МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

БУТЛЕРСЬКИЙ ДМИТРО СЕРГІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
к. т. н., доцент
_____ О. В. Зелінська
«___»_____20___р.

ВЕБ-ДОДАТОК ДЛЯ ВЕДЕННЯ ЕЛЕКТРОННОГО РЕЄСТРУ ПАЦІЄНТІВ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
П. К. Ніколюк, д-р фіз.-мат. наук,
професор кафедри інформаційних технологій
_____

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

Вінниця – 2024

АНОТАЦІЯ

Бутлерський Д.С. Веб-додаток для ведення електронного реєстру пацієнтів. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні технології обробки даних», Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (магістерській) роботі досліджена проблема ведення електронного реєстру пацієнтів. Проаналізовані існуючі веб-додатки для вирішення даної проблеми, розроблений веб-додаток на основі цих даних, що вирішує поставлену задачу.

73 с., 27 рис., 0 дод., 37 джерел.

Ключові слова: веб-додаток, електронний реєстр, пацієнт, JavaScript, React.

ABSTRACT

Butlerskyi D.S. Web application for managing the electronic patient registry. Specialization 122 "Computer Science", educational program "Data Science", Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

The qualification (master's) thesis addresses the problem of managing an electronic patient registry. Existing web applications for solving this problem were analyzed, and a web application was developed based on these findings to address the specified task.

73 p., 27 figures., 0 app., 37 ref.

Keywords: web-application, electronic registry, patient, JavaScript, React.

ЗМІСТ

ВСТУП

Інформаційні технології стали невід'ємною частиною сучасного суспільства, сприяючи автоматизації та вдосконаленню процесів у різних сферах діяльності, зокрема в охороні здоров’я. Одним із ключових завдань медичної галузі є забезпечення ефективного управління даними пацієнтів, що дозволяє оптимізувати обслуговування, покращити якість наданих послуг і знизити адміністративні витрати. У цьому контексті веб-додатки для ведення електронних реєстрів пацієнтів є важливим кроком до модернізації медичних установ. Дана робота присвячена розробці такого додатку, що відповідає сучасним вимогам безпеки, функціональності та зручності використання.

Актуальність теми дослідження зумовлена тим, що в умовах активного впровадження цифрових технологій у різні сфери життя, зокрема в охорону здоров’я, питання ефективної організації медичних даних є надзвичайно актуальним. Традиційні методи ведення документації часто є неефективними, потребують значних ресурсів та часу, а також можуть призводити до помилок через людський фактор. Розробка веб-додатку для ведення електронного реєстру пацієнтів дозволяє автоматизувати процеси обліку, зменшити ризик втрати даних, покращити комунікацію між медичними працівниками та забезпечити конфіденційність даних пацієнтів.

Об'єктом дослідження є процес цифровізації ведення реєстру пацієнтів у медичних установах.

Предмет дослідження – веб-додатки для ведення електронного реєстру пацієнтів.

Мета дослідження полягає в тому, щоб розробити веб-додаток для ведення електронного реєстру пацієнтів, який забезпечує зручний доступ, обробку та збереження даних пацієнтів відповідно до сучасних стандартів інформаційної безпеки.

Завдання дослідження:

1) Проаналізувати сучасний стан автоматизації ведення медичної документації.
2) Дослідити існуючі технології та інструменти для розробки веб-додатків.
3) Розробити структуру та функціонал веб-додатку.
4) Реалізувати веб-додаток з урахуванням вимог безпеки та зручності користування.

Наукова актуальність роботи полягає у розробці сучасного інструменту для оптимізації процесів у медичній сфері що забезпечує ефективне та безпечне ведення електронного реєстру пацієнтів.

Практична цінність – результати роботи можуть бути використані медичними закладами для оптимізації обробки даних пацієнтів, зменшення паперового документообігу та підвищення ефективності медичного обслуговування.

Достовірність результатів підтверджена шляхом тестування розробленого додатку на прикладі умовної медичної установи.

За результатами роботи опубліковано тези доповіді [30].

РОЗДІЛ 1
АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз цифровізації інструментів для ведення реєстру пацієнтів

Цифровізація[1] у сфері охорони здоров'я стала ключовим елементом трансформації медичних послуг та оптимізації управління пацієнтськими записами. Перехід від паперових до електронних медичних записів (ЕМЗ) сприяв значному покращенню ефективності, безпеки та якості медичних послуг. Цей розділ детально розглядає основні аспекти цифровізації інструментів для управління пацієнтськими записами, а також їх переваги, виклики та перспективи.

Цифровізація медичних записів має численні переваги[2] для медичних закладів, пацієнтів та медичних працівників. По-перше, покращена доступність та обмін даними значно підвищують ефективність медичного обслуговування. Електронні медичні записи забезпечують легкий доступ до пацієнтських даних у режимі реального часу, що особливо важливо для координації лікування між різними медичними фахівцями. Це зменшує ризик помилок та дублювання діагностичних процедур, забезпечуючи безперервність та узгодженість лікування.

По-друге, підвищена точність та зменшення помилок є однією з найбільших переваг цифровізації. Цифрові системи знижують ризик людських помилок, пов'язаних із ручним введенням даних. Автоматизовані системи нагадувань та попереджень допомагають медичним працівникам уникнути помилок у призначенні лікування або дозування ліків, що сприяє підвищенню безпеки пацієнтів.

Третя перевага стосується ефективного управління та адміністрування. Системи ЕМЗ спрощують процеси адміністрування, такі як виставлення рахунків, планування прийомів та управління ресурсами. Це дозволяє медичним закладам оптимізувати свою роботу, знижуючи витрати та покращуючи якість послуг.

Четвертою важливою перевагою є підвищення якості медичних послуг. Завдяки швидкому доступу до повних та актуальних медичних даних лікарі можуть приймати більш обґрунтовані рішення, що підвищує якість лікування та результативність медичних втручань. Крім того, цифрові інструменти дозволяють більш точно стежити за станом пацієнтів і оперативно вносити корективи в лікування.

П'ятою перевагою є покращення безпеки даних. Електронні системи забезпечують вищий рівень безпеки даних через використання шифрування та контролю доступу. Це допомагає захистити конфіденційну інформацію пацієнтів від несанкціонованого доступу та кібератак, що є критично важливим у сучасному світі, де загрози кібербезпеки постійно зростають.

Незважаючи на численні переваги, цифровізація медичних записів стикається з певними викликами, які необхідно враховувати. Одним з основних викликів є необхідність значних інвестицій у впровадження та навчання. Впровадження систем ЕМЗ вимагає значних фінансових ресурсів для придбання обладнання, програмного забезпечення та навчання персоналу. Це може бути складно для деяких медичних закладів, особливо для малих клінік з обмеженими бюджетами.

Другим викликом є питання конфіденційності та безпеки. Захист конфіденційної інформації пацієнтів є критично важливим завданням для медичних закладів. Вони повинні забезпечити відповідні заходи безпеки для захисту даних від кібератак та інших загроз. Це включає використання сучасних технологій шифрування, багатофакторної автентифікації та регулярних аудитів безпеки.

Третій виклик пов'язаний із сумісністю та інтеграцією систем. Часто медичні заклади використовують різні системи ЕМЗ, що може ускладнювати обмін даними між різними установами. Важливо забезпечити сумісність та інтеграцію цих систем для забезпечення безперешкодного обміну інформацією та покращення координації лікування.

Четвертий виклик стосується сприйняття та прийняття технологій. Медичні працівники можуть зустрічати труднощі у переході на нові технології, особливо якщо вони звикли до паперових записів. Необхідно забезпечити відповідне навчання та підтримку для полегшення цього переходу та підвищення рівня прийняття нових систем.

Розглянемо більш детально перспективи цифровізації в медичній сфері. Цифрові технології мають великий потенціал у поліпшенні доступності медичних послуг. Наприклад, телемедицина дозволяє пацієнтам отримувати консультації від лікарів без необхідності фізичного відвідування медичного закладу. Це особливо корисно для тих, хто мешкає в віддалених районах або має обмежений доступ до медичних установ. Також, телемедицина може значно полегшити співпрацю між лікарями та забезпечити швидкий обмін медичною інформацією між медичними закладами.

Ще одним важливим напрямком є використання аналітики даних та штучного інтелекту в медицині. За допомогою цих технологій можна аналізувати великі обсяги медичних даних, виявляти складні зв'язки між різними параметрами здоров'я та розробляти індивідуалізовані підходи до діагностики та лікування. Наприклад, алгоритми машинного навчання можуть допомогти виявити патерни, які вказують на ризик розвитку певних захворювань, що дозволяє розпочати профілактичні заходи вчасно.

Одним із важливих аспектів цифровізації є розвиток мобільних додатків для здоров'я. Ці додатки дозволяють користувачам вести облік свого здоров'я, контролювати фізичну активність, харчування та прийом

ліків. Вони стають справжніми помічниками у підтримці здорового способу життя та веденні здорового образу життя.

Крім того, віртуальна та розширена реальності відкривають нові можливості у медицині. Ці технології використовуються для навчання медичних працівників, симуляції складних процедур та навіть для психотерапії та зниження болю у пацієнтів. Вони допомагають покращити навчальний процес та забезпечити більш реалістичну та ефективну медичну практику.

Загалом, цифрові технології в медицині відкривають безліч перспектив для поліпшення якості та доступності медичних послуг. Однак важливо забезпечити безпеку та конфіденційність медичних даних, а також враховувати етичні та правові аспекти використання цих технологій.

1.2 Огляд аналогів

Телемедицина та віртуальні платформи здоров'я стають все більш популярними завдяки своїй зручності та доступності. У цьому розділі я детально розглянув кілька провідних платформ для телемедицини та медичних консультацій в онлайн-режимі.

Кожна з цих платформ пропонує свій унікальний підхід до надання медичних послуг в дистанційному режимі, маючи свої переваги та особливості. Я розгляну їхні можливості, переваги та недоліки, а також визначимо ключові функції, що вирізняють кожну з них.

1. Epic Systems Corporation[3]

Epic Systems Corporation - це компанія, що спеціалізується на розробці програмного забезпечення для медичних установ. Їхнє головне рішення - Epic Electronic Health Record (EHR) - це високофункціональна система, яка дозволяє зберігати та обмінюватися медичною інформацією про пацієнтів. Epic EHR включає в себе широкий спектр модулів, від зберігання медичних записів до керування лікарськими призначеннями та планування прийому пацієнтів.

Переваги:

- Epic EHR має потужний функціонал, що включає в себе всі аспекти медичної практики, що робить його привабливим для великих медичних установ та мереж клінік.
- Система гарантує високий рівень безпеки та конфіденційності медичної інформації.

Мінуси:

- Впровадження системи Epic EHR може бути дорогим та часомістким процесом через необхідність налагодження та навчання персоналу.
- Інтерфейс системи може здаватися складним для деяких користувачів.

Основні функції:

- Електронний медичний реєстр пацієнтів з можливістю доступу до історії лікування, результатів аналізів, лікарських призначень та іншої медичної інформації.
- Модуль для планування та прийому пацієнтів з можливістю онлайн запису на прийом.
- Функціонал для аналізу медичних даних та генерації звітів для адміністративних цілей.

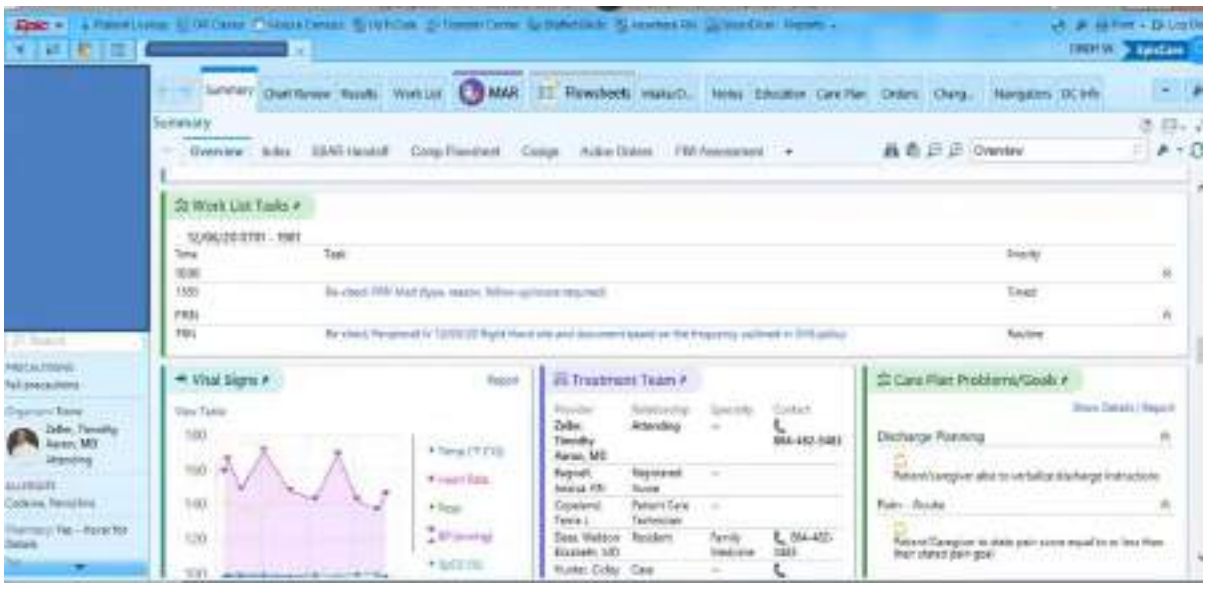

Рис. 1 – Головний екран Epic Electronic Health Record

2. Cerner Corporation[4]

Cerner Corporation є глобальним лідером у сфері медичного програмного забезпечення, який надає інтегровані рішення для ефективного управління медичними закладами. Одним з їхніх ключових продуктів є Cerner Millennium, що включає рішення для електронного реєстру пацієнтів, керування лікарськими призначеннями, а також аналітичні інструменти.

Переваги:

- Cerner Millennium забезпечує інтегрований підхід до управління медичними закладами, що спрощує роботу медичного персоналу та підвищує ефективність надання медичних послуг.
- Система гнучка та масштабована, що дозволяє використовувати її як у невеликих медичних кабінетах, так і у великих лікарнях чи мережах клінік.

Мінуси:

- Впровадження Cerner Millennium може вимагати значних витрат на налаштування та інтеграцію з існуючими системами.
- Інтерфейс системи може здатися складним для деяких користувачів, особливо для тих, хто не має достатнього досвіду роботи з комп'ютерами.

Основний функціонал:

- Електронний медичний реєстр пацієнтів з можливістю зберігання та доступу до історії лікування, результатів аналізів, лікарських призначень та іншої медичної інформації.
- Модуль керування лікарськими призначеннями, що дозволяє лікарям зручно оформляти рецепти та замовляти ліки для пацієнтів.

- Інтегровані аналітичні інструменти для аналізу медичних даних та генерації звітів для управлінських рішень.

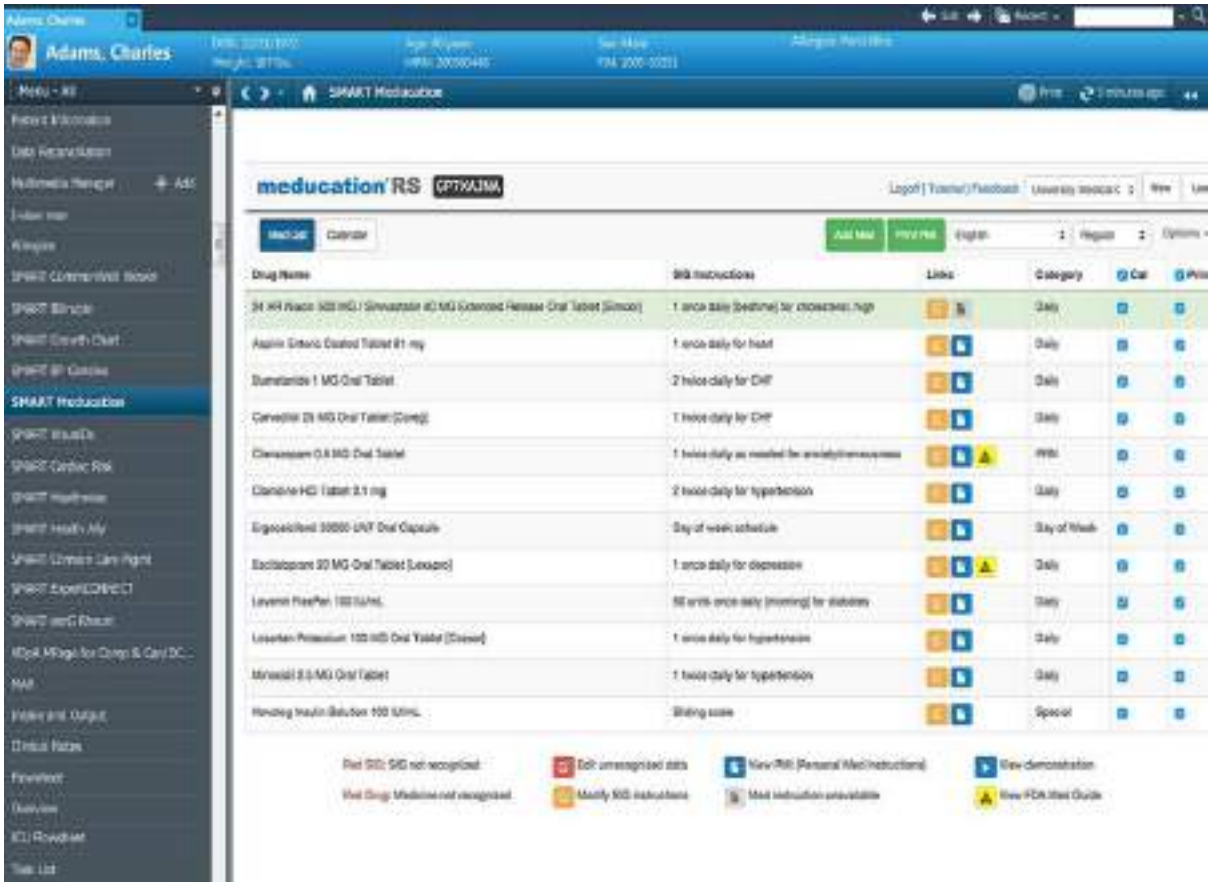

Рис. 2 – Головний екран Cerner Corporation

3. Allscripts Healthcare Solutions[5]

Allscripts Healthcare Solutions є провідним постачальником програмного забезпечення для медичних практик різних розмірів. Їхній продукт включає рішення для електронного реєстру пацієнтів, керування лікарськими призначеннями, планування прийому пацієнтів та аналізу даних.

Переваги:

- Allscripts пропонує гнучку та легко налаштовувану систему, яка може відповісти на потреби медичних закладів різних розмірів та спеціалізацій.
- Інтеграція з іншими системами медичних установ дозволяє забезпечити безперервний обмін даними та забезпечити єдиний інформаційний простір.

Мінуси:

- Потенційна складність інтеграції з існуючими системами та процесами медичних закладів може стати перешкодою під час впровадження.
- Недостатній рівень кастомізації для деяких особливих потреб медичних установ.

Основний функціонал:

- Електронний медичний реєстр пацієнтів, який забезпечує зручний доступ до медичної інформації та історії лікування.
- Модуль керування лікарськими призначеннями, що дозволяє швидко та безпечно оформляти рецепти та виписувати ліки.
- Інструменти для планування та керування прийомами пацієнтів, що дозволяють оптимізувати робочі процеси та збільшити ефективність надання послуг.

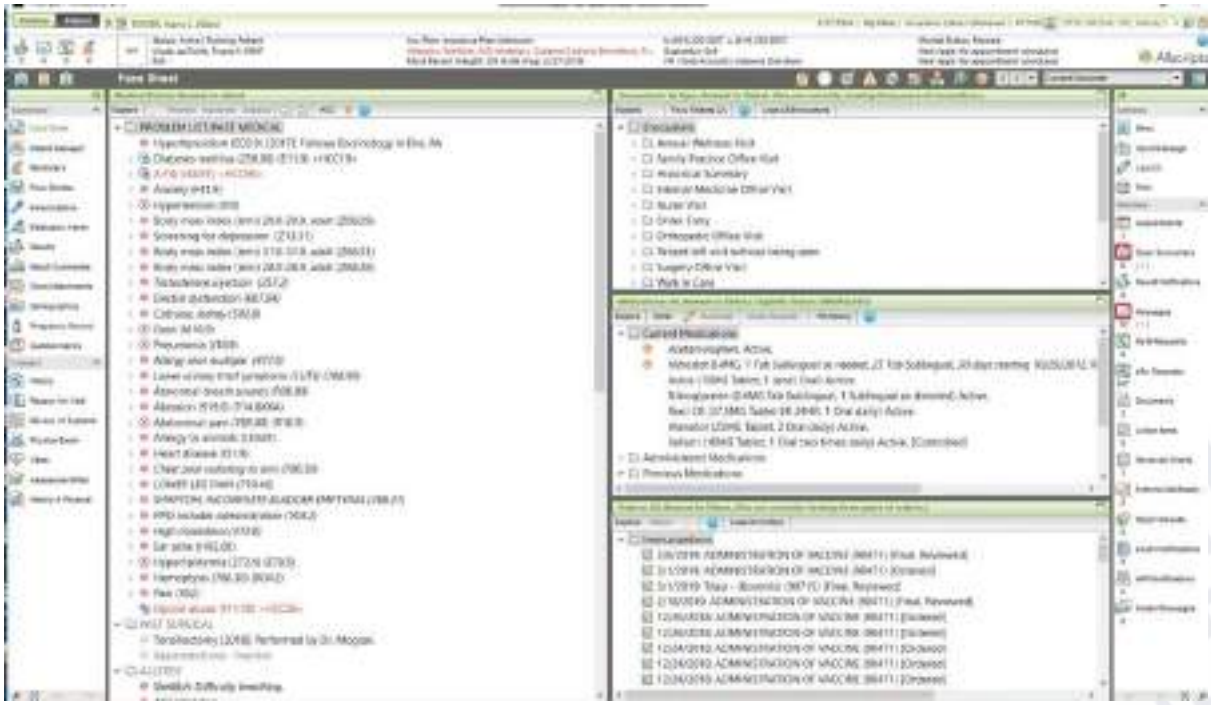

Рис. 3 – Головний екран Allscripts Healthcare Solutions

4. NextGen Healthcare Information Systems[6]

NextGen Healthcare Information Systems - це компанія, яка спеціалізується на розробці програмного забезпечення для медичних практик різних масштабів. Їхній продукт включає в себе рішення для електронного реєстру пацієнтів, керування лікарськими призначеннями, аналізу даних та інші інструменти, призначені для автоматизації та оптимізації медичної практики.

Переваги:

- NextGen пропонує гнучке та налагоджене програмне забезпечення, яке може відповісти на потреби будь-якого типу медичного закладу.
- Інтеграція з іншими системами медичних установ дозволяє легко обмінюватися даними та забезпечити єдиний інформаційний простір для усіх сторін.

Мінуси:

- Настройка системи та інтеграція з існуючими процесами медичних закладів можуть вимагати значних зусиль та ресурсів.
- Для деяких користувачів інтерфейс програми може здатися складним або перевантаженим.

Основний функціонал:

- Електронний медичний реєстр пацієнтів з можливістю зберігання та швидкого доступу до всієї необхідної інформації.
- Модуль керування лікарськими призначеннями, який дозволяє лікарям ефективно оформляти рецепти та замовляти ліки для пацієнтів.
- Аналітичні інструменти для обробки та аналізу медичних даних, що допомагають приймати обґрунтовані управлінські рішення.

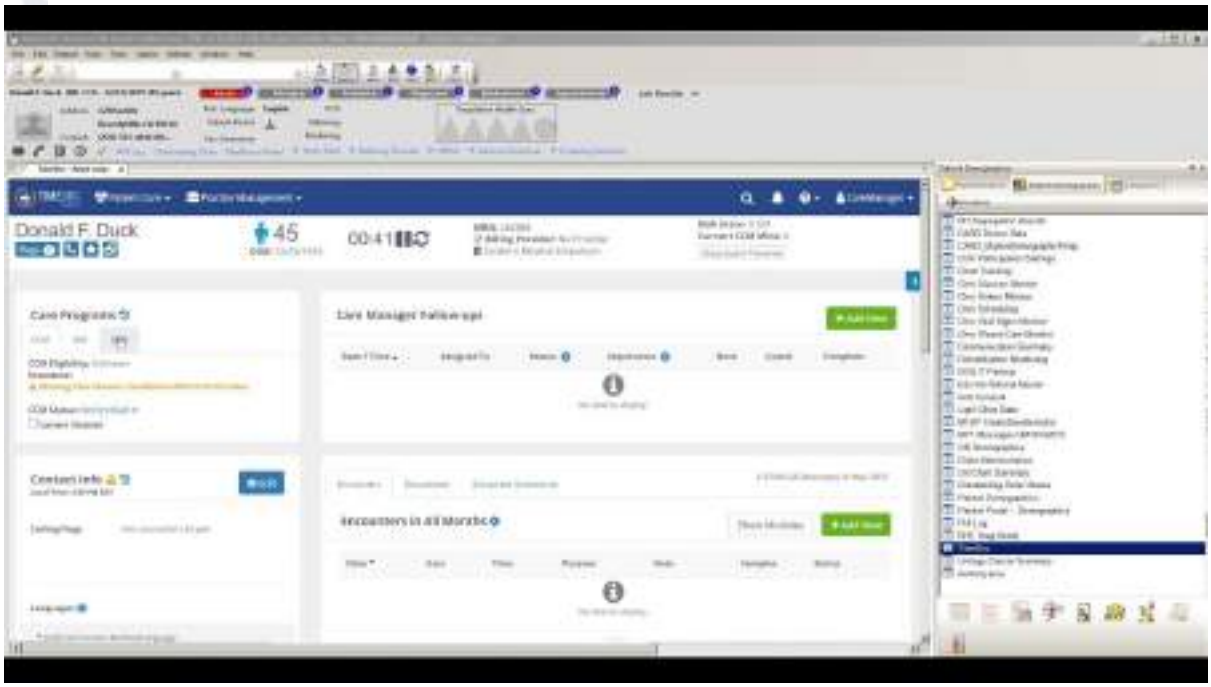

Рис. 4 – Головний екран NextGen Healthcare Information Systems

5. Doxy.me[7]

Doxy.me - це платформа для телемедицини, що дозволяє лікарям та пацієнтам проводити консультації в реальному часі через веб-браузер або мобільний додаток. Платформа пропонує зручний і безкоштовний спосіб зв'язку з медичними працівниками без фізичного відвідування медичних установ. Веб-додаток є простим, безкоштовним і безпечним рішення для телемедицини. Розроблений з думкою про охорону здоров'я, doxy.me - це телекомунікаційна платформа, сумісна з HIPAA, якою легко користуватися як пацієнтам, так і лікарям. Вона дозволяє організаціям охорони здоров'я надавати віртуальну допомогу, яка легко доступна кожному, скрізь і з будь-якого пристрою.

Doxy.me - одна з небагатьох телемедичних платформ, яка пропонує безкоштовну версію, що відповідає вимогам HIPAA. Це означає, що будь-хто може створити обліковий запис і випробувати платформу абсолютно без ризику. Продукт локалізовано 100 мовами, а також пропонує цілодобовий усний переклад більш ніж 240 мовами, що робить його доступним для використання в усьому світі.

Переваги:

- Простий та інтуїтивно зрозумілий інтерфейс для лікарів та пацієнтів.
- Захищений з'єднання та висока рівень конфіденційності медичних даних.
- Можливість вести консультації з будь-якого пристрою з доступом до Інтернету.
- Широкий спектр функцій, включаючи відеоконференції, текстовий чат та обмін файлами.

Мінуси:

- У безкоштовному тарифному плані обмежена кількість одночасних консультацій та обмін файлами.
- Деякі користувачі відзначають можливі проблеми з якістю зв'язку та відставанням звуку під час відеоконференцій.

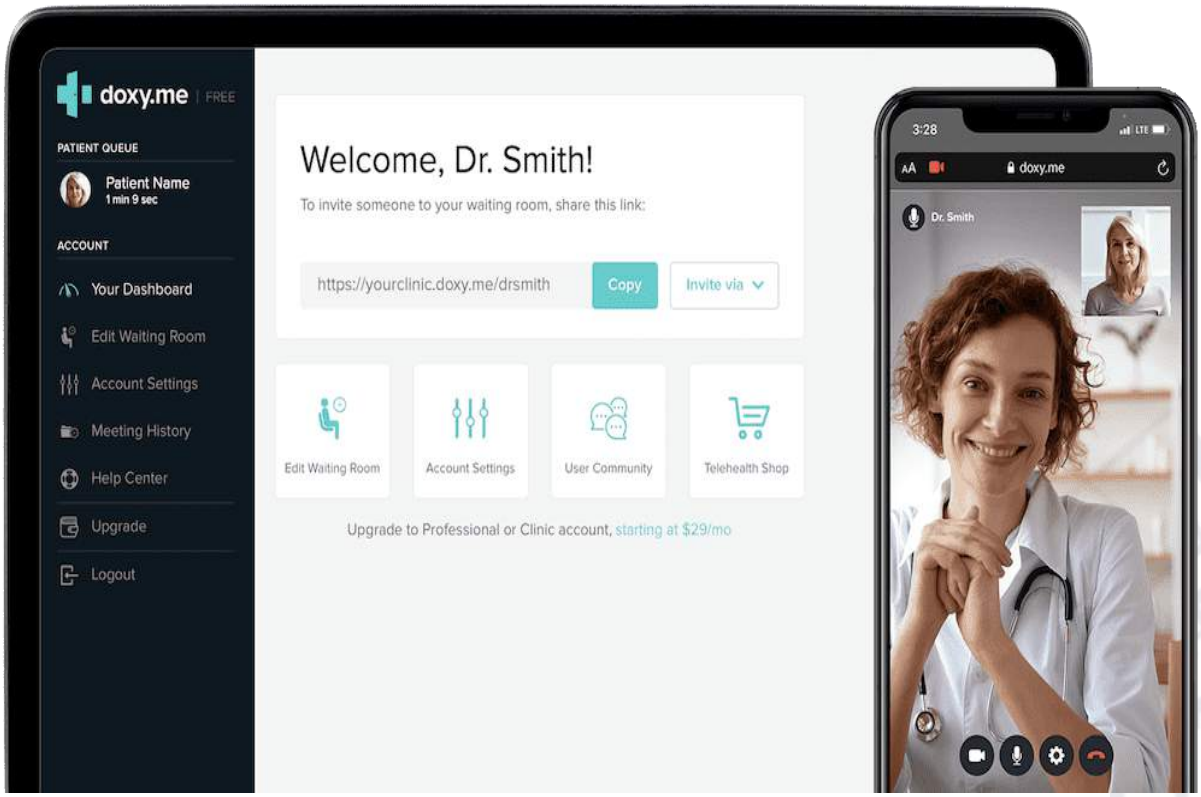

Рис. 5 – Головний екран Doxy.me

6. Mindly[8]

Mindly - це платформа, яка допомагає користувачам створювати і керувати своїми думками, ідеями та проектами шляхом створення ментальних карт. Основна ідея Mindly полягає в тому, щоб візуалізувати ідеї та організувати їх у логічні структури, що допомагає в подальшому аналізі та розвитку цих ідей.

Mindly надає користувачам інтуїтивно зрозумілий інтерфейс для створення та редагування ментальних карт. Користувачі можуть додавати вузли, з'єднуючі лінії, різні кольори та іконки, щоб краще візуалізувати свої думки. Платформа дозволяє створювати складні структури з багатьох рівнів, що дозволяє організовувати ідеї у логічні та структуровані способи.

Однією з особливостей Mindly є його універсальність - він може бути використаний для будь-якого типу проектів або завдань, від особистих цілей до бізнесу. Користувачі можуть використовувати Mindly для

створення планів, організації проектів, управління завданнями, роботи над ідеями та багато іншого.

Mindly також надає можливості спільної роботи, що дозволяє користувачам спільно працювати над ментальними картами та обмінюватися ними з іншими користувачами.

Узагальнюючи, Mindly - це потужний інструмент для візуалізації та організації думок і ідей, який допомагає користувачам краще розуміти, аналізувати та розвивати свої проекти та ідеї.

Переваги:

- Інтуїтивний інтерфейс: Mindly пропонує простий та легкий у використанні інтерфейс, що дозволяє швидко створювати та редагувати ментальні карти без особливих навичок.
- Візуалізація думок: Платформа дозволяє візуалізувати складні ідеї та концепції у вигляді структурованих ментальних карт, що полегшує їхнє розуміння та аналіз.
- Універсальність в застосуванні: Mindly може бути використаний для будь-якого типу проектів, завдань або думок, незалежно від їхньої складності чи специфіки.
- Спільна робота: Платформа надає можливість спільної роботи над ментальними картами, що дозволяє користувачам спільно працювати над проектами та обмінюватися ідеями з колегами.

Недоліки:

- Обмежена функціональність: У порівнянні з деякими іншими програмами для створення ментальних
- карт, Mindly може мати обмежений набір функцій та можливостей.
- Відсутність розширених інструментів аналізу: Деякі користувачі можуть відчувати нестачу розширених інструментів для аналізу даних чи візуалізації.

Основний функціонал:

- Створення ментальних карт: Користувачі можуть легко створювати ментальні карти з різноманітних елементів та зв'язків.
- Додавання вузлів та зв'язків: Можливість додавати нові вузли та зв'язки між ними для організації думок та ідей.
- Візуалізація даних: Ментальні карти можуть містити різні кольори, іконки та стилізацію для кращої візуалізації думок.
- Експорт та імпорт даних: Можливість експорту та імпорту ментальних карт у різних форматах для обміну даними з іншими користувачами чи програмами.

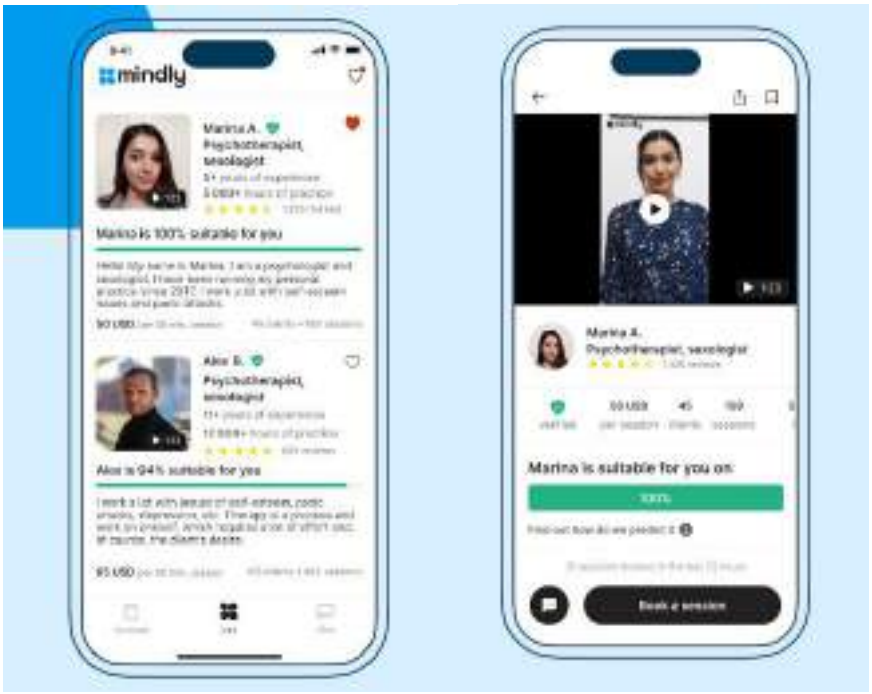

Рис. 6 – Головний екран Mindly

1.3 Правові та етичні аспекти ведення електронних реєстрів пацієнтів

Правові та етичні аспекти[9] відіграють важливу роль у процесі цифровізації медичних реєстрів. Забезпечення конфіденційності, безпеки та доступності медичних даних потребує дотримання суворих правових норм та етичних стандартів. В умовах швидкого розвитку технологій та збільшення обсягу оброблюваної інформації виникає необхідність у

врегулюванні питань, пов'язаних із захистом персональних даних пацієнтів, отриманням згоди на їх обробку, а також забезпеченням етичного підходу до використання та обміну цією інформацією.

Правові вимоги до обробки персональних даних пацієнтів є ключовим аспектом цифровізації медичних реєстрів. В різних країнах діють різні закони та регламенти, які регулюють обробку медичних даних. Наприклад, в Європейському Союзі діє Загальний регламент захисту даних (GDPR), який встановлює суворі вимоги до збору, зберігання та обробки персональних даних. GDPR передбачає, що організації повинні отримувати чітку та однозначну згоду від пацієнтів на обробку їхніх даних, а також забезпечувати можливість для пацієнтів відкликати цю згоду в будь-який момент. Крім того, закон зобов'язує організації повідомляти пацієнтів про те, які дані збираються, з якою метою і хто матиме до них доступ. У випадку порушення вимог GDPR передбачаються суворі штрафи та інші санкції, що стимулює організації дотримуватися високих стандартів захисту даних.

В Україні правове регулювання обробки медичних даних здійснюється на основі Закону України "Про захист персональних даних", який набрав чинності 1 січня 2011 року. Цей закон визначає правові засади захисту персональних даних при їх обробці, зокрема, в медичних інформаційних системах. Закон встановлює, що обробка персональних даних може здійснюватися лише за згодою суб'єкта персональних даних або на інших законних підставах.

Закон України "Про захист персональних даних" передбачає, що медичні заклади зобов'язані забезпечити належний захист персональних даних пацієнтів від незаконної обробки, включаючи втрату, знищення, пошкодження, несанкціонований доступ та розповсюдження. Для цього медичні установи повинні впроваджувати технічні та організаційні заходи, що забезпечують безпеку даних, такі як шифрування, контроль доступу, резервне копіювання та аудит.

Закон також визначає права суб'єктів персональних даних, включаючи право на доступ до своїх даних, право на їх виправлення, видалення та блокування, а також право на заперечення проти обробки даних. Пацієнти повинні бути проінформовані про свої права та мати можливість реалізувати їх без будь-яких ускладнень.

Крім закону "Про захист персональних даних", в Україні діють інші нормативні акти, що регулюють обробку медичних даних. Наприклад, Закон України "Про інформацію" та Закон України "Про електронні документи та електронний документообіг" визначають загальні засади обробки інформації та використання електронних документів, включаючи медичну інформацію. Закон України "Про основи законодавства України про охорону здоров'я" також містить положення щодо конфіденційності медичних даних та забезпечення прав пацієнтів на захист їхньої інформації.

Отримання інформованої згоди пацієнтів на обробку їхніх даних є одним з основних принципів правового регулювання цифрових медичних реєстрів. Пацієнти повинні бути повністю поінформовані про те, які дані будуть зібрані, з якою метою вони будуть використовуватися, як довго вони будуть зберігатися та хто матиме до них доступ. Це забезпечує прозорість обробки даних та захист прав пацієнтів на приватність.

Процес отримання згоди має бути зрозумілим та прозорим. Пацієнтам повинні бути надані чіткі та доступні пояснення щодо обробки їхніх даних, а також можливість задати питання та отримати на них відповіді. Важливо, щоб згода була отримана до початку обробки даних і могла бути відкликана у будь-який момент без негативних наслідків для пацієнта.

Конфіденційність медичних даних є одним з основних етичних питань у сфері охорони здоров'я. Лікарі та медичний персонал зобов'язані забезпечувати конфіденційність інформації про пацієнтів, дотримуючись принципу лікарської таємниці. Використання електронних реєстрів

повинно забезпечувати такий самий рівень захисту конфіденційності, як і традиційні паперові медичні записи.

Етичні стандарти вимагають, щоб медична інформація пацієнтів використовувалася виключно в межах, необхідних для надання медичної допомоги. Необхідно мінімізувати доступ до даних та забезпечувати, щоб тільки уповноважені особи мали можливість доступу до медичної інформації. Всі операції з даними повинні бути ретельно документовані, а доступ до них контролюватися та перевірятися.

Етичні питання доступу до медичних даних включають визначення, хто має право на доступ до цієї інформації, та в яких умовах. Пацієнти повинні мати можливість отримувати доступ до своїх медичних даних, перевіряти їхню точність та вносити необхідні корективи. Це забезпечує пацієнтам контроль над своєю медичною інформацією та дозволяє уникнути помилок у медичній документації.

Крім того, доступ до медичних даних для наукових досліджень є важливим аспектом. Використання медичних даних у дослідженнях може сприяти розробці нових методів лікування та покращенню якості медичних послуг. Однак, необхідно забезпечити, щоб такі дослідження проводилися з дотриманням етичних норм та отриманням згоди пацієнтів на використання їхніх даних.

Забезпечення безпеки даних пацієнтів є не лише правовою вимогою, але й етичним обов'язком. Медичні заклади повинні впроваджувати надійні методи захисту даних від несанкціонованого доступу, втрати або викрадення. Використання сучасних криптографічних методів для шифрування даних під час їх зберігання та передачі є критично важливим для забезпечення безпеки.

Багатофакторна аутентифікація, яка вимагає від користувачів підтвердження своєї особи за допомогою декількох факторів, таких як пароль, смс-код або біометричні дані, значно підвищує рівень захисту.

Контроль доступу до даних повинен бути суворим, із визначенням чітких правил та процедур доступу до медичної інформації.

Впровадження систем моніторингу та аудиту дозволяє вчасно виявляти та реагувати на потенційні загрози безпеці. Це включає регулярний перегляд логів доступу, проведення аудиторських перевірок та оцінку ризиків. Етичний підхід до безпеки даних передбачає постійну роботу над удосконаленням методів захисту та врахування нових загроз і викликів.

Порушення конфіденційності та безпеки медичних даних можуть мати серйозні правові наслідки для організацій, що обробляють ці дані. У випадку виявлення порушень, такі організації можуть бути притягнуті до відповідальності та зобов'язані сплатити значні штрафи. Відповідно до українського законодавства, за порушення вимог щодо захисту персональних даних передбачено адміністративну та кримінальну відповідальність, включаючи штрафи, виправні роботи та обмеження волі.

Крім штрафів, організації можуть зазнати репутаційних втрат, що може мати негативний вплив на їхню діяльність та відносини з пацієнтами. Важливо, щоб медичні заклади розуміли наслідки можливих порушень і вживали всіх необхідних заходів для їх запобігання.

Використання медичних даних для наукових досліджень та покращення медичних послуг є важливим, але воно має відповідати етичним стандартам. Необхідно забезпечити, щоб дані використовувалися з дотриманням принципів етики, зокрема принципу поваги до особистості, принципу справедливості та принципу не нанесення шкоди.

Принцип поваги до особистості передбачає, що пацієнти повинні бути інформовані про те, як їхні дані будуть використовуватися, та давати згоду на таке використання. Принцип справедливості вимагає, щоб результати досліджень приносили користь усім групам пацієнтів, а не лише окремим категоріям. Принцип ненанесення шкоди означає, що дослідження

повинні проводитися таким чином, щоб мінімізувати будь-які ризики для пацієнтів та їхніх даних.

Забезпечення відповідності етичним нормам у використанні медичних даних є ключовим для довіри пацієнтів до системи охорони здоров'я та для підтримки високих стандартів якості медичних послуг.

Висновок до розділу 1

У першому розділі було досліджено цифровізацію інструментів для ведення реєстру пацієнтів, а також проведено огляд аналогів та розглянуто правові й етичні аспекти. Було проаналізовано переваги переходу до електронних медичних записів, включаючи підвищення доступності, точності, ефективності та безпеки даних, а також виклики, пов’язані із впровадженням, конфіденційністю та сумісністю систем. Було оглянуто провідні програмні рішення в цій сфері, їхній функціонал, переваги та недоліки. Також розглянуто правові та етичні норми, яким має відповідати використання електронних реєстрів, і визначено необхідність забезпечення безпеки та прозорості обробки даних для підвищення довіри пацієнтів і підтримки високих стандартів медичних послуг.

РОЗДІЛ 2
АНАЛІЗ ТА ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ

2.1 Вибір архітектури веб-додатка

Для побудови веб-додатка для ведення електронного реєстру пацієнтів було обрано архітектуру типу клієнт-сервер з використанням підходу REST API[10] для взаємодії між клієнтом (фронтендом) та сервером (бекендом). Така архітектура забезпечує чіткий поділ між логікою інтерфейсу користувача та серверною логікою, що сприяє масштабованості, гнучкості та зручності у підтримці системи.

Архітектура клієнт-сервер розділяє додаток на дві основні частини:

1. Клієнтська частина (фронтенд)[11] відповідає за взаємодію з користувачем і відображення інформації. Для реалізації клієнтської

частини було обрано технологію React.js[12], яка дозволяє створювати динамічні інтерфейси користувача за допомогою компонентів. Фронтенд виконує запити до бекенду через API і отримує дані, які відображаються в інтерфейсі.

2. Серверна частина (бекенд)[13] відповідає за обробку запитів клієнта, роботу з базою даних та бізнес-логіку. Для бекенд-частини було обрано Node.js[14], що дозволяє швидко обробляти запити завдяки асинхронній природі JavaScript[15]. Сервер також виступає посередником між клієнтом і базою даних, забезпечуючи захист і обробку даних пацієнтів.

Архітектура клієнт-сервер[16] має кілька альтернативних архітектурних підходів, які використовуються для розробки веб-додатків, таких як:

- Монолітна архітектура
- Архітектура мікросервісів

Монолітна архітектура[17]

Монолітна архітектура вирізняється простотою розробки та налаштування. Всі компоненти додатка знаходяться в одному проекті, що дозволяє швидко і без ускладнень створювати програму. Такий підхід усуває потребу в складній інфраструктурі для комунікації між окремими сервісами, що робить моноліт зручним для невеликих команд або стартапів, які прагнуть швидко розробити MVP (Minimum Viable Product)[18]. Завдяки тому, що всі компоненти інтегровані в єдину систему, процес тестування також значно спрощується: немає необхідності тестувати мережеві зв'язки між модулями або сервісами, що дозволяє проводити наскрізне тестування на рівні всього додатка. Розгортання монолітних систем теж менш

проблемне, адже весь код розгортається як єдиний блок. Це спрощує управління версіями і знижує складність процесів розгортання, оскільки потрібно лише одна точка для розгортання всього додатка. Також, монолітна архітектура має нижчі витрати на підтримку інфраструктури, оскільки немає потреби в оркестрації мікросервісів або окремих серверів, що може знижувати витрати на хостинг і зменшувати управлінське навантаження.

Однак, монолітна архітектура має свої недоліки. Основна проблема полягає у складнощах із внесенням змін і оновленням. Оскільки всі частини програми тісно пов'язані між собою, будь-яка зміна в одному компоненті може вимагати перегляду або навіть перетестування всієї системи, що значно уповільнює процес оновлення і впровадження нових функцій. У великих монолітних додатках код може стати занадто складним для підтримки, що призводить до накопичення технічного боргу та ускладнює подальші оновлення. Ще однією проблемою є масштабованість: коли один модуль потребує більше ресурсів, доводиться масштабувати всю систему, навіть якщо інші частини в цьому не мають потреби. Це може призвести до неефективного використання ресурсів і ускладнити адаптацію додатка до зростаючих вимог. Окрім цього, монолітні системи вразливі до помилок: збій в одному компоненті може вплинути на роботу всього додатка. Наприклад, витік пам'яті в одному місці може призвести до падіння всієї програми. Цикл розгортання великих монолітних додатків зазвичай досить повільний, оскільки будь-яка зміна вимагає повного розгортання системи, що стає особливо проблематичним у великих програмах з багатьма залежностями.

Монолітна архітектура може бути ефективним рішенням для невеликих проектів і команд, які прагнуть швидкого запуску продукту. Проте у міру зростання додатка та команди ця архітектура може призвести

до складнощів з масштабованістю та підтримкою, що робить її менш ефективною для великих або складних систем.

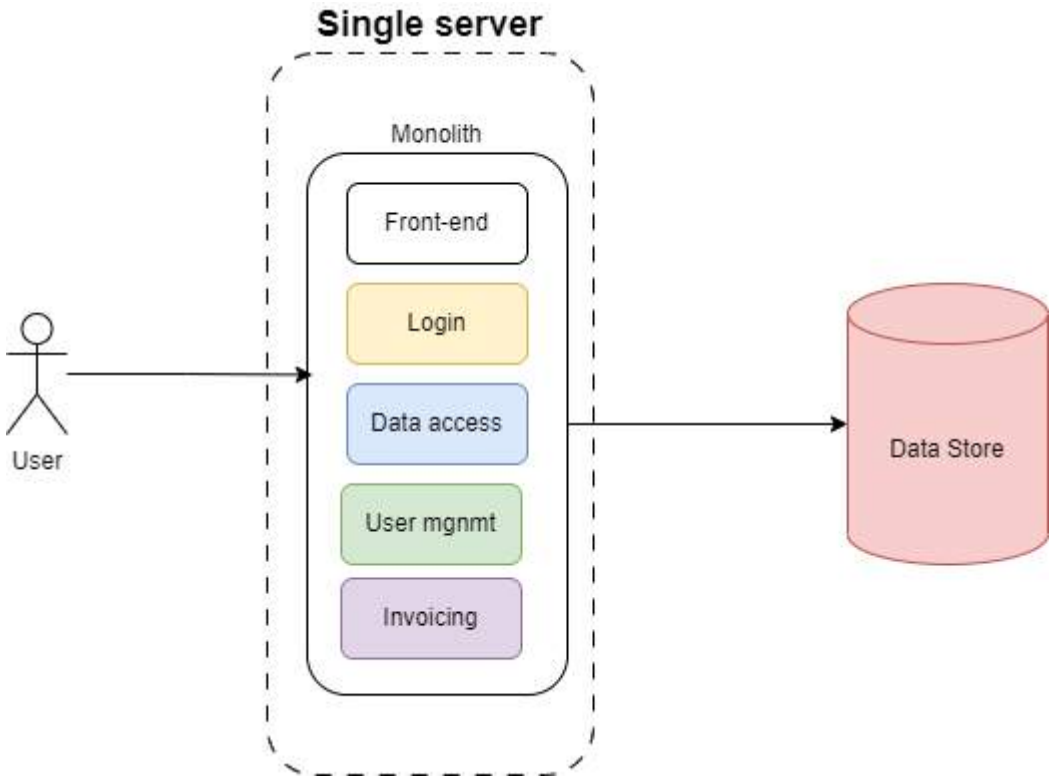

Рис. 7 – Схема монолітної архітектури

Архітектура мікросервісів[19]

Архітектура мікросервісів є сучасним підходом до створення програмного забезпечення, при якому додаток складається з невеликих, незалежних сервісів. Кожен мікросервіс відповідає за одну конкретну функцію або бізнес-логіку додатка, і вони взаємодіють між собою через стандартизовані інтерфейси, зазвичай за допомогою API (часто REST або gRPC[20]). Це дає можливість окремо розробляти, тестувати, розгортати і масштабувати кожен сервіс незалежно від інших.

Цей підхід полягає в розділенні додатка на незалежні сервіси, кожен з яких виконує певну задачу. Всі ці сервіси можуть бути розгорнуті на

окремих серверах або контейнерах, маючи власні бази даних, що дозволяє кожному з них працювати ізольовано. Замість централізованого управління сервісами (як в моноліті), мікросервіси взаємодіють один з одним через мережеві запити, такі як HTTP[21] або протоколи передачі повідомлень (наприклад, через брокери повідомлень, як RabbitMQ або Kafka).

Мікросервісна архітектура надає значну модульність і незалежність. Кожен мікросервіс діє як окремий модуль, що полегшує процеси розробки та підтримки. Це дозволяє окремим командам працювати паралельно над різними частинами системи, прискорюючи загальну розробку. Використання різних технологій і мов програмування для кожного сервісу додає гнучкості в виборі інструментів для конкретних задач. Однією з найбільших переваг мікросервісів є можливість масштабування кожного сервісу окремо. Якщо певний компонент системи стикається з підвищеним навантаженням, наприклад, при роботі з базою даних або авторизацією, його можна масштабувати без необхідності змінювати інші частини програми. Це дозволяє ефективніше використовувати ресурси й економити на витратах.

Ще одна важлива перевага мікросервісів полягає в їх стійкості до помилок. Оскільки кожен мікросервіс функціонує незалежно, помилка одного з них не призводить до збою всієї системи. Наприклад, у разі відмови сервісу оплати, інші сервіси, як обробка замовлень або управління користувачами, можуть продовжувати працювати. Такий підхід підвищує загальну надійність системи. Мікросервіси також дозволяють здійснювати швидкі оновлення й релізи. Завдяки тому, що кожен сервіс може бути оновлений незалежно, можна швидко впроваджувати нові функції або виправляти помилки, не чекаючи на загальне оновлення системи. Це також створює сприятливі умови для роботи розподілених команд, оскільки різні сервіси можуть розроблятися окремими командами, що підвищує продуктивність.

Однак архітектура мікросервісів має й певні недоліки. Вона може ускладнити процеси розробки та управління системою, оскільки кожен сервіс повинен бути інтегрований із іншими. Для цього потрібна добре продумана інфраструктура, що включає моніторинг, логування та забезпечення безпеки. Взаємодія між сервісами через мережу також може створювати проблеми. Мережеві запити збільшують час відгуку і підвищують ризик збоїв через неполадки в мережі, що вимагає налаштування додаткових механізмів надійності, таких як тайм-аути та повтори запитів.

Управління даними в мікросервісній архітектурі також стає складнішим, оскільки кожен сервіс може мати власну базу даних. Це створює додаткові труднощі у підтримці узгодженості та синхронізації даних між сервісами. Тестування системи стає більш складним, адже потрібно тестувати не лише окремі мікросервіси, а й їх взаємодію між собою. Це ускладнює організацію тестувань і вимагає автоматизації процесів. Нарешті, мікросервісна архітектура часто потребує складнішої інфраструктури для управління контейнерами, оркестрацією та моніторингом, що може підвищити загальні витрати на розробку і підтримку.

Архітектура мікросервісів дозволяє створювати складні, масштабовані та стійкі до збоїв системи. Вона надає гнучкість у виборі технологій, полегшує масштабування окремих частин додатка і прискорює процес розробки. Однак, цей підхід також супроводжується підвищеною складністю у розробці, тестуванні та підтримці інфраструктури. Тому його використання виправдане, якщо система потребує високого рівня масштабованості, незалежності компонентів та гнучкості в розробці.

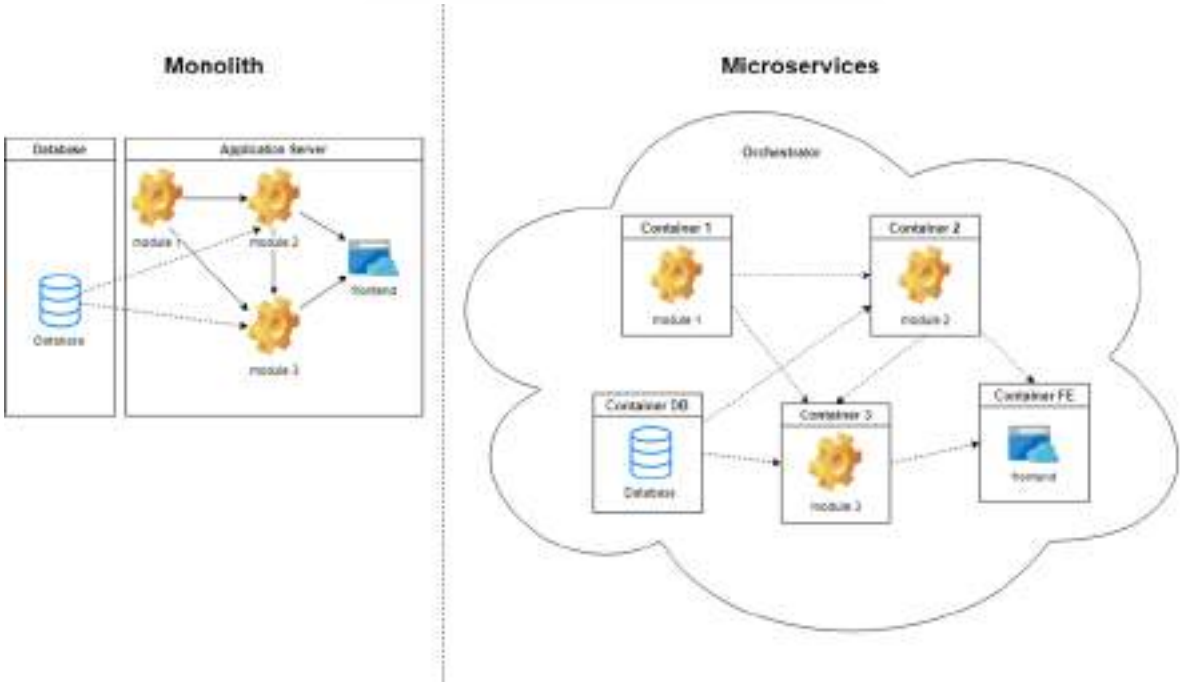

Рис. 8 – Порівняння монолітної та мікросервісної архітектури

2.2 Вибір технологій та інструментів для розробки

Вірність вибору технологічного стеку є важливим фактором успіху в реалізації проєкту. У цьому розділі будуть розглянуті ключові технології та інструменти, які були використані для розробки веб-додатку. Буде надано опис їх можливостей, переваг та недоліків.

Мова програмування JavaScript

JavaScript є високорівневою, інтерпретованою мовою програмування, яка використовується для розробки веб-додатків. Вона є однією з найпопулярніших мов програмування, оскільки надає багатий набір функцій і можливостей для створення динамічних інтерактивних веб-сторінок.

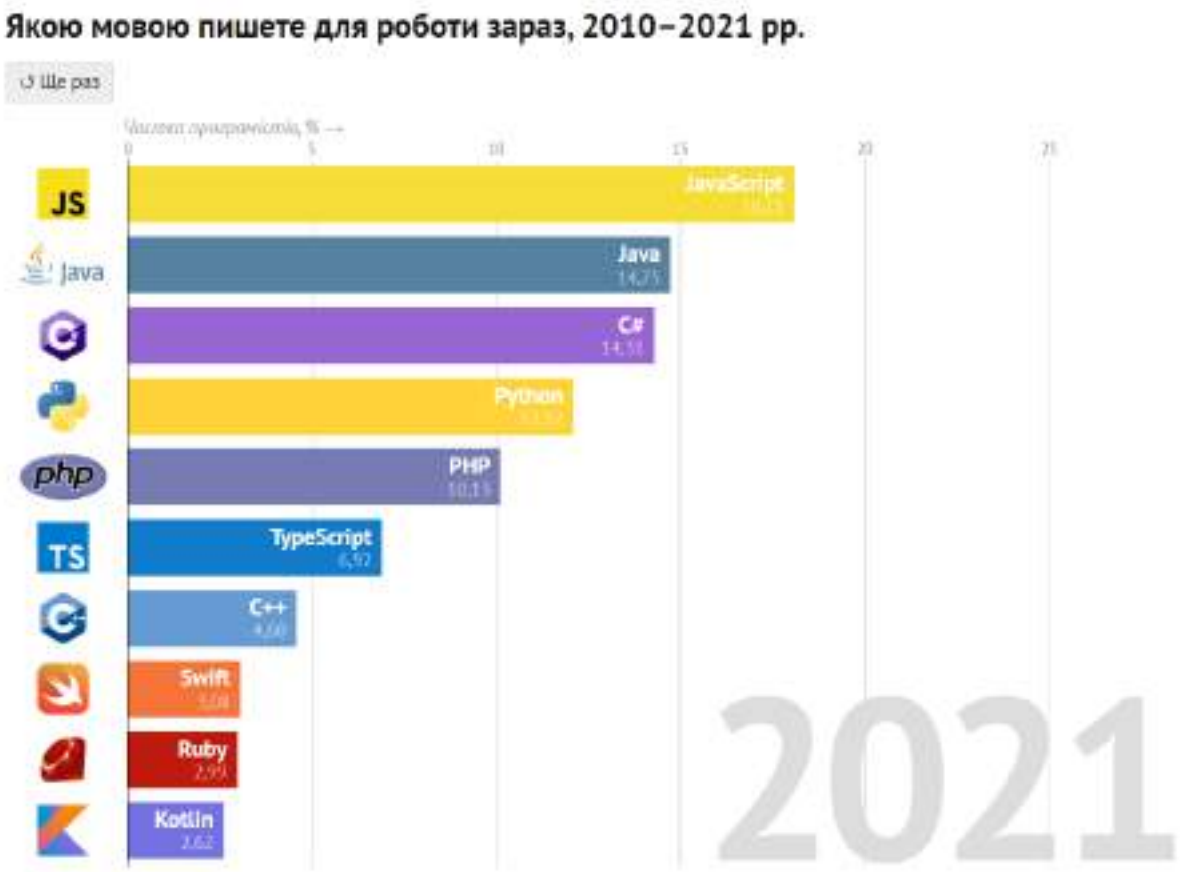

Рис. 9 – Рейтинг популярності мов в Україні (за DOU)[22] за 2021 рік

Основні плюси мови програмування JavaScript:

- Універсальність: JavaScript використовується як на стороні клієнта (у браузері), так і на стороні сервера (за допомогою Node.js). Це дає можливість розробляти повнофункціональні веб-додатки як на фронтенді, так і на бекенді, що спрощує розробку повного стеку веб-додатків.
- Інтерактивність: JavaScript надає засоби для взаємодії з користувачем на веб-сторінці, такі як валідація форм, обробка подій, анімація та зміна контенту. Він дозволяє створювати динамічні інтерфейси, які реагують на взаємодію користувача.
- Широке співтовариство: JavaScript має велику та активну спільноту розробників. Це означає, що існує безліч ресурсів, бібліотек,

фреймворків та інструментів, які полегшують розробку та розширення JavaScript-додатків.

- Переносимість: JavaScript підтримується на більшості сучасних браузерів і операційних систем. Це дозволяє запускати JavaScript-код на різних платформах без потреби переписувати його.

Основні мінуси мови програмування JavaScript:

- Обмежена безпека: JavaScript має обмежені можливості для контролю доступу до ресурсів користувача. Це означає, що існує потенційна загроза безпеки, така як скрипти, що виконуються на стороні клієнта, які можуть бути зловживані для зламу та крадіжки даних.
- Продуктивність: JavaScript інтерпретується, а не компілюється, що може призвести до меншої продуктивності порівняно з мовами, які компілюються перед виконанням. Однак, використання нових технологій, таких як JIT-компіляція, дозволяє значно покращити продуктивність JavaScript-коду.
- Нестриманість типів даних: JavaScript є динамічно типізованою мовою, що означає, що типи даних можуть змінюватися під час виконання програми. Це може призводити до непередбачуваної поведінки та помилок, особливо у великих проектах.

JavaScript використовується для створення інтерактивних веб-сторінок, односторінкових додатків, веб-ігор, мобільних додатків та навіть настільних додатків за допомогою фреймворків, таких як Electron. Вона є невід'ємною частиною розробки веб-додатків і забезпечує можливість створювати багатофункціональні, динамічні та інтерактивні веб-додатки.

Проте використання «чистого» JavaScript рідко використовується на практиці, зазвичай краще розробляти додатки за допомогою бібліотек або фреймворків[23]. Їх використання має декілька переваг, які перелічені нижче:

Прискорення розробки: Фреймворки надають готові структури та шаблони, які спрощують створення додатків. Вони забезпечують готові компоненти, модулі та інструменти, що допомагають прискорити процес розробки. Фреймворки забезпечують високу рівень абстракції, що дозволяє зосередитися на бізнес-логіці додатку, а не на деталях реалізації.

Єдність стандартів: Фреймворки надають стандартизований підхід до розробки, що полегшує колаборацію між розробниками. Вони мають визначені правила та конвенції, які спрощують структурування та організацію коду. Це полегшує розуміння та підтримку проекту навіть для нових розробників.

Розширюваність: Фреймворки надають можливості для розширення функціональності за допомогою додаткових модулів, бібліотек та плагінів. Це дозволяє використовувати готові рішення для загальних задач і зосередитися на унікальних аспектах проекту.

Підтримка та спільнота: Популярні фреймворки мають активну спільноту розробників, яка надає документацію, підручники, приклади коду та відповіді на питання. Це допомагає вирішувати проблеми та швидко знаходити відповіді на запитання.

Підтримка кросс-браузерності[24]: Фреймворки надають абстракцію від різних браузерних особливостей, допомагаючи забезпечити сумісність додатку з різними браузерами. Вони автоматично вирішують питання сумісності, оптимізації та роботи з різними версіями JavaScript-двигунів.

В наш час вже створено безліч бібліотек та фреймворків для більш комфортної роботи з JavaScript проте кожна з них надає певні вирішення проблем, яких немає в інших, тому потрібно правильно обрати оптимальну.

Зробимо коротке порівняння найбільших аналогів для того щоб обрати ідеальний варіант.

- React.js: React є одною з найпопулярніших бібліотек JavaScript для розробки користувацького інтерфейсу. Він дозволяє розбити

інтерфейс на компоненти, що полегшує керування станом додатку та створення перевикористовуваних компонентів. React використовує віртуальний DOM[25] для ефективного оновлення інтерфейсу.

- Angular[26]: Angular є повнофункціональним фреймворком JavaScript, який дозволяє розробляти складні односторінкові додатки. Він надає широкий набір функціональності, таку як залежності впродовж компонентів, маршрутизацію, форми та багато іншого. Angular пропонує структуру та набір правил для розробки додатків.
- Vue.js[27]: Vue є прогресивним фреймворком JavaScript, який легкий у вивченні та використанні. Він дозволяє побудувати інтерфейс, який складається з компонентів, забезпечуючи простоту управління станом та взаємодію з компонентами. Vue має активну спільноту та широкий вибір плагінів для розширення його функціональності.
- Ember.js[28]: Ember є повнофункціональним фреймворком для розробки веб-додатків. Він надає широкий набір інструментів та стандартів, які полегшують розробку складних додатків. Ember забезпечує конвенцію над конфігурацією, що полегшує колаборацію та розширення проектів.

Бібліотека React.js

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці застосунків.

Вибір випав саме на цю бібліотеку через деякі причини, наприклад, масштабованість. React.js добре справляється з розробкою великих та складних проектів, де необхідно керувати багато компонентами та станом. Він забезпечує чудову масштабованість та легку розширюваність. Також немало важливими причинами є швидкість розробки завдяки

компонентному підходу, багаторазовому використанню компонентів та активній спільноті, розробка на React.js є швидкою та ефективною. Багато завдань можна вирішити за допомогою готових компонентів та бібліотек та підтримка від Facebook, адже бібліотека створена та підтримується Facebook, що гарантує його стабільність, постійне оновлення та актуальність. Компанія активно використовує React.js у власних проектах, що свідчить про його надійність та розширені можливості.

Основна філософія полягає в розподілі інтерфейсу на незалежні компоненти, які можуть бути перевикористані та легко керуються станом додатку.

Ось деякі з особливостей React.js:

- React використовує віртуальний DOM (Document Object Model), що дозволяє забезпечити швидке та ефективне оновлення інтерфейсу. Віртуальний DOM - це внутрішня представлення структури інтерфейсу, яке знаходиться в пам'яті. Зміни в стані додатку призводять до оновлення віртуального DOM, після чого React виконує різницю між попереднім та новим станом та оновлює реальний DOM лише в тих місцях, де відбулися зміни. Цей підхід дозволяє зменшити навантаження на браузер та підвищити продуктивність додатку.

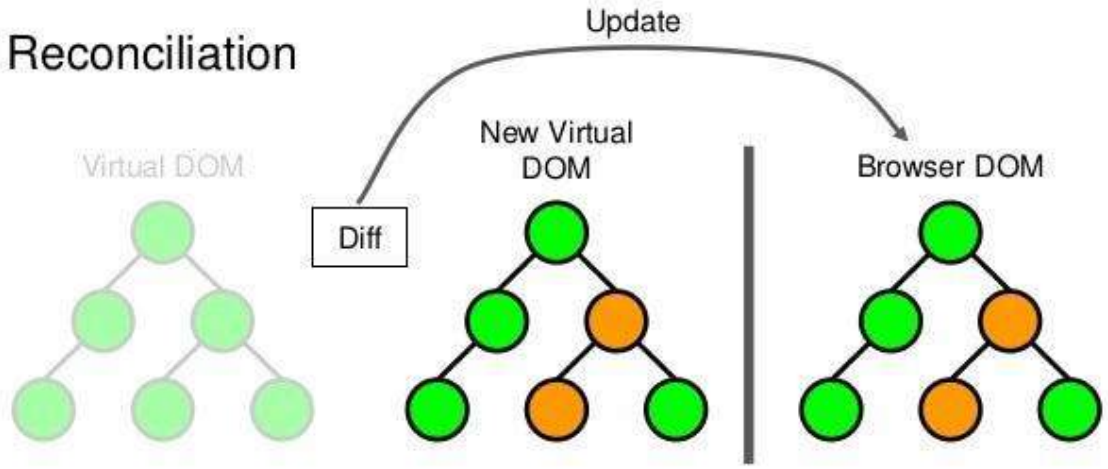

Рис. 10 – Діаграма використання DOM

- React пропонує компонентний підхід до розробки, де інтерфейс розбивається на незалежні компоненти. Кожен компонент може мати свій стан (state) та властивості (props) і може бути перевикористаний в різних частинах додатку. Цей підхід забезпечує модульність, розширюваність та покращує читабельність коду.

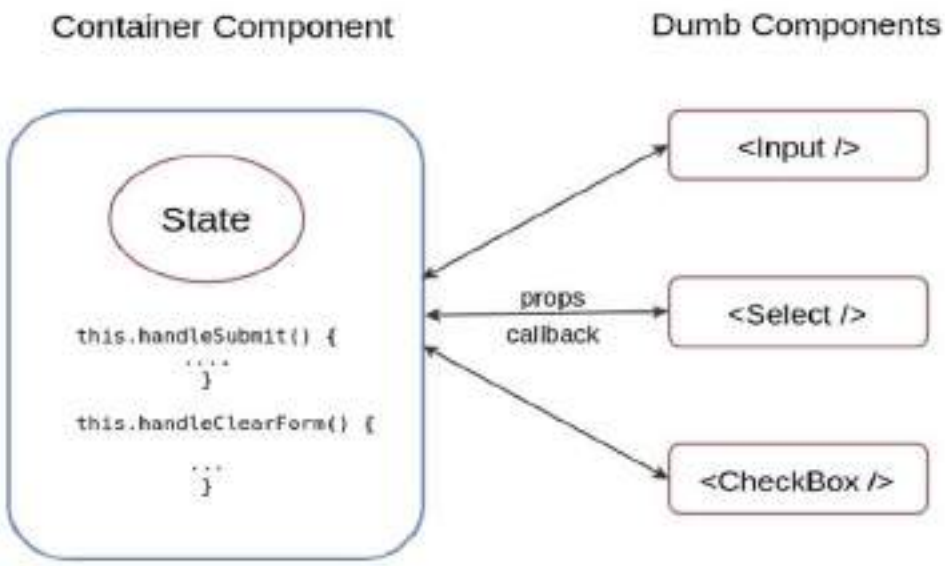

Рис. 11 – Діаграма використання компонентного підходу

- У React даних потік йде в одному напрямку, що сприяє простоті управління станом додатку. Зміни стану відбуваються через спеціальні функції-обробники, ініційовані від користувача або від змін в системі. Це сприяє відслідковуванню змін та полегшує розуміння поведінки додатку.

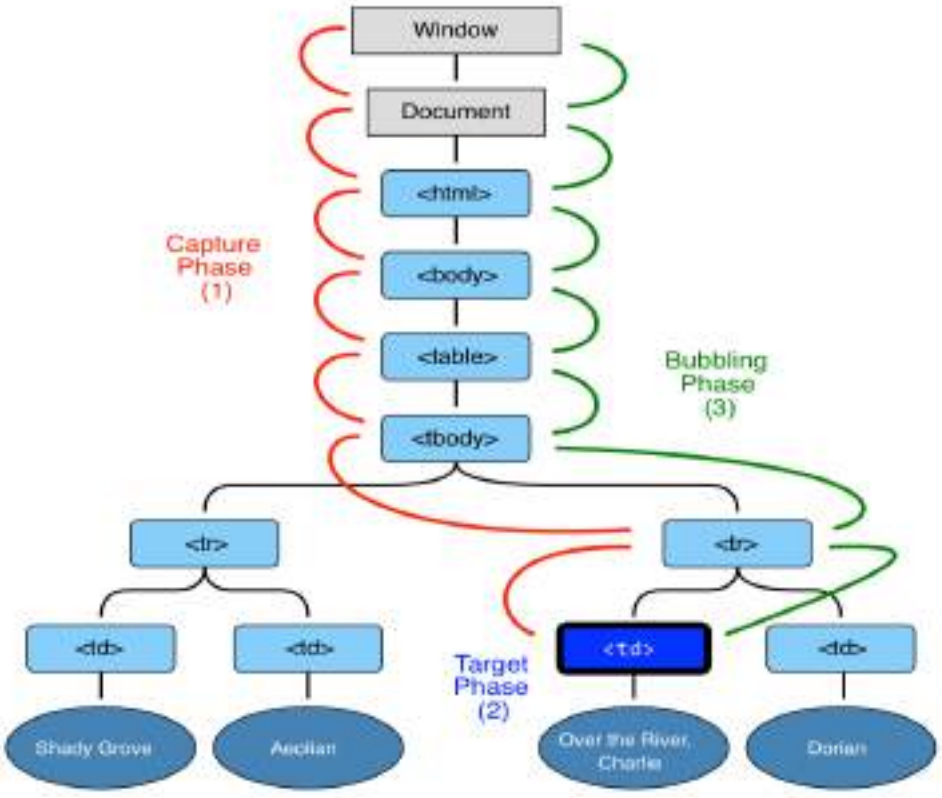

Рис. 12 – Діаграма однопоточного напрямку

Платформа Node.js

Хоч дана дипломна робота більш сфокусована на розробці Front-end частини, ні один великий веб-додаток не може існувати без Back-end. Для розробки серверної частини я обрав Node.js – відкрите, платформонезалежне середовище виконання JavaScript, яке дозволяє розробникам виконувати JavaScript-код поза браузером. Воно побудоване на двигуні V8, що розроблений компанією Google для браузера Chrome. Node.js використовує однопотокову асинхронну модель виконання, що

робить його особливо ефективним для розробки сумісних з масштабуванням та високонавантажених додатків.

Основною причиною такого вибору була можливість використовувати JavaScript як мову розробки як на фронтенді так і на бекенді, що спрощує взаємодію та дозволяє перевикористовувати код. Також немало важливою є висока продуктивність, оскільки Node.js працює на основні асинхронної моделі вводу/виводу (I/O), він може обробляти багато одночасних запитів без блокування інших операцій. Також Node.js підтримує масштабованість горизонтального та вертикального типів. Горизонтальна масштабованість включає розподілення додатків на багато серверів, щоб забезпечити більше ресурсів та обробляти більше запитів. Вертикальна масштабованість означає, що можна збільшити продуктивність додатку, оновивши апаратне забезпечення, яке він використовує. Так само як і бібліотека React.js – платформа має безліч бібліотек та модулів, які полегшують розробку, а також велику та активну спільноту розробників, яка надає підтримку, документацію та постійно вдосконалює середовище.

Visual Studio Code[29]

Visual Studio Code (VS Code) — це безкоштовне і відкрите інтегроване середовище розробки, створене компанією Microsoft, яке стало одним із найпопулярніших інструментів серед розробників програмного забезпечення. Це середовище пропонує широкі можливості для створення різних типів додатків і надає користувачам потужні функції, включаючи підтримку різних мов програмування, розширення через плагіни, інтелектуальне автодоповнення коду та вбудовану систему налагодження. Простий і зручний інтерфейс, разом із великою спільнотою та регулярними оновленнями, робить VS Code ідеальним вибором для професійних розробників і новачків.

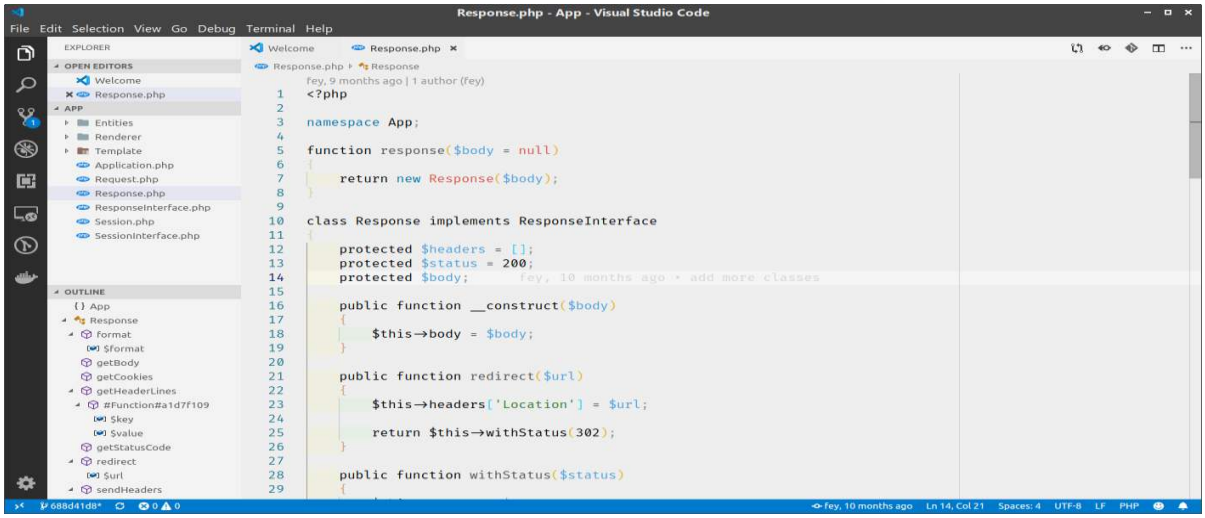

Рис. 13 – Початковий екран Visual Studio Code

Нижче перелічені деякі переваги використання VS Code:

- Кросплатформеність: VS Code підтримує Windows, macOS та Linux, що робить його доступним для розробників на різних платформах. Ви можете використовувати той самий редактор незалежно від операційної системи.
- Легкість використання та налаштування: VS Code має інтуїтивно зрозумілий інтерфейс користувача, що дозволяє швидко орієнтуватися та приступити до роботи. Він також надає багато налаштувань, які дозволяють адаптувати редактор під власні потреби.
- Широкий вибір розширень: VS Code має активну спільноту розробників, яка створює розширення для різних мов програмування, фреймворків та інструментів. Ви можете знайти розширення для підтримки своїх улюблених технологій та покращення робочого процесу.
- Інтеграція з іншими інструментами: VS Code інтегрується з багатьма популярними інструментами розробки, такими як системи керування версіями (Git)[30], інструменти тестування, засоби розгортання та багато інших. Це полегшує інтеграцію з існуючими розробчими процесами.

- Висока продуктивність: VS Code працює досить швидко, навіть при роботі з великими проектами. Він має вбудовані оптимізації та функції, що допомагають забезпечити ефективну роботу з кодом.

Visual Studio Code має досить мінімалістичний дизайн в порівнянні з іншими популярними IDE, проте це не заважає йому містити у собі безліч обов'язкових, у розробці, функцій. Основними з них є:

- Редагування коду: VS Code надає розширену підтримку для редагування коду різних мов програмування. Воно підтримує підсвічування синтаксису, автодоповнення, перевірку помилок та багато інших корисних функцій, що полегшують написання коду.

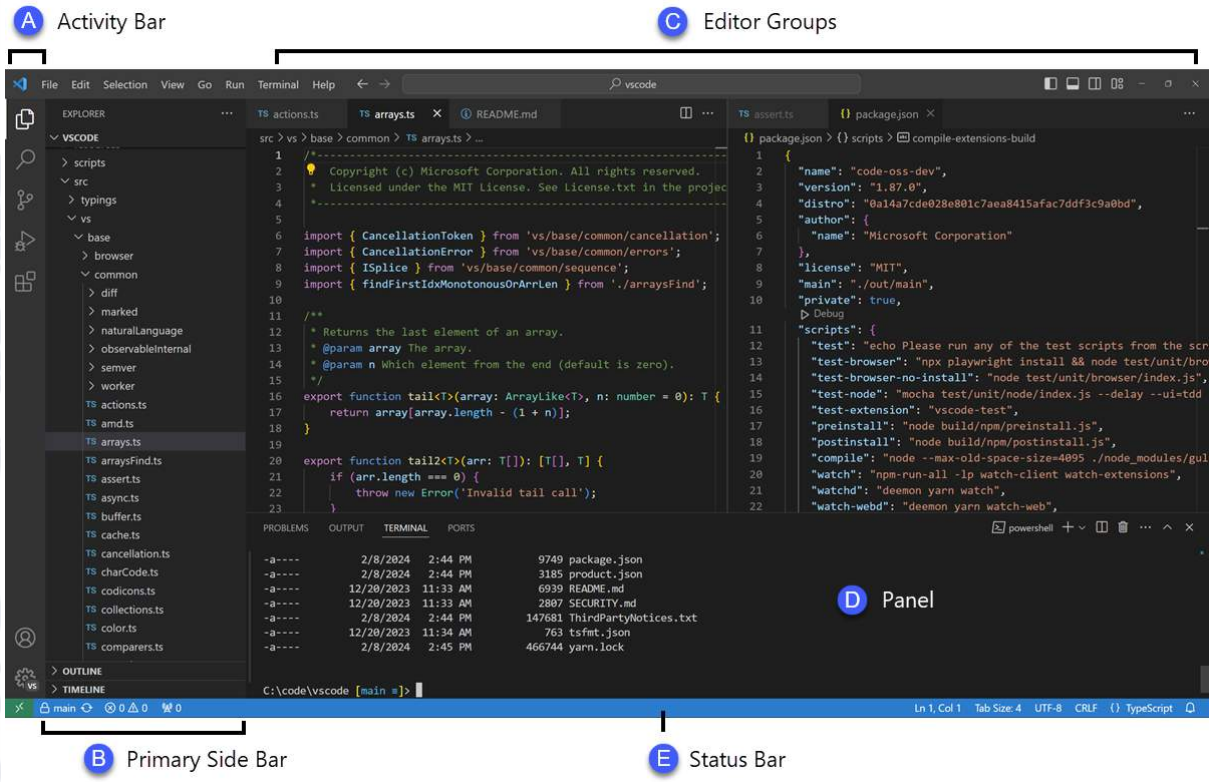

Рис. 14 – Редагування коду в середовищі VS code

- Розширення та плагіни: VS Code має широкий вибір розширень та плагінів, які дозволяють розширити його функціональність. Розробники можуть встановлювати розширення для підтримки конкретних мов програмування, фреймворків, інструментів та розширити можливості редактора за своїми потребами.

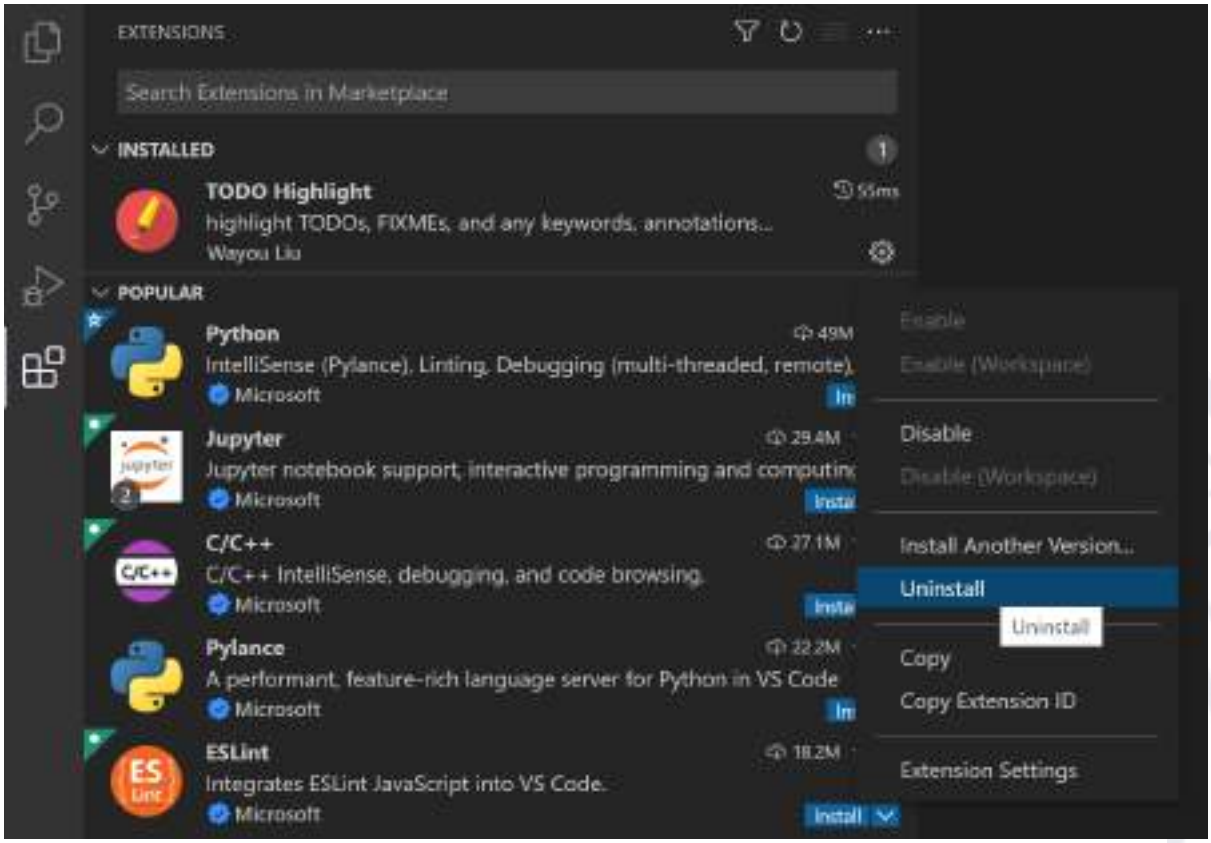

Рис. 15 – Бібліотека розширень та плагінів

- Відладка коду є однією з ключових функцій Visual Studio Code, яка пропонує користувачам широкий набір інструментів для ефективного виявлення та виправлення помилок. Серед основних можливостей — установка точок зупинки, що дозволяє зупинити виконання програми на певних етапах, а також крокування по коду для детального аналізу його виконання. Розробники можуть переглядати значення змінних під час виконання, що значно спрощує процес налагодження. Ці потужні функції роблять VS Code особливо зручним для тих, хто прагне швидко виявляти і виправляти проблеми у своїх програмах.
- Керування версіями: VS Code інтегрується з популярними системами керування версіями, такими як Git. Він надає інтерфейс для відстеження змін в коді, комітів, гілок, злиття та інших операцій, пов'язаних з керуванням версіями.

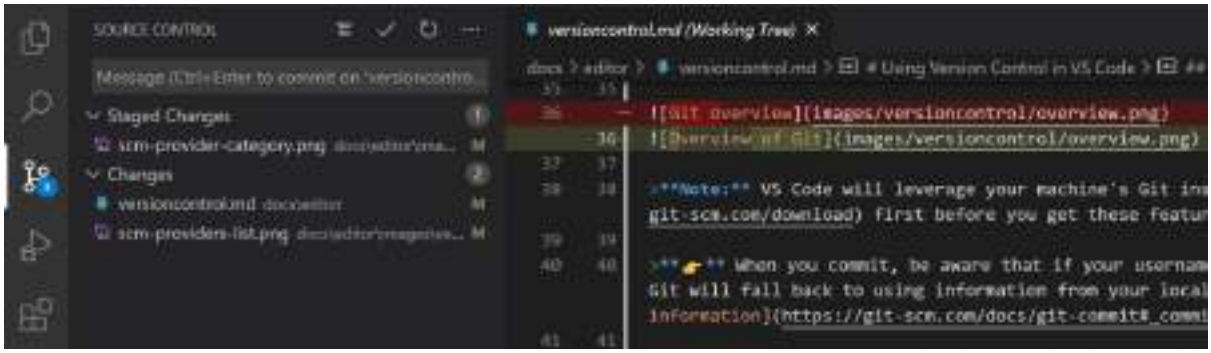

Рис. 16 – Режим керування версіями

Як висновок можна сказати, що Visual Studio Code це потужне та популярне середовище розробки з багатьма корисними функціями, зручним інтерфейсом та можливістю розширення. Використання VS Code дозволяє розробникам працювати ефективно та комфортно, прискорюючи процес розробки програмного забезпечення.

Висновок до розділу 2

У другому розділі було проаналізовано та спроєктовано архітектуру і технологічний стек веб-додатку "Електронний реєстр пацієнтів". Було обрано архітектуру типу клієнт-сервер із використанням REST API, яка забезпечує гнучкість, масштабованість і простоту підтримки. Під час вибору технологій для фронтенду обрано React.js завдяки його компонентному підходу та високій продуктивності, а для бекенду — Node.js, що дозволяє використовувати єдину мову програмування для обох частин. Також розглянуто альтернативні архітектури (монолітна та мікросервісна) і обґрунтовано вибір клієнт-серверної моделі як оптимальної для цього проєкту. Було встановлено стандарти та визначено інструменти, що забезпечують швидкість розробки, зручність роботи та відповідність сучасним вимогам до веб-додатків.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

3.1 База даних

База даних[31] є основним компонентом сучасних веб-додатків, особливо тих, які працюють із великою кількістю користувачів та інформації. У веб-додатку для ведення електронного реєстру пацієнтів база даних використовується для зберігання, організації та швидкого доступу до важливих даних, таких як дані про користувачів, медичні записи, історії хвороб та інше. Ефективна організація даних забезпечує надійність роботи системи, швидкість виконання запитів, а також безпеку конфіденційної інформації.

Для цього проєкту було обрано MongoDB[32] — сучасну документно-орієнтовану базу даних, яка ідеально підходить для динамічних і масштабованих веб-додатків. Нижче розглянуто її переваги та причини вибору для роботи з технологічним стеком ReactJS та NodeJS.

Переваги MongoDB:

- Гнучка схема даних - MongoDB дозволяє зберігати дані без жорсткої схеми, що є великою перевагою під час розробки динамічних систем. У випадку змін у вимогах проєкту розробник може легко додати нові поля до існуючих документів без необхідності модифікувати структуру бази даних або оновлювати всі старі записи. Це знижує ризики та витрати на обслуговування і вдосконалення системи, особливо у проєктах, які активно розвиваються.
- Документно-орієнтована модель - Дані в MongoDB зберігаються у вигляді JSON-подібних документів (BSON). Ця структура природно відповідає об’єктам у JavaScript/TypeScript, що робить MongoDB ідеальним вибором для проєктів, написаних на NodeJS. Завдяки цьому, серіалізація та десеріалізація даних між клієнтом і сервером

відбувається швидко й ефективно, що значно покращує продуктивність.

- Висока продуктивність - MongoDB забезпечує високу швидкість операцій читання та запису, навіть при роботі з великими обсягами даних. Вона оптимізована для паралельної обробки запитів, що особливо важливо для систем із великою кількістю одночасних користувачів. Для зберігання даних MongoDB використовує індекси, які значно прискорюють виконання пошукових запитів.
- Масштабованість - MongoDB підтримує горизонтальне масштабування через шардінг. Це означає, що дані можуть бути розподілені між декількома серверами, що дозволяє працювати з великими наборами даних і підтримувати стабільну роботу навіть при значному навантаженні. Такий підхід ідеально підходить для масштабування веб-додатків із динамічним ростом.
- Інтеграція з сучасними інструментами - MongoDB має широкий спектр інструментів і драйверів для інтеграції. Наприклад, Mongoose дозволяє працювати з MongoDB у середовищі NodeJS через ORM (Object-Relational Mapping), що забезпечує зручне визначення схем і валідаторів для даних. Це знижує ризик помилок і покращує якість коду.

Чому MongoDB підходить для ReactJS + NodeJS?

Природна робота з JSON-структурами[33]. У ReactJS та NodeJS дані часто передаються у вигляді JSON-об’єктів. MongoDB, яка використовує BSON, дозволяє мінімізувати перетворення між різними форматами даних. Це спрощує процес розробки та забезпечує високу швидкість передачі даних між клієнтом і сервером.

Швидка розробка і гнучкість. У динамічних проєктах, таких як веб-додатки, часто виникає потреба у швидкому внесенні змін до структури даних. MongoDB, з її гнучкою схемою, дозволяє безболісно адаптуватися до нових вимог. ReactJS та NodeJS також орієнтовані на швидку розробку, тому їх поєднання з MongoDB дозволяє зберігати високу продуктивність на всіх етапах розробки.

Асинхронна природа NodeJS. NodeJS відомий своєю асинхронною архітектурою, що дозволяє обробляти велику кількість запитів одночасно. MongoDB органічно працює з цією архітектурою завдяки підтримці асинхронних операцій, що знижує затримки у взаємодії з базою даних.

Підтримка масштабування. ReactJS забезпечує масштабованість на клієнтському рівні, а NodeJS і MongoDB дозволяють масштабувати сервер і базу даних. MongoDB, з її горизонтальним масштабуванням, дозволяє легко додавати сервери для обробки зростаючого потоку даних, що ідеально підходить для проєктів, які активно розширюються.

База даних веб-додатку складається з кількох колекцій, а саме: users, doctors, records, calendars, events, payments, healthInformations. Дані в колекціях пов'язані між собою через унікальні ідентифікатори (_id). Наприклад:

- Користувач із users може бути пацієнтом із прив’язаними медичними записами в records та даними про здоров’я в healthInformation.
- Лікар із doctors пов'язаний із записами у calendar та конкретними прийомами в events.
- Платежі в payments пов’язані як із користувачами, так і з наданими послугами чи подіями в events.

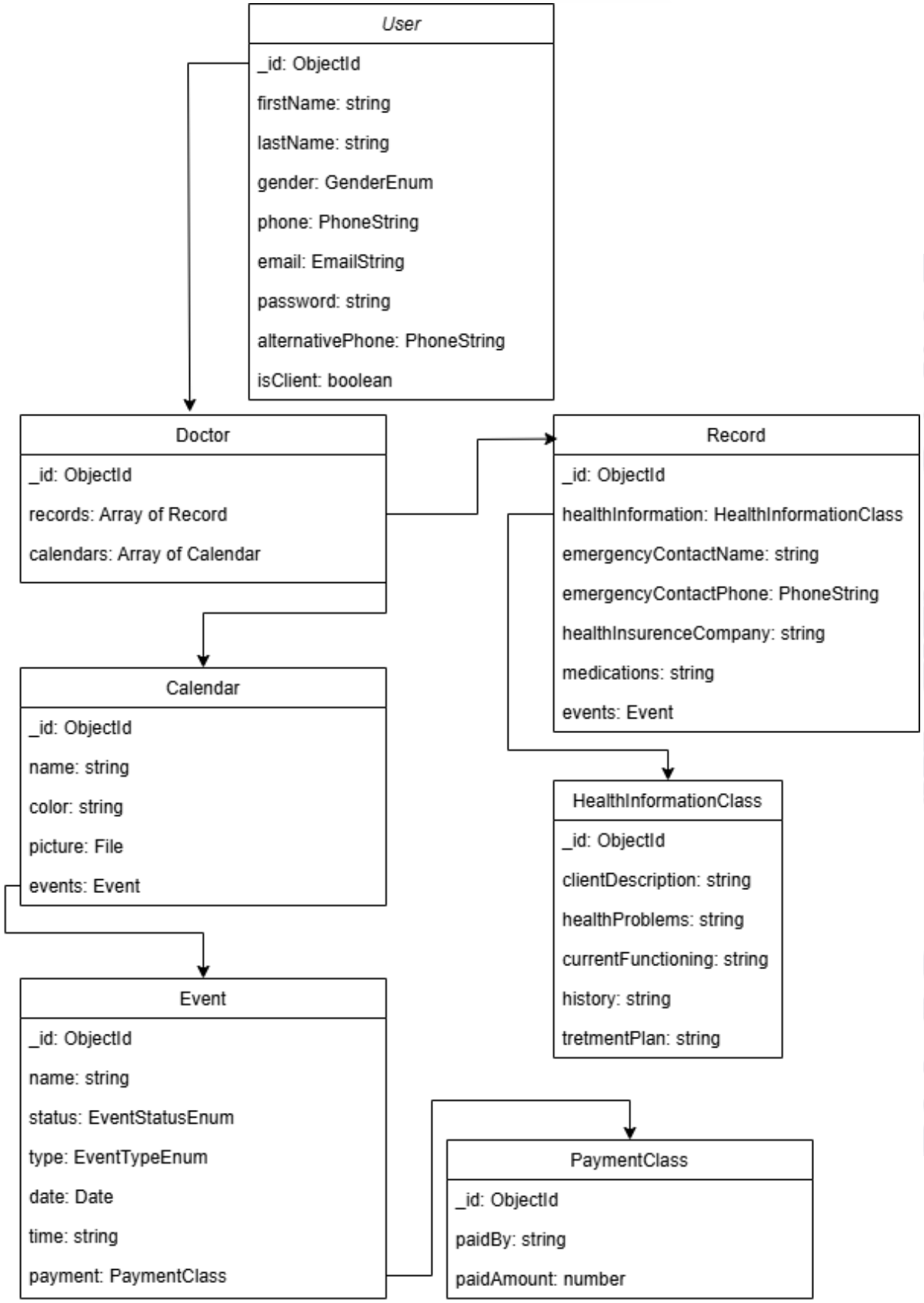

Рис. 17 - Колекції бази даних

Кожна із зазначених колекцій буде розглянута окремо та більш детально, оскільки загальний опис не завжди розкриває всі особливості їх структури та призначення. Детальний опис кожної колекції наведено нижче:

- users

Ця колекція є основою для управління користувачами системи. Вона використовується для зберігання основних даних, а також для встановлення зв’язків із іншими колекціями, такими як records (медичні записи), payments (платежі) та events (події) та містить наступні поля:

  - _id (ObjectId) – це унікальний ідентифікатор документа, який автоматично генерується MongoDB. Він забезпечує однозначну ідентифікацію кожного користувача у колекції.
  - firstName (string) – аргумент, що зберігає ім'я користувача. Це поле використовується для звернення до користувача у системі.
  - lastName (string) – аргумент, що зберігає прізвище користувача. Воно дозволяє ідентифікувати користувача серед інших осіб із подібними іменами.
  - gender (string, enum) – аргумент, який зберігає інформацію про стать користувача. Може приймати значення Male, Female або Other. Це поле використовується для персоналізації взаємодії з користувачем.
  - phone (string) – аргумент, що зберігає основний контактний номер телефону користувача. Використовується для зв'язку або ідентифікації.
  - email (string) – аргумент, що зберігає електронну пошту користувача. Це поле необхідне для реєстрації, входу в систему та отримання сповіщень.

- alternativePhone (string) – необов'язкове поле, яке зберігає додатковий номер телефону. Воно використовується для резервного зв’язку у разі недоступності основного.
- isClient (boolean) – аргумент, який визначає, чи є користувач пацієнтом системи (true) або має іншу роль (false). Це поле допомагає розмежувати функціонал доступу до системи для різних категорій користувачів.

- doctors

Колекція doctors використовується для управління даними про лікарів. Вона допомагає системі забезпечувати зручний доступ до їхніх графіків, а також до пов’язаних з ними медичних записів. Ця структура забезпечує швидкий пошук і зв’язок з іншими колекціями, такими як records (медичні записи) та calendar (розклади).

- _id (ObjectId) – унікальний ідентифікатор лікаря, що автоматично генерується MongoDB. Використовується для однозначної ідентифікації кожного лікаря у колекції.
- records (Array of Record) – список посилань на медичні записи, пов’язані з лікарем. Кожен елемент масиву є об'єктом, який містить інформацію про пацієнтів, що проходили обстеження чи лікування у цього лікаря. Це поле дозволяє швидко отримувати історію взаємодій лікаря з пацієнтами.
- calendars (Array of Calendar) – список посилань на розклади лікаря. Кожен елемент масиву містить інформацію про графік роботи, прийоми, заплановані зустрічі та інші події, що стосуються діяльності лікаря.

- records

Колекція records забезпечує зберігання важливої інформації про стан здоров’я пацієнтів, їхній контакт для екстрених випадків, дані про страхування та медикаменти. Вона також слугує зв'язком між

пацієнтом та іншими аспектами медичних послуг, такими як події (events) та загальні медичні дані (healthInformation).

- healthInformation (HealthInformationClass) – об'єкт, що містить загальну медичну інформацію про пацієнта. Включає дані про хронічні захворювання, алергії, результати аналізів та інші важливі аспекти здоров’я пацієнта.
- emergencyContactName (string) – ім'я контактної особи для екстрених випадків. Використовується для швидкого зв’язку у разі надзвичайних ситуацій, пов'язаних зі станом здоров’я пацієнта.
- emergencyContactPhone (string) – номер телефону екстреного контакту. Дозволяє оперативно зв'язатися з відповідальною особою у разі потреби.
- healthInsuranceCompany (string) – назва страхової компанії, яка обслуговує пацієнта. Це поле використовується для забезпечення взаємодії з медичним страхуванням.
- medications (string) – список медикаментів, які приймає пацієнт. Може містити інформацію про поточні курси лікування, дозування та призначення лікаря.
- events (Event) – об'єкт, що містить інформацію про події, пов'язані з медичними записами. Наприклад, це можуть бути прийоми, заплановані обстеження або госпіталізації.

- healthInformations

Колекція healthInformationClass забезпечує зберігання повної та актуальної інформації про медичний стан пацієнтів. Вона використовується для аналізу стану здоров’я, планування лікування та надання комплексного підходу до обслуговування пацієнтів.

- clientDescription (string) – опис стану пацієнта, наданий лікарем. Це поле може включати суб’єктивні та об’єктивні дані, отримані під час обстеження або консультації.
- healthProblems (string) – список основних проблем зі здоров’ям, діагнозів або симптомів, які має пацієнт. Це поле слугує основою для формування лікувальних заходів.
- currentFunctioning (string) – опис поточного функціонального стану пацієнта, включаючи фізичні, емоційні або когнітивні аспекти.
- history (string) – історія хвороби пацієнта, включаючи попередні діагнози, проведені лікування, госпіталізації або травми. Це поле є важливим для аналізу змін стану здоров’я з часом.
- treatmentPlan (string) – план лікування, запропонований лікарем. Може включати рекомендації щодо медикаментів, фізіотерапії, дієти або інших методів терапії.

- events

Колекція events дозволяє зберігати деталі подій, пов'язаних з пацієнтами. Вона допомагає організувати прийоми, медичні процедури та інші важливі заходи, забезпечуючи ефективний менеджмент подій у системі. Це також дозволяє легко відстежувати статус події та пов'язану з нею оплату через поле payment.

- name (string) – назва події. Це поле містить короткий опис або назву заходу, наприклад, прийом у лікаря, обстеження або планова операція.
- status (EventStatusEnum) – статус події, що вказує на її поточний стан. Може приймати значення, такі як Planned(запланована), Completed (завершена), Cancelled (скасована), або інші варіанти відповідно до потреб системи.

- type (EventTypeEnum) – тип події, що вказує на характер заходу. Може бути таким, як Physical (фізична зустріч) або Video (онлайн зустріч)
- date (Date) – дата проведення події. Це поле зберігає інформацію про день, коли подія відбудеться або відбулася.
- time (string) – час початку події. Використовується для точного визначення, коли подія має відбутися.
- payment (PaymentClass) – об'єкт, що містить інформацію про оплату події. Включає дані про суму, метод оплати, а також статус платежу.

- payments

Колекція PaymentClass допомагає вести облік оплат за медичні послуги, зберігаючи дані про метод оплати та суму. Вона є важливою частиною процесу управління фінансами в медичному закладі, оскільки дозволяє зв’язувати платежі з конкретними подіями або медичними записами, полегшуючи контроль за фінансами.

- paidBy (string) – метод оплати, що вказує, яким чином здійснено платіж. Може бути значенням "Cash" (готівка) або "Card" (картка), залежно від способу, яким пацієнт або інша особа здійснила оплату.
- paidAmount (number) – сума, яку було сплачено за медичну послугу чи подію. Це поле зберігає точну величину платежу, що відображає вартість послуги або заходу.

Власне, на цьому огляд колекцій закінчено. Для кожної із існуючих був наведений детальний опис з поясненням про кожне існуюче поле та як воно впливає на роботу і які ціль має.

3.2 Реалізація графічного інтерфейсу

Графічний інтерфейс користувача (UI – User Interface)[34] є одним із ключових елементів будь-якого сучасного веб-додатку. Саме він забезпечує користувачам зручний доступ до функціоналу системи, сприяючи інтуїтивній та ефективній взаємодії з додатком. Водночас, якісний інтерфейс користувача значно підвищує загальне враження від використання додатку, що є критичним фактором у конкурентному цифровому середовищі.

Не менш важливим аспектом є досвід користувача (UX – User Experience)[35], який охоплює всі аспекти взаємодії користувача з додатком, включаючи зручність навігації, швидкість виконання операцій, привабливість дизайну та доступність функцій. Збалансований підхід до дизайну UI/UX дозволяє створити веб-додаток, який не лише виглядає естетично, але й задовольняє потреби користувачів, полегшуючи виконання їхніх завдань.

У контексті нашого веб-додатку, орієнтованого на електронний реєстр пацієнтів, якісний графічний інтерфейс є особливо важливим. Він дозволяє медичному персоналу швидко знаходити потрібну інформацію, зручно керувати записами пацієнтів і ефективно взаємодіяти з іншими функціями системи. Простота та зрозумілість інтерфейсу сприяють зниженню ймовірності помилок під час роботи, що є критичним для медичних систем.

Таким чином, реалізація графічного інтерфейсу є одним із ключових етапів розробки нашого веб-додатку і її деталі я розгляну у цьому підрозділі.

Графічний інтерфейс мого додатку складається з 10 повноцінних сторінок, які відповідають певному функціоналу, їх я розгляну далі:

Першою сторінкою є вітальний екран, що містить форму для входу в додаток. Форма в свою чергу має два обов’язкових поля: «E-mail» та «Пароль». Задля покращення UX (user experience) кожне поле також містить placeholder в якому написані підказки для користувача, що саме потрібно ввести. Також на цьому екрані є кнопка “Створіть аккаунт”, яка перенаправляє користувача на екран реєстрації.

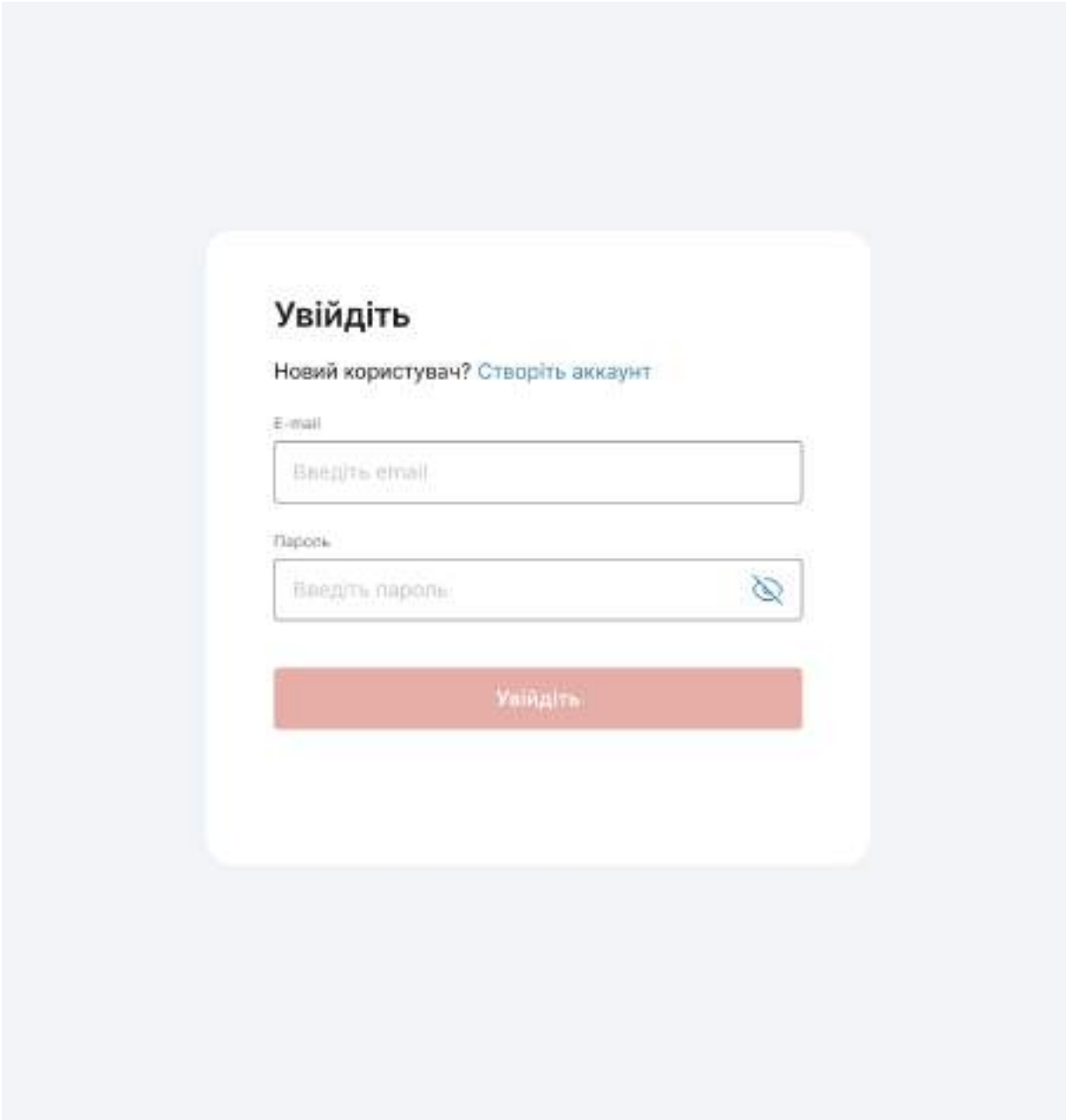

Рис. 18 - «Вітальний екран»

Сторінка реєстрації є важливим компонентом веб-додатку, оскільки саме через неї нові користувачі отримують доступ до системи. Ця сторінка забезпечує збір необхідної інформації для створення облікового запису, а також гарантує безпеку та юридичну відповідність процесу реєстрації.

Одним із перших полів, які користувач заповнює, є «Ім'я». Це поле призначене для введення особистого імені користувача. Ім'я необхідне для персоналізації взаємодії з системою, наприклад, для відображення вітальних повідомлень або формування звітів. Воно також створює враження, що система орієнтована на користувача, підвищуючи його довіру та комфортність роботи.

Наступним є поле «Прізвище», яке доповнює інформацію про користувача. Введення прізвища дозволяє більш точно ідентифікувати користувача серед інших записів у системі, що є особливо важливим у великих базах даних. Прізвище часто використовується для формальних документів, листування або в записах пацієнтів, тому його наявність є критично важливою для правильного функціонування додатку.

Поле «Email» виконує декілька важливих функцій. По-перше, це унікальний ідентифікатор користувача в системі, що дозволяє забезпечити унікальність облікового запису. По-друге, електронна пошта використовується для авторизації, відновлення пароля та надсилання важливих повідомлень, таких як нагадування чи оновлення. Електронна пошта також слугує засобом для комунікації між користувачем та адміністрацією додатку, що забезпечує оперативність і зручність взаємодії.

Одним із ключових елементів безпеки облікового запису є «Пароль». Це поле дозволяє користувачам створювати захищений доступ до свого облікового запису. Пароль має бути достатньо складним, щоб запобігти можливості злому, і водночас таким, який користувач легко запам'ятає. Щоб уникнути помилок при введенні пароля, на сторінці також є поле «Повторіть пароль», яке дозволяє користувачу підтвердити правильність

введених даних. Це зменшує ризик помилок і забезпечує впевненість у встановленні бажаного пароля.

Окрім полів для введення даних, важливим елементом сторінки є чекбокс із прийняттям умов та політики конфіденційності. Перед тим як завершити реєстрацію, користувач повинен погодитися з умовами «використання системи» та «політикою конфіденційності», які представлені у вигляді посилань. Це дозволяє інформувати користувачів про їхні права та обов’язки, а також забезпечує дотримання юридичних норм, таких як захист персональних даних. Посилання на ці документи дають змогу користувачам ознайомитися з правилами та переконатися в прозорості роботи системи.

Ще одним важливим елементом є «reCAPTCHA»[36], яка перевіряє, чи є користувач реальним, а не автоматизованим ботом. Це важливо для захисту системи від спаму та автоматизованих атак. Завдяки цьому елементу веб-додаток залишається стабільним і безпечним, а його ресурси не витрачаються на обробку небажаних запитів.

Усі поля та елементи на сторінці реєстрації спрямовані на те, щоб зробити процес створення облікового запису інтуїтивно зрозумілим, безпечним і відповідним сучасним стандартам роботи з даними. Вони забезпечують не лише базову функціональність, але й створюють позитивний користувацький досвід, що є критичним для успішного впровадження веб-додатку.

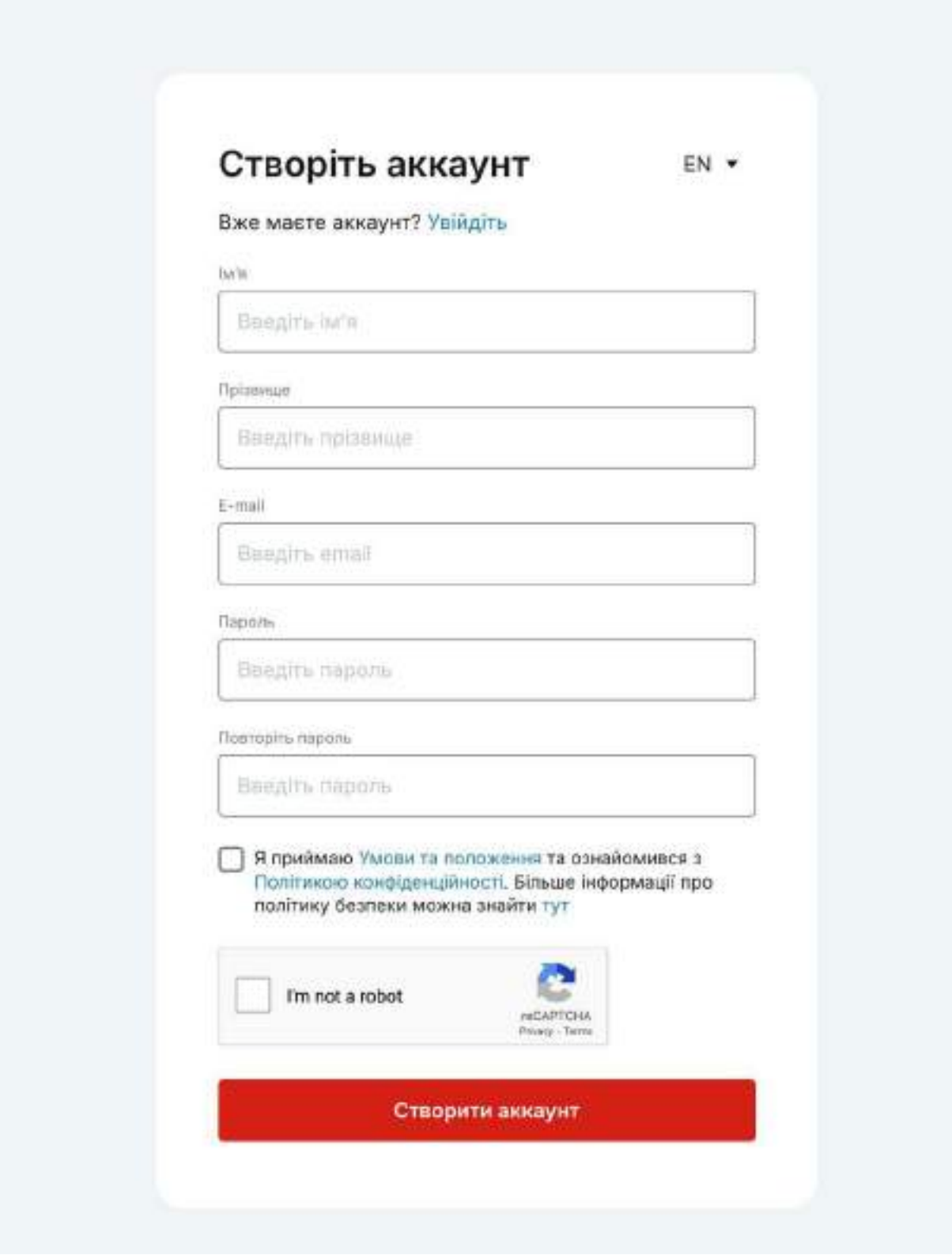

Рис. 19 - «Сторінка реєстрації»

Сторінка "Пацієнти" є ключовим елементом веб-додатку, який дозволяє лікарям та адміністраторам ефективно управляти списком пацієнтів. Вона надає зручний інтерфейс для перегляду, фільтрації та доступу до детальної інформації про кожного пацієнта.

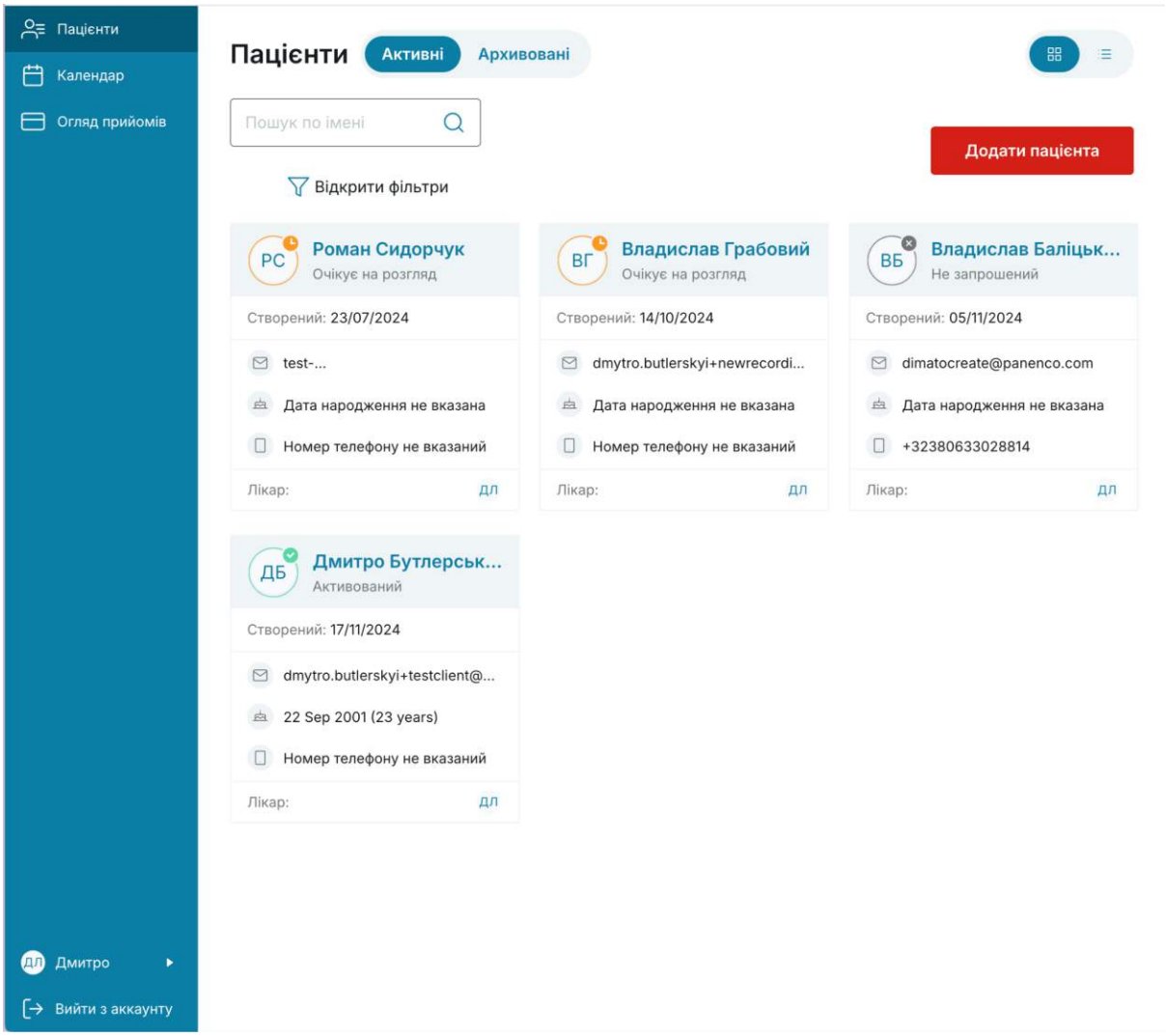

Рис. 20 - сторінка «Пацієнти»

- Головне меню

З лівого боку сторінки розташоване навігаційне меню, яке містить розділи:

  - Пацієнти: поточна активна сторінка для управління пацієнтами.
  - Календар: для перегляду графіка та додавання нових прийомів.
  - Огляд прийомів: для аналізу прийомів пацієнтів, де видно чи оплачений прийом та суму оплати.

Меню забезпечує швидкий доступ до основних функцій додатку, що сприяє зручності використання.

- Список пацієнтів

У центральній частині сторінки відображається список пацієнтів у вигляді карток. Кожна картка містить ключову інформацію про пацієнта:

  - Ім'я та прізвище пацієнта: дозволяє швидко ідентифікувати особу.
  - Статус: наприклад, "Активований", "Очікує на розгляд" або "Не запрошений".
  - Дата створення запису: вказує, коли було створено обліковий запис пацієнта.
  - Контактна інформація: електронна пошта, дата народження, номер телефону.
  - Лікар: інформація про закріпленого лікаря, якщо така є.

Ці картки дозволяють швидко оцінити стан кожного пацієнта та його взаємодію з системою.

- Фільтри та пошук

У верхній частині сторінки розташоване поле пошуку, яке дозволяє знайти пацієнта за ім’ям.

  - Кнопка "Відкрити фільтри" дозволяє застосовувати різноманітні параметри для більш точного відбору пацієнтів (наприклад, за статусом чи датою створення).
  - Табуляція активних і архівованих пацієнтів
  - Сторінка розділена на дві основні вкладки:
  - Активні: список пацієнтів, які перебувають під спостереженням або активно взаємодіють із системою.
  - Архівовані: пацієнти, чий статус було змінено (наприклад, завершено лікування або припинено співпрацю).

Це розділення сприяє впорядкованості та зручності в роботі з великими базами даних.

- Додаткові функції

У верхньому правому куті сторінки є кнопка "Додати пацієнта", яка дозволяє створити новий запис у базі даних. Також доступна кнопка перемикання режиму перегляду карток пацієнтів: таблиця чи плитки.

- Робота з профілем користувача
- У лівому нижньому куті знаходиться меню профілю, де користувач може переглянути свій обліковий запис або вийти із системи.

Бокова панель для додавання нового пацієнта є важливим функціональним компонентом сторінки "Пацієнти". Вона дозволяє адміністраторам або лікарям швидко створювати нові записи про пацієнтів із базовими даними.

У центральній частині панелі знаходяться три основних поля:

- Ім’я: це поле дозволяє ввести ім’я пацієнта. Воно є обов’язковим для заповнення, оскільки ідентифікує пацієнта в системі. Ім’я допомагає лікарям у персоналізації взаємодії з пацієнтом.
- Прізвище: поле для введення прізвища пацієнта, що також є обов’язковим. Це ключовий елемент для однозначної ідентифікації пацієнта, особливо у випадках, коли імена можуть збігатися.
- E-mail: це поле відповідає за введення електронної адреси пацієнта. Електронна адреса є важливою для подальшої комунікації: надсилання сповіщень, нагадувань про прийом або звітів. Поруч із полем розташовано іконку конверта, яка підкреслює функцію цього поля.

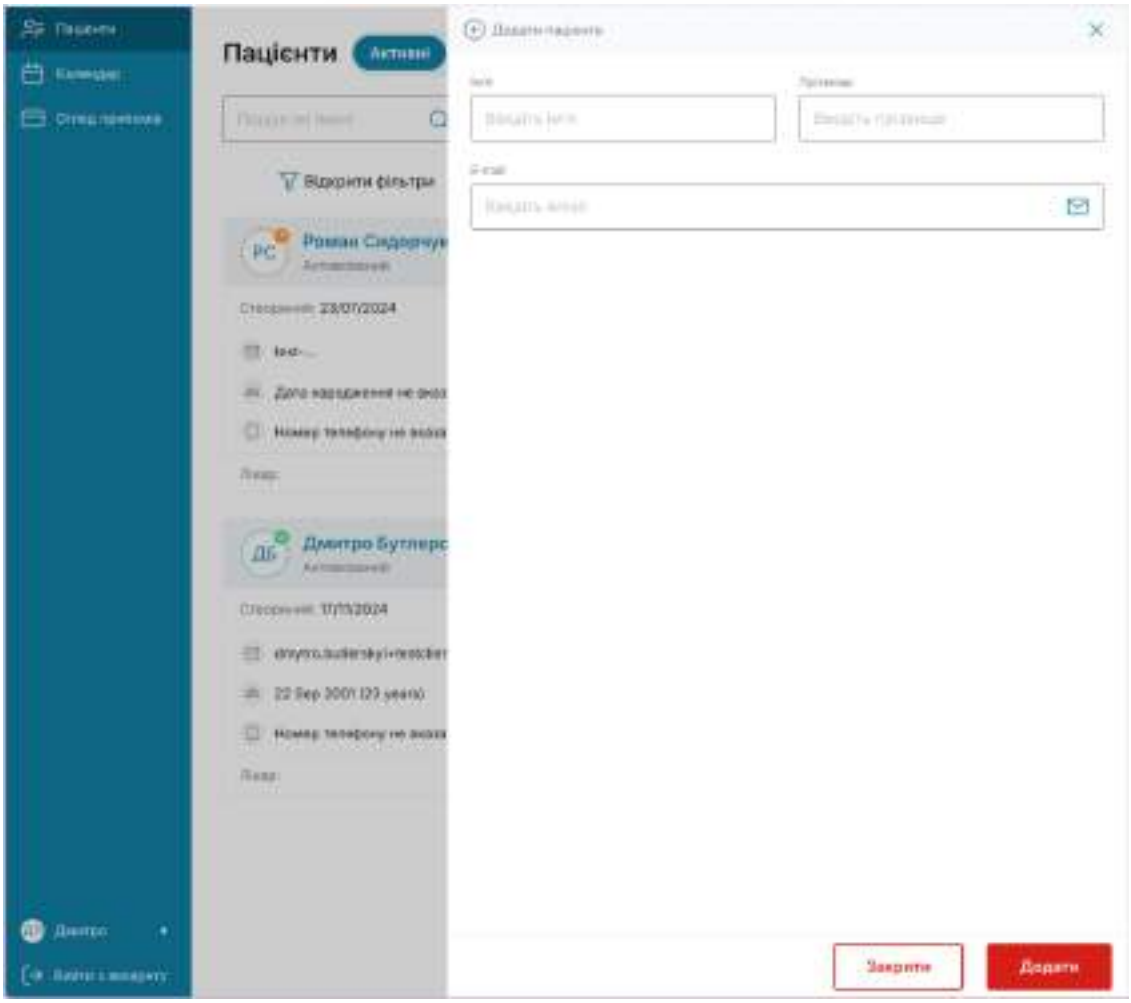

Рис. 21 - «Бокова панель»

Панель забезпечує зручний доступ до форми додавання пацієнтів без необхідності переходу на окрему сторінку. Це дозволяє швидко створювати нові записи без порушення загального робочого процесу. Чітка структура форми та мінімальна кількість полів забезпечують інтуїтивне заповнення даних навіть для некваліфікованих користувачів.

Сторінка "Прийоми" є першою вкладкою профілю пацієнта та надає користувачам доступ до детальної інформації про всі заплановані, завершені або скасовані прийоми конкретного пацієнта. Ця сторінка побудована з акцентом на зручну організацію даних та ефективний пошук необхідної інформації.

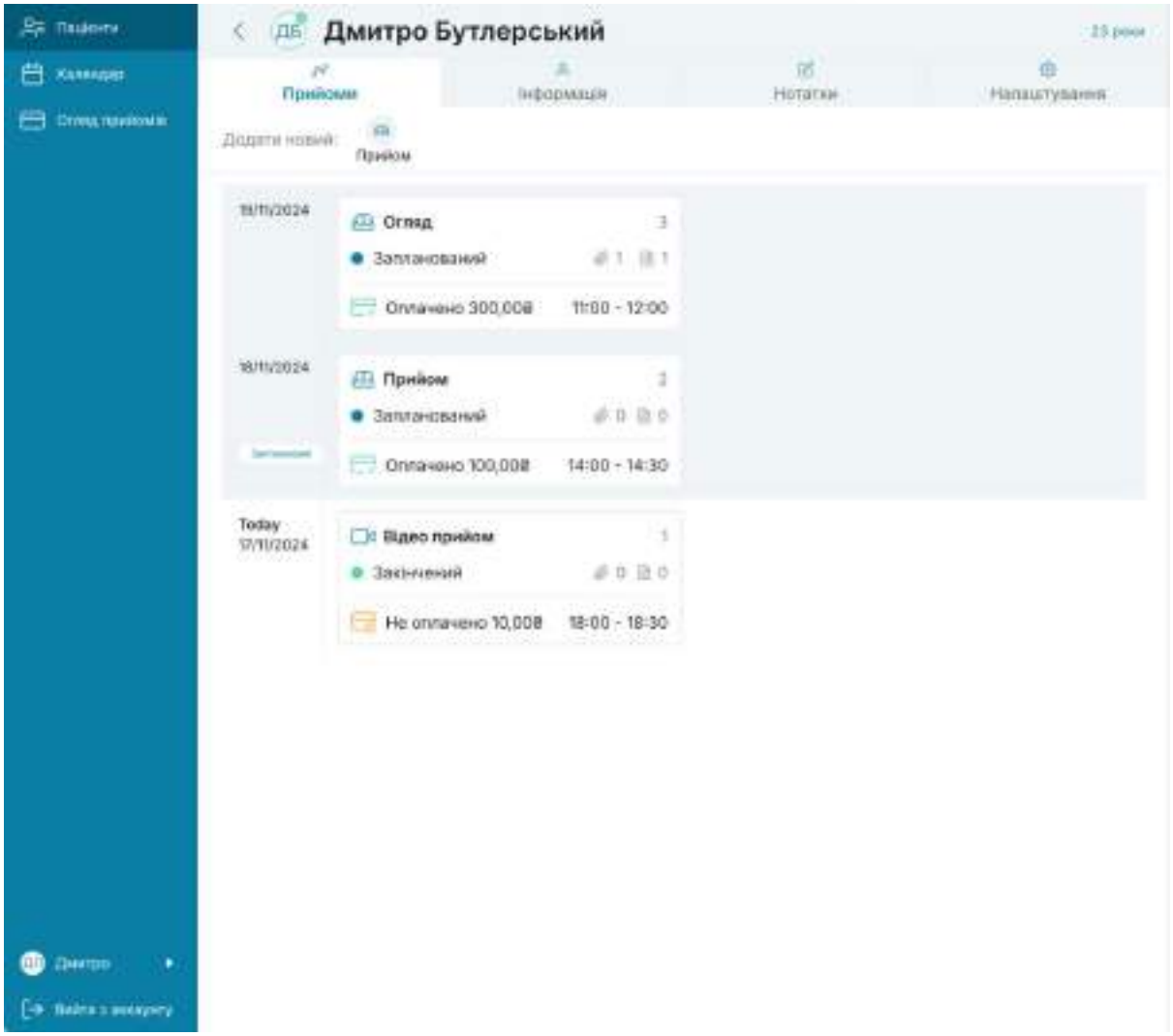

Рис. 22 - Сторінка «Прийоми»

У верхній частині сторінки розташовано ім'я та прізвище пацієнта (наприклад, "Дмитро Бутлерський") разом із віком (23 роки). Ця інформація дозволяє однозначно ідентифікувати пацієнта, особливо у випадках роботи з великою кількістю записів.

Під заголовком є чотири основні вкладки:

- Прийоми: відображає всі заплановані або завершені прийоми пацієнта.
- Інформація: містить загальні дані про пацієнта.
- Нотатки: дозволяє додавати нотатки щодо пацієнта та його лікування.
- Налаштування: забезпечує зміну параметрів профілю пацієнта.

У верхній частині секції "Прийоми" є кнопка "Додати новий", яка дозволяє створити новий запис прийому. Натискання на цю кнопку відкриває форму для заповнення інформації про прийом (тип, дата, час, оплата тощо).

Сторінка також містить хронологічний перелік прийомів пацієнта, згрупованих за датами. Кожен прийом представлено у вигляді компактної картки, що включає наступну інформацію:

- Назва прийому: наприклад, "Огляд", "Відео прийом", "Прийом".
- Статус: статуси прийому чітко позначені кольорами (наприклад, "Запланований" із синьою позначкою, "Закінчений" із зеленою).
- Оплата: інформація про суму, наприклад, "Оплачено 100,00₴" або "Не оплачено". Це важливо для відстеження фінансових операцій.
- Дата та час: чітко зазначено проміжки часу для кожного прийому (наприклад, 14:00 – 14:30).

Перелік прийомів розподілений за днями. Для зручності користувач може швидко орієнтуватися між записами завдяки хронологічній структурі. Наприклад, сьогоднішній день відмічено як "Сьогодні".

Після натискання на кнопку “додати прийом” відкривається бокова панель для створення нового прийому, яка має простий та інтуїтивно зрозумілий дизайн. Вона забезпечує швидке введення даних для створення запису.

У центральній частині панелі знаходить форма з таких полів:

- Тип прийому - користувач може обрати між фізичним (очним) прийомом або відео-прийомом, що дозволяє адаптувати записи залежно від формату консультації. Вибір здійснюється за допомогою радіо-кнопок, які зрозуміло відображають можливі варіанти.
- Дата та час прийому - інтерфейс дозволяє вказати дату прийому за допомогою інтегрованого календаря, що відкривається при

натисканні на відповідне поле. Для часу передбачено окремі поля, що дають змогу вибрати точний час початку та закінчення прийому.

- Оплата - поле для статусу оплати дає можливість зазначити, чи був прийом оплачений. Якщо оплата проведена, можна вказати суму у відповідному полі. Окремо передбачено вибір способу оплати: готівка або картка.

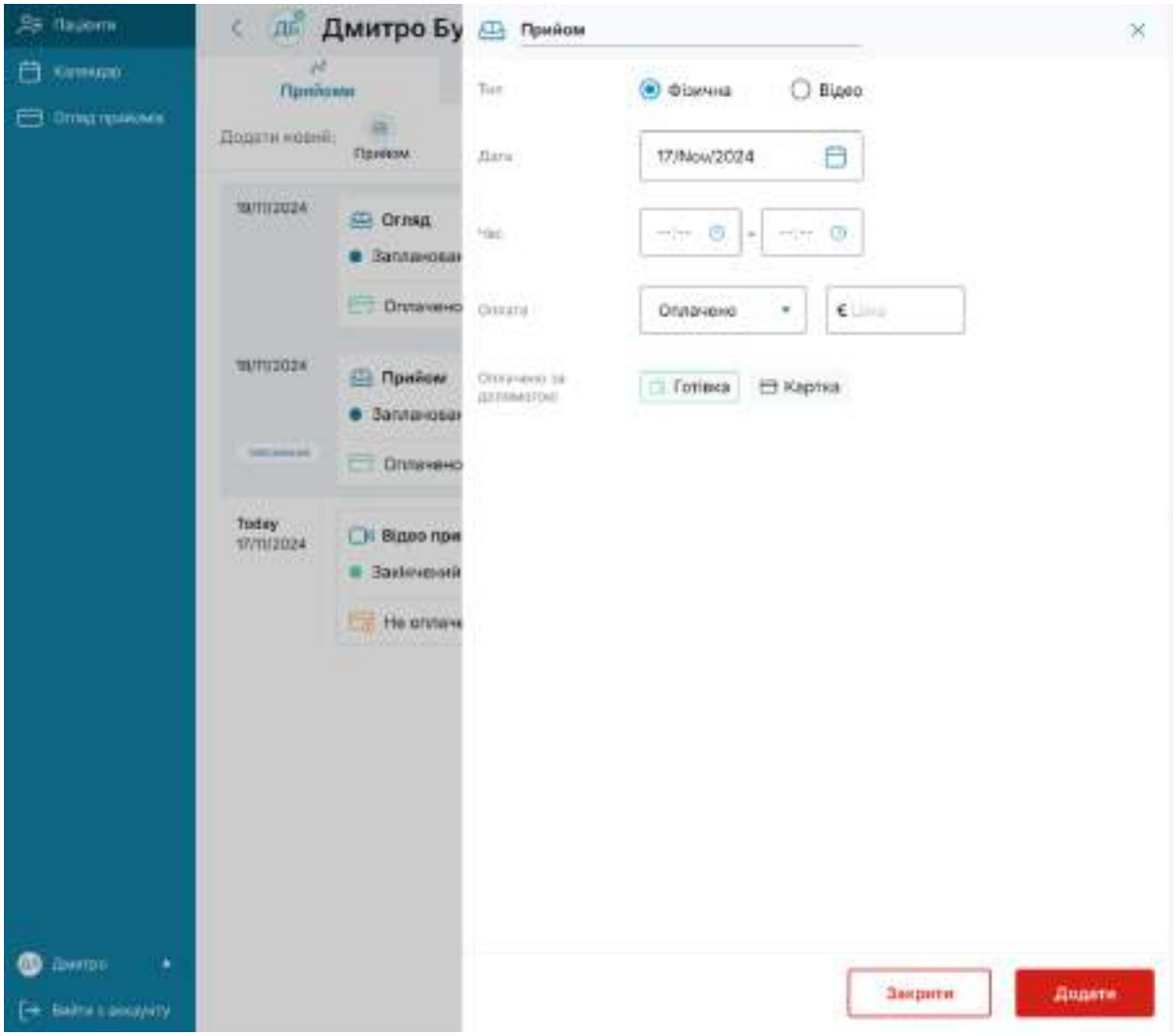

Рис. 23 - Бічна панель «Створення нового прийому»

Також користувач має змогу подивитись детальний опис створених прийомів. У бічній панелі з’являється інформація про статус прийому, а також всі введені раніше деталі: дата, час, спосіб та сума оплати. Також тут є нові кнопки: “Додати файл” та “Додати нотатку”, які відповідно дають змогу приєднати файл до прийому або залишити текстову нотатку.

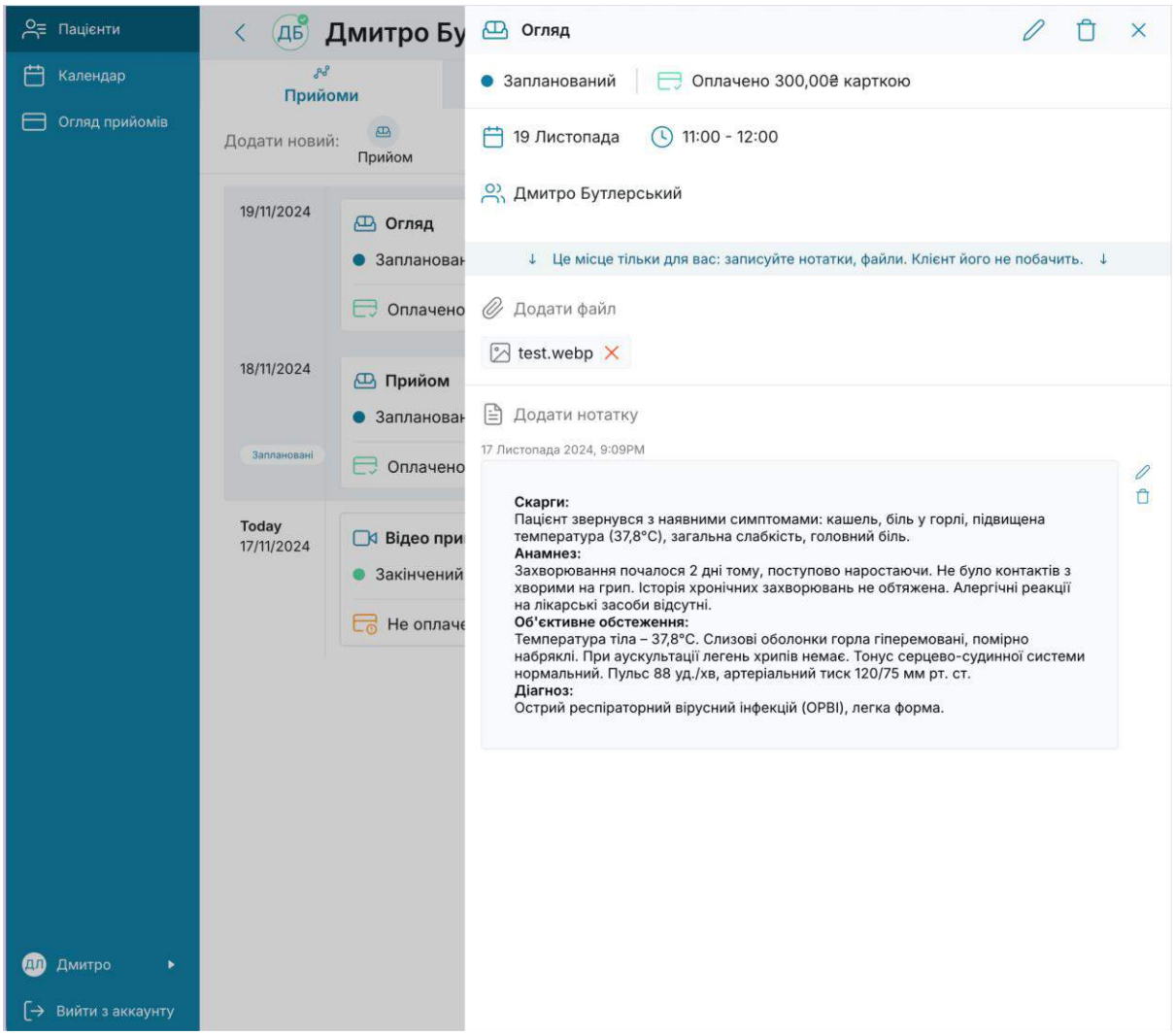

Рис. 24 - Бічна панель «Деталі прийому»

Сторінка "Інформація" в профілі пацієнта поділена на два основні підрозділи: "Особисті дані" та "Лікування". Вони забезпечують користувачів зручним інструментом для управління ключовою інформацією про пацієнтів.

Секція "Особисті дані" дозволяє зберігати основну інформацію про пацієнта. Вона поділена на кілька логічних груп:

1. Контактна інформація:
    - Поле для введення e-mail пацієнта, забезпечене іконкою конверта для наочності.
    - Поля для введення імені, прізвища та дати народження з чітким поділом на день, місяць та рік.
2. Гендерна ідентифікація:
    - Радіо-кнопки для вибору статі (чоловіча, жіноча, інша), що спрощує введення інформації.
3. Контактні дані:
    - Поле для номера телефону з інтегрованою іконкою телефону, яка вказує на тип інформації.
4. Адресна інформація:
    - Поля для введення міста та адреси пацієнта, доповнені іконкою для відображення геолокації.
5. Медична інформація:
    - Поля для введення імені контактної особи та номера телефону для надзвичайних ситуацій.

Секція "Лікування" орієнтована на збереження медичних даних, які недоступні для пацієнта, але важливі для лікаря.

Клінічна картина - поля для опису різних аспектів медичного стану:

- Опис клієнта та його перших контактів.
- Тип реєстрації проблеми.
- Поточний функціональний статус.
- Історія життя та важливі події.

Прогрес лікування - поле для плану лікування, яке дозволяє фіксувати рекомендації, частоту та час сеансів.

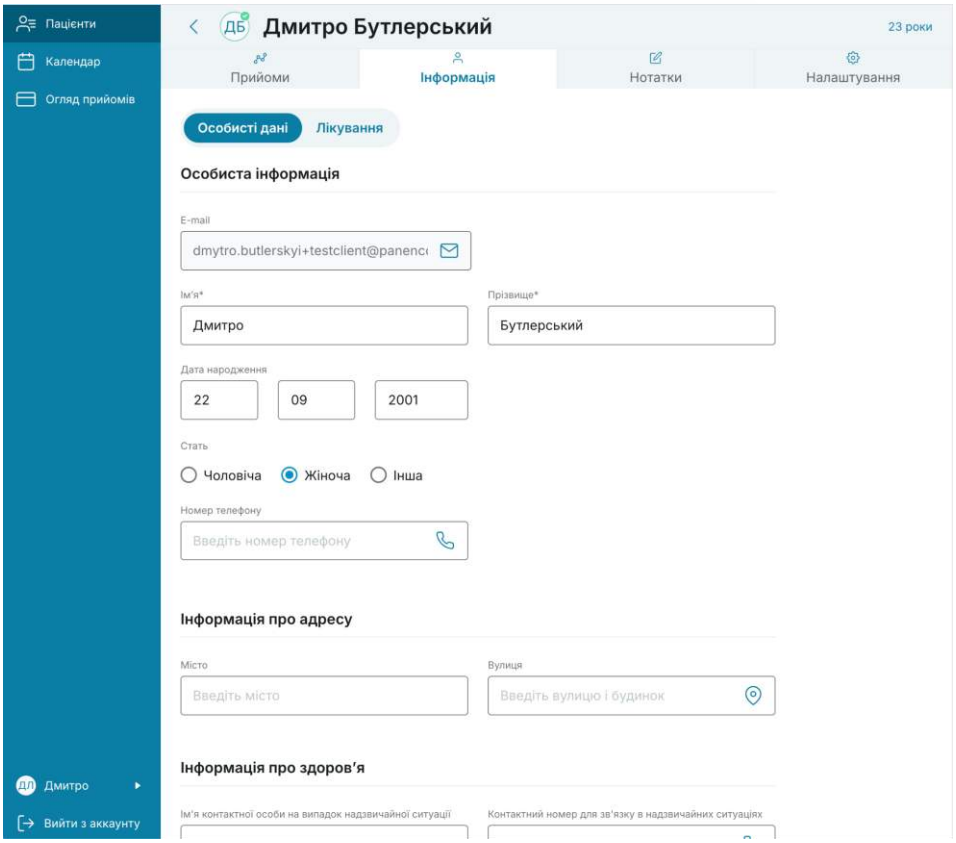

Рис. 25 - Секція «Особисті дані»

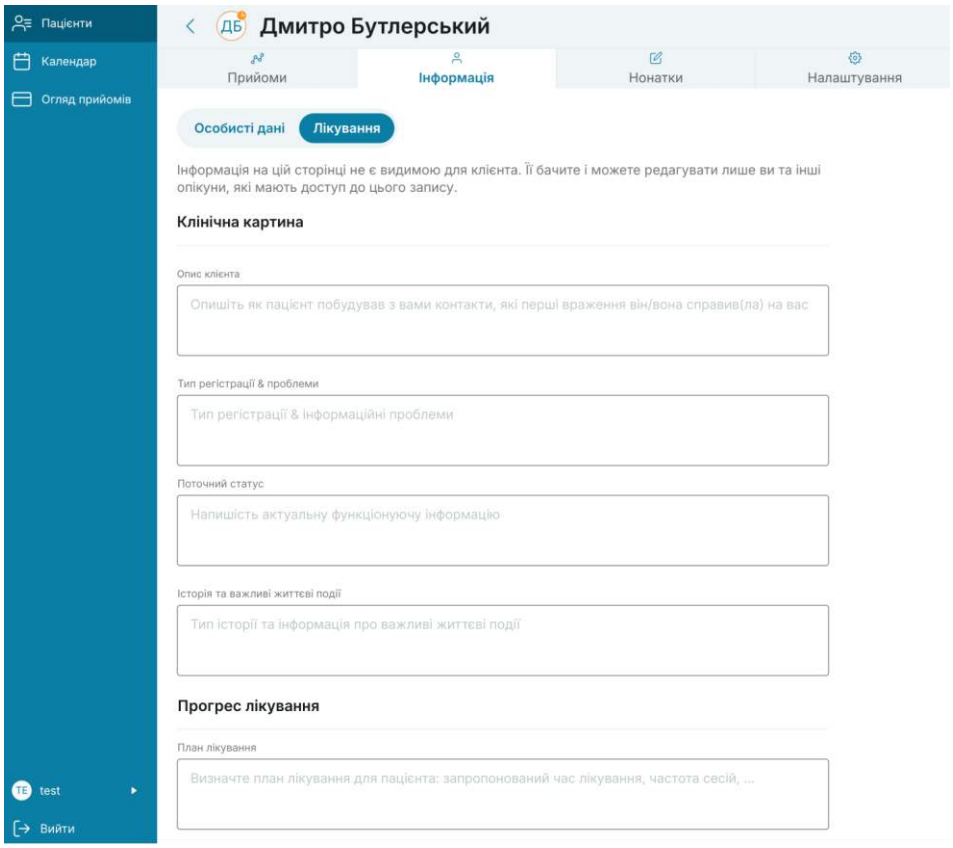

Рис. 26 - Секція «Лікування»

Екран "Календар" у веб-додатку "Електронний реєстр пацієнтів" призначений для зручного перегляду та управління записами на прийом пацієнтів.

- У верхній частині екрану є випадаюче меню для вибору вигляду календаря: місяць (за замовчуванням), тиждень або день. Це дозволяє гнучко змінювати формат відображення залежно від потреб користувача.
- Стрілки ліворуч та праворуч дозволяють переміщатися між періодами (наприклад, між місяцями в режимі місяця).

Сітка календаря:

- У режимі місяця календар відображає всі дні поточного місяця, починаючи з понеділка.
- Записи на прийоми відображаються у вигляді кольорових блоків, розташованих у відповідних днях. Кожен блок містить час прийому, ім'я пацієнта та іконки, які вказують на формат (наприклад, відео-консультації позначені значком камери).
- Над календарем розташована кнопка "Додати прийом", яка дозволяє створити новий запис безпосередньо з цього екрану.

Візуалізація записів:

- Записи мають чіткий вигляд з використанням піктограм для додаткової інформації. Наприклад, фізичні прийоми позначені одним типом іконок, а відео-консультації — іншим.

Екран виконаний у мінімалістичному стилі з чіткою організацією інформації. Сітка календаря є інтуїтивно зрозумілою, а використання іконок та кольорових акцентів полегшує швидке сприйняття інформації.

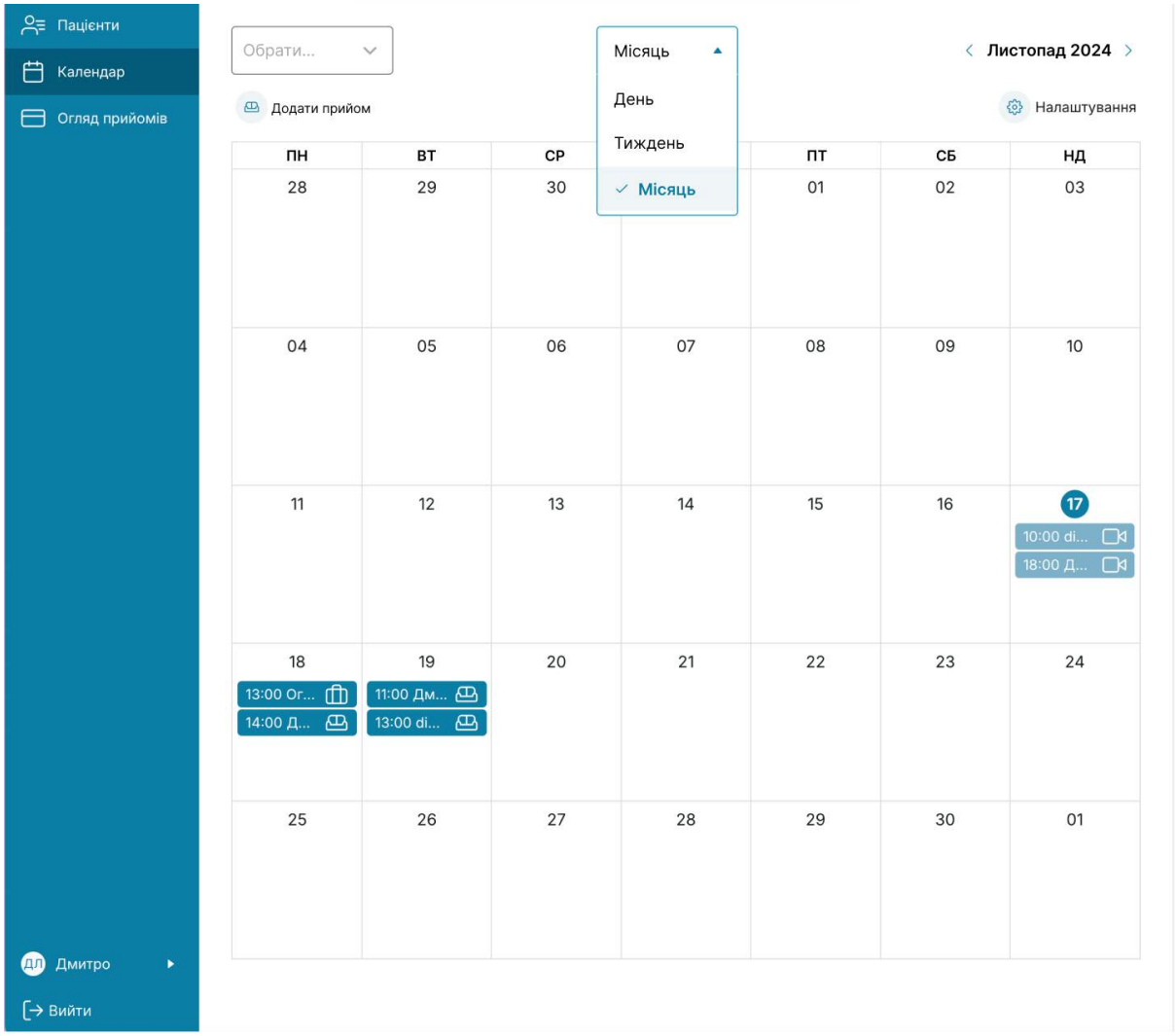

Рис. 27 - Сторінка «Календар»

Висновок до розділу 3

У розділі було досліджено програмну реалізацію веб-додатку "Електронний реєстр пацієнтів", яка включала дві ключові складові: базу даних та графічний інтерфейс. Було проаналізовано структуру бази даних, визначено основні таблиці, їх взаємозв’язки та оптимізацію для забезпечення безпеки й масштабованості. Під час аналізу графічного інтерфейсу були описані основні екрани з акцентом на зручність і відповідність принципам UX/UI дизайну. Було підбито підсумки щодо інтеграції технічних аспектів із вимогами зручності користування та встановлено, що фінальний продукт має відповідати сучасним стандартам безпеки, ефективності та інтуїтивності використання.

ВИСНОВКИ

У ході проведеного дослідження було досягнуто мету та виконано всі поставлені завдання, що підтверджує високий рівень наукової підготовки та відповідність вимогам Національної рамки кваліфікацій і стандартам вищої освіти. Теоретичне дослідження дозволило систематизувати сучасні підходи до цифровізації реєстрів пацієнтів, визначити основні переваги та виклики цього процесу, а також оцінити правові й етичні аспекти впровадження електронних медичних записів. Практичні результати включають розробку архітектури клієнт-серверного веб-додатку, проектування бази даних та реалізацію інтуїтивного графічного інтерфейсу, що забезпечує ефективність і зручність роботи з системою.

У першому розділі досліджено теоретичні основи цифровізації медичних записів, включаючи її переваги, виклики та перспективи. Було здійснено аналіз існуючих аналогів та підкреслено важливість правових і етичних аспектів у впровадженні цифрових інструментів. Результати розділу створили міцну основу для практичної частини дослідження.

Другий розділ присвячено аналізу та проєктуванню веб-додатку. Було обґрунтовано вибір клієнт-серверної архітектури, визначено переваги та недоліки альтернативних підходів, а також підібрано технології, що найкраще відповідають вимогам масштабованості та зручності у розробці. Це забезпечило необхідну базу для реалізації практичної частини роботи.

У третьому розділі здійснено програмну реалізацію веб-додатку. Було розроблено структуру бази даних, реалізовано інтуїтивно зрозумілий графічний інтерфейс для управління реєстром пацієнтів, а також інтегровано взаємодію між фронтендом і бекендом. Отриманий результат демонструє відповідність сучасним вимогам до програмних продуктів у медичній сфері.

Основними науковими результатами є обґрунтування вибору технологій і архітектурних підходів, які відповідають вимогам

масштабованості, безпеки та простоти підтримки. Практична цінність роботи полягає у створенні прототипу веб-додатку "Електронний реєстр пацієнтів", що може бути використаний як основа для впровадження в медичних закладах.

Перспективи подальшого використання отриманих результатів включають можливість адаптації розробленої системи до специфічних вимог медичних установ, інтеграції з телемедичними сервісами та розширення функціоналу для роботи з аналітикою пацієнтських даних. Рекомендовано використовувати результати дослідження для подальших розробок у сфері цифровізації медичних послуг, а також для вдосконалення підходів до захисту медичної інформації в умовах сучасних кіберзагроз

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Цифровізація - це поступове перетворення усіх державних послуг на зручні онлайн-сервіси [Електронний ресурс]. Режим доступу до ресурсу: https://www.rv.gov.ua/news/cifrovizaciya-ce-postupove-peretvorennya-usihderzhavnih-poslug-na-zruchni-onlajn-servisi

[2] Євдокимов В. В. Впровадження електронних медичних реєстрів: переваги та виклики. Український медичний журнал. 2019. 54 с.

[3] Epic System Corporation [Електронний ресурс]. Режим доступу до ресурсу: https://www.epic.com/

[4] Cerner Corporation [Електронний ресурс]. Режим доступу до ресурсу: https://www.oracle.com/health/

[5] Allscripts Healthcare Solutions [Електронний ресурс]. Режим доступу до ресурсу: https://veradigm.com/

[6] NextGen Healthcare Information Systems [Електронний ресурс]. Режим доступу до ресурсу: https://www.nextgen.com/

[7] Doxy.me [Електронний ресурс]. Режим доступу до ресурсу: https://doxy.me/en/

[8] Mindly [Електронний ресурс]. Режим доступу до ресурсу: https://app.mindlyspace.com/flow/2zs6zt29yx

[9] Панасюк Н. П. Етичні аспекти ведення електронних медичних реєстрів. Етика і медицина. 2020. 30 с.

[10] What is REST? [Електронний ресурс]. Режим доступу до ресурсу: https://www.codecademy.com/article/what-is-rest

[11] What Does a Front-End Developer Do? Complete Guide to the Front-End Developer Profession [Електронний ресурс]. Режим доступу до ресурсу: https://frontendmasters.com/guides/front-end-handbook/2018/what-is-a-FD.html

[12] React [Електронний ресурс]. Режим доступу до ресурсу: https://react.dev/

[13] Backend [Електронний ресурс]. Режим доступу до ресурсу: https://foxminded.ua/front-end-back-end-riznytsia/#:~:text=Backend

[14] Node.js [Електронний ресурс]. Режим доступу до ресурсу: https://nodejs.org/uk/docs

[15] Сучасний підручник з JavaScript [Електронний ресурс]. Режим доступу до ресурсу: https://uk.javascript.info/

[16] Каплун О. Програмна архітектура: принципи проектування. Львів: Видавництво "Львівська політехніка". 2018. 93 с.

[17] Кривенко М. Монолітна та мікросервісна архітектура: переваги та недоліки. Вісник Національного технічного університету "Харківський політехнічний інститут". 2019. 85 с.

[18] Ерік Райс. The Lean Startup. 2022. 320 с.

[19] Дьяков В. Мікросервіси: новий підхід до розробки програмного забезпечення. Журнал "Інформаційні технології". 2021. 45-50 с.

[20] gRPC [Електронний ресурс]. Режим доступу до ресурсу: https://grpc.io/

[21] What is HTTP? [Електронний ресурс]. Режим доступу до ресурсу: https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/

[22] DOU [Електронний ресурс]. Режим доступу до ресурсу: https://dou.ua/

[23] What is framework? [Електронний ресурс]. Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/framework

[24] What Is Cross Browser Testing And How To Perform It: A Complete Guide [Електронний ресурс]. Режим доступу до ресурсу: https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/

[25] What is the HTML DOM [Електронний ресурс]. Режим доступу до ресурсу: https://www.w3schools.com/whatis/whatis_htmldom.asp

[26] Angular [Електронний ресурс]. Режим доступу до ресурсу: https://angular.io/

[27] Vue [Електронний ресурс]. Режим доступу до ресурсу: https://vuejs.org/

[28] Ember [Електронний ресурс]. Режим доступу до ресурсу: https://emberjs.com/

[29] Visual Studio Code [Електронний ресурс]. Режим доступу до ресурсу: https://code.visualstudio.com/

[30] GIT [Електронний ресурс]. Режим доступу до ресурсу: https://git-scm.com/

[31] Чаріті Мейджорс. Лейн Кемпбел. Database Reliability Engineering: Designing and Operating Resilient Database Systems. 2018. 300 с.

[32] MongoDB [Електронний ресурс]. Режим доступу до ресурсу: https://www.mongodb.com/cloud/atlas/lp/try4?utm_content=controlhterms&utm_source=google&utm_campaign=search_gs_pl_evergreen_atlas_core_prosp-brand_gic-null_emea-ua_ps-all_desktop_eng_lead

[33] JSON [Електронний ресурс]. Режим доступу до ресурсу: https://www.json.org/json-en.html

[34] Алан Купер. Про інтерфейс. 2016. 433 с.

[35] Джош Сейден. Джефф Готельф. Lean UX. 2024. 206 с.

[36] What is reCaptcha [Електронний ресурс]. Режим доступу до ресурсу: https://support.google.com/recaptcha/answer/6080904?hl=en

[37] Бутлерський Д.С., Ніколюк П.К. Розробка веб-додатку для ведення електронного реєстру пацієнтів. Матеріали Третьої міжнародної науково-практичної конференції «Прикладні аспекти сучасних міждисциплінарних досліджень», Вінниця: ДонНУ імені Василя Стуса, 2024. Прийнято до друку.