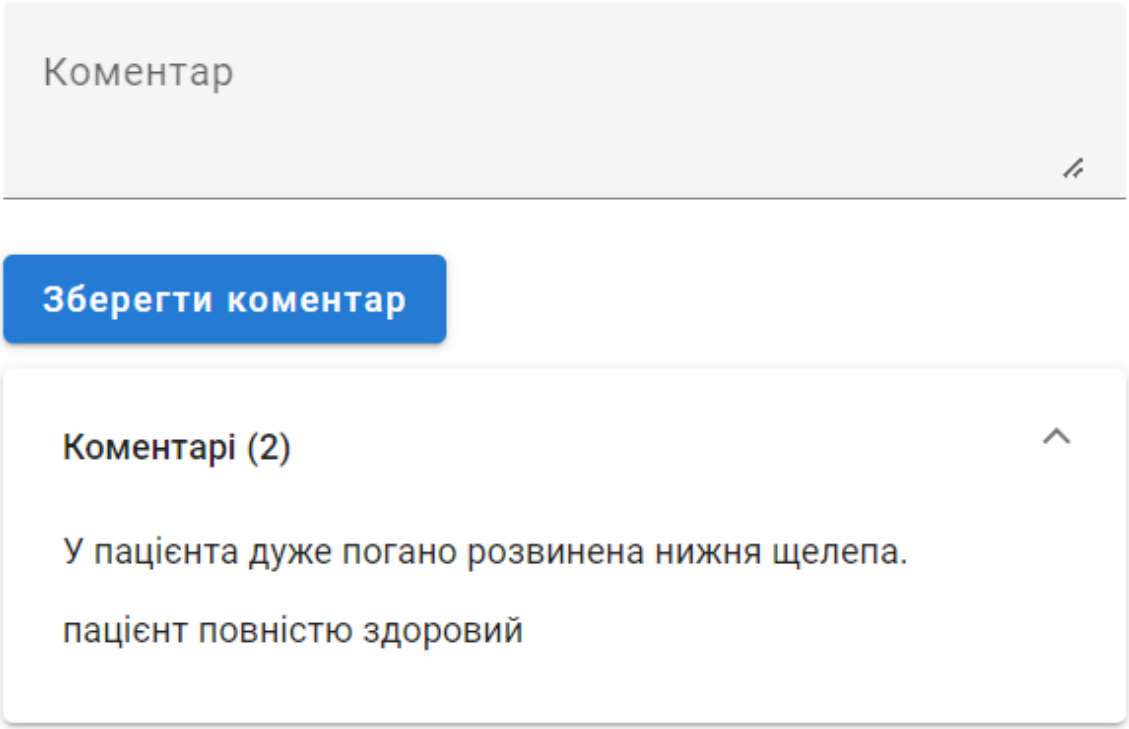

Рис. 3.40 Приклад коментаря залишеного голосом

3.6 Тестування продукту

При розробці проекту виникало багато проблем та багів, які мені успішно вдалось виправити. Однією з таких проблем, була неможливість повернути зуб, а саме зробити його видимим якщо до того його було видалено, проблема була в тому що я використовував такий css параметр як visibility нажаль він чомусь працював не коректно на відновлення зуба, видалявся він добре, а от навпаки ні. Якщо я встановлював стилі безпосередньо у стилях файлу scss то visibility не працювало взагалі, тому було прийнято рішення створити директиву tooth-visibility.directive (рис. 3.9) та встановлювати стилі безпосердньо через директиву та методу setStyle(). Також я замінив змінну visibility на opacity тепер зуб з'являється добре.

Проте виникла інша проблема, коли зуб в коді видалений то через те, що ми використовували властивість opacity було можливо взаємодіяти з цим зубом коли він зниклий, тобто можна було увімкнути скажімо режим «Дистильно жувальна пломба» і клікнувши по тому місцю де був зуб ми встановлювали туди пломбу, хоча пломба на видалений зуб встановлюватись не має. Було прийнято рішення видаляти будь який активний стан зуба якщо зуб було видалено, в такому випадку якщо зуб було видалено, то немає сенсу більше зберігати якусь інформацію про нього, і при відновленні зуба карієс та пломби скидаються. Тоді не важливо чи буде користувач випадково натискати на видалений зуб бажаючи туди поставити, скажімо, пломбу, чи ні при відновленні всі частини зуба повертаються в статус false.

```
togglePart(toothNumber: string, part: string) {
  if (this.removeTooth) {
    this.activeParts[toothNumber]['removed'] = true;

  } else if (this.restoreTooth) {
    if (this.activeParts[toothNumber]['removed']) {
      this.activeParts[toothNumber]['removed'] = false;
      this.activeParts[toothNumber]['plomba'] = false;
      this.activeParts[toothNumber]['bottom'] = false;
      this.activeParts[toothNumber]['left'] = false;
      this.activeParts[toothNumber]['right'] = false;
      this.activeParts[toothNumber]['top'] = false;
      this.activeParts[toothNumber]['center'] = false;
    }
  } else if (this.showKaries) {
    this.activeParts[toothNumber][part] = !this.activeParts[toothNumber][part];
  } else if (this.plomba) {
    if (!this.activeParts[toothNumber]['removed']) {
      this.activeParts[toothNumber]['plomba'] = !this.activeParts[toothNumber]['plomba'];
    }
  }
  this.saveData();
}
```

Рис. 3.41 Відредагований метод togglePart

ВИСНОВКИ

У цій роботі було розроблено комплексну систему для підвищення ефективності та взаємодії зубного лікаря з пацієнтами завдяки впровадженню інтерактивної зубної карти та голосового асистента в наш сайт. Використовуючи сучасні веб-технології, такі як AngularJS та Figma, цей проект успішно інтегрує складні функціональні можливості у зручний інтерфейс.

Наукова цінність цього дослідження полягає в інноваційному підході до оцифрування та впорядкування стоматологічних записів та взаємодії. Практичні наслідки є значними, пропонуючи інструмент, який дозволяє стоматологам швидко і точно оновлювати інформацію про пацієнта і взаємодіяти з цифровою картою зубів за допомогою голосових команд. Цей метод значно скорочує час, що витрачається на ручне введення даних, і мінімізує ймовірність помилок, тим самим підвищуючи загальну якість обслуговування пацієнтів.

У порівнянні з іншими відомими результатами, ця система виділяється завдяки унікальному поєднанню інтерактивної карти зубів і голосового асистента. Хоча інші системи управління стоматологією пропонують різні функціональні можливості, жодна з них не забезпечує такого ж рівня взаємодії та зручності в режимі реального часу. Особливої уваги заслуговує використання голосового асистента в цій системі, оскільки він дозволяє стоматологам працювати без допомоги рук, підвищуючи таким чином ефективність і гігієну під час процедур.

Методологія, застосована в цій роботі, включала розробку карти зубів у Figma, а потім перетворення цього дизайну в код за допомогою AngularJS. Такий підхід гарантував, що кінцевий продукт буде і візуально привабливим, і функціонально надійним. Система була ретельно протестована, щоб переконатися, що вона відповідає практичним потребам стоматологів.

Майбутні розробки цієї системи можуть включати розширення її функціональних можливостей за рахунок включення більш просунутих функцій, інтеграція з іншими системами медичних записів та обробка більш складних стоматологічних запитів. Це ще більше підвищить її корисність і забезпечить, щоб вона залишалася на передовій технологій управління стоматологічною практикою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Healthcare Web Development in 2024: Everything You Need to Know to Become a Leader on the Market. URL: https://qarea.com/blog/healthcare-web-development/ (дата звернення: 25.05.2024)
2. Актуальність Web-розробки у 2024 році. URL: https://it-rating.ua/aktualnist-web-rozrobki-u-2024-rotsi (дата звернення: 26.05.2024)
3. Попередній перегляд застовованого FxLayout до елементів. URL: https://tburleson-layouts-demos.firebaseapp.com/#/docs (дата звернення 19.05.2024)
4. Офіційний сайт Angular Material URL: https://v15.material.angular.io/ (дата звернення 22.05.2024)
5. AngularJS Up & Running. Shyam Seshadri & Brad Green. URL: https://pepa.holla.cz/wp-content/uploads/2015/10/AngularJS-Up-and-Running.pdf (дата звернення: 28.05.2024)
6. Angular Crash Course 2024. URL: https://www.youtube.com/watch?v=f7BJFTEbc10&t=1s (дата звернення 10.05.2024)
7. Designing and Prototyping Interfaces with Figma. Second edition. Fabio Staino. URL: https://books.google.com.ua/booksid=7zvrEAAAQBAJ&pg=PR7&hl=uk&source=gbs_selected_pages&cad=1#v=onepage&q&f=false (дата звернення: 15.05.2024)
8. Мій проект з готовою зубною мапою на Figma. URL: https://www.figma.com/design/liER3HwXvsEriLScWbtQa5/Max-Dvorak's-team-library?node-id=0-1&t=rWbLN6VjYMvsnKCL-0 (дата звернення 01.05.2024)
9. Build a Virtual Assistant using Javascript. URL: https://www.youtube.com/watch?v=U-mDg7QgOJo (дата звернення 18.05.2024)

10. Керівництво з використання Web Speech API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API (дата звернення 18.05.2024)

ДОДАТКИ

Програмний код