МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
Навчально-науковий інститут кібернетики, національних технологій та інженерії
Кафедра комп'терних наук та прикладної математики

«До захисту допущена»
Зав. кафедри комп'ютерних наук та
прикладної математики д.т.н.,
професор Турбал Ю.В.
____ _____2025 р.

БАКАЛАВРСЬКА РОБОТА

на тему:

РОЗРОБКА САЙТУ ЦЕНТРА РЕАБІЛІТАЦІЇ ПАЦІЄНТІВ НА ОСНОВІ ФРЕЙМВОРКУ DJANGO(PYTHON)

Виконав: Лустюк Назар Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Група ІПЗ-41інт

Керівник: ст. викладач кафедри Харів Наталія Олексіївна _____
(науковий ступінь, вчене звання, посада, прізвище та ініціали) (підпис)

Рівне-2025

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 58 сторінок, 2 рисунка, 7 таблиць, 30 джерел.

Мета роботи – розробити комплексний веб-додаток для автоматизації діяльності центра реабілітації пацієнтів на основі фреймворку Django, що забезпечить ефективне управління процесами реабілітації, моніторинг прогресу пацієнтів.

Об’єкт дослідження – процеси організації та управління реабілітаційною діяльністю в медичних закладах, включаючи планування програм відновлення, моніторинг виконання вправ.

Предмет дослідження – методи та технології розробки веб-орієнтованих інформаційних систем для автоматизації процесів реабілітації на основі фреймворку Django, включаючи архітектурні рішення, моделі даних, алгоритми обробки медичної інформації та підходи до забезпечення безпеки.

Методи вивчення – аналітичні методи для вивчення джерел та існуючих рішень, методи системного аналізу для декомпозиції процесів та моделювання, методи проектування (об'єктно-орієнтоване, UML, бази даних) для створення архітектури, методи розробки програмного забезпечення (ітеративна розробка, фреймворки, шаблони) для реалізації, а також методи тестування (модульне, інтеграційне, безпеки, юзабіліті) для забезпечення якості системи.

Ключові слова: Django, Python, веб-розробка, реабілітаційний центр, веб-сайт, база даних, функціональне тестування, ERD.

ВСТУП

Сучасне суспільство перебуває у стані безперервної трансформації під впливом технологічних інновацій, соціальних змін та економічних викликів. Медицина, як одна з ключових сфер життєдіяльності, не залишається осторонь такої еволюції. Згідно з офіційними даними Всесвітньої організації охорони здоров'я (ВОЗ), понад 2.4 мільярда людей у світі потребують реабілітаційних послуг, що свідчить про величезний масштаб та важливість цієї галузі [1]. Нещодавня глобальна пандемія COVID-19 стала каталізатором для глибинних змін у медичній сфері, значно прискоривши її цифровізацію [4]. Впровадження телемедицини, дистанційного моніторингу та онлайн-платформ для консультацій стало невід'ємною частиною сучасної практики, доводячи свою ефективність та доступність, особливо в умовах обмежень пересування та соціального дистанціювання [4].

З початком повномасштабного вторгнення Російської Федерації в Україну у 2022 році, потреба в реабілітаційних послугах набула нового, надзвичайно критичного розміру. Величезна кількість військових та цивільних осіб, постраждалих від бойових дій, отримали фізичні та психологічні травми, що потребують тривалої та спеціалізованої реабілітації. Особливо актуальною стала потреба в ресурсах, що дозволятимуть ефективно працювати з пацієнтами, які через стан здоров'я, місцевість проживання або безпекові обставини не мають можливості фізично прибути до традиційних реабілітаційних закладів, часто розташованих у віддалених або небезпечних регіонах.

Реабілітація – це складний, багатогранний та тривалий процес [1], який вимагає постійної, тісної взаємодії між медичними фахівцями (лікарями, фізіотерапевтами, психологами, ерготерапевтами) та пацієнтами. Ключовими елементами є гнучкість планування індивідуальних програм реабілітації, регулярне корегування, моніторинг прогресу пацієнтів та об’єктивний контроль результатів [22]. Запровадження сучасних онлайн-платформ дозволяє значно покращити ці

аспекти, підвищуючи якість, доступність та ефективність надання реабілітаційних послуг. Вони надають можливість оптимізувати використання ресурсів клініки, зменшити адміністративне навантаження на персонал, зробити догляд більш персоналізованим та адаптованим до індивідуальних потреб пацієнта, а також розширити географію надання послуг для тих, хто не може звернутися віч-на-віч [3].

У даній бакалаврській роботі розглядається процес розробки веб-сайту для центру реабілітації пацієнтів. Як основу для створення веб застосунку обрано фреймворк Django, написаний на мові програмування Python [7, 8]. Вибір Django був обґрунтований його унікальними перевагами: високою гнучкістю, що дозволяє адаптувати платформу до специфічних потреб центру; вбудованою системою безпеки [16], яка критично важлива для захисту чутливих даних пацієнтів; розширюваністю, що забезпечує можливість додавання нових функцій у майбутньому; та зручністю розробки як клієнтської (фронтенд), так і серверної (бекенд) частини веб застосунку.

Такі сучасні системи здатні значно спростити адміністративні процедури [3] (наприклад, запис на прийом, управління графіками, звітування), автоматизувати рутинні завдання, покращити обмін інформацією між медичним персоналом та пацієнтами (наприклад, надання результатів досліджень, рекомендацій, відстеження прогресу). Крім того, ці платформи можуть розширити можливості надання послуг, включаючи функціонал телемедицини [4], віртуальних консультацій, дистанційного моніторингу стану пацієнта. Перехід до цифрових рішень дозволяє реабілітаційним центрам не лише підвищити свою операційну ефективність, але й забезпечити безперервність та доступність надання допомоги, що безпосередньо впливає на результати лікування пацієнтів та стабільність функціонування закладу, особливо в складних умовах. Таким чином, розробка веб-сайту для центру реабілітації є не просто технічним завданням, а стратегічною відповіддю на еволюційні вимоги сучасної медичної галузі та очікування пацієнтів, сприяючи трансформації центру з

суто фізичної установи у гібридну модель, здатну надавати постійну, доступну та персоналізовану реабілітаційну допомогу.

РОЗДІЛ 1. Теоретико-методологічні аспекти розробки веб-сайту для реабілітаційного центру

1.1 Системний аналіз предметної області: реабілітаційні центри та їх потреби.

Для ефективної розробки веб-системи для реабілітаційного центру необхідно глибоко зрозуміти його функціонування, включаючи як клінічні аспекти реабілітаційного процесу, так і адміністративні робочі процеси [1]. Це дозволяє створити рішення, яке відповідатиме реальним потребам усіх зацікавлених сторін.

Процес реабілітації пацієнтів

Реабілітація – це комплексний та динамічний підхід, спрямований на покращення, підтримання та відновлення фізичного здоров'я людини, особливо після хвороб, операцій або травм, що обмежили її рухливість [1]. Цей процес є життєво важливим для підвищення якості життя, незалежно від того, чи йдеться про управління хронічним захворюванням, відновлення після спортивної травми або повернення рухливості після операції.

Типовий шлях пацієнта в реабілітаційному центрі включає кілька ключових етапів [1]. Перший етап – це прийом та оцінка, що передбачає ретельне вивчення лікарями історії хвороби пацієнта, його фізичного стану та функціональних обмежень. Використовуються різноманітні процедури оцінки, такі як функціональні оцінки, оцінки сили та діапазону рухів, для виявлення конкретних проблем та визначення обсягу необхідної реабілітації [1]. Цей етап також враховує соціальні та психологічні аспекти, що можуть вплинути на одужання пацієнта. Важливо, щоб пацієнти співпрацювали з медичними працівниками для встановлення конкретних цілей та розробки індивідуального плану лікування, що відповідає їхнім унікальним потребам та обставинам.

Наступним етапом є постановка цілей. На основі результатів оцінки пацієнти та лікарі спільно розробляють чіткі, вимірювані, обмежені в часі та значущі цілі [1]. Ці цілі надають чіткий напрямок реабілітаційному процесу, охоплюючи такі сфери,

як покращення сили, гнучкості, зменшення болю та відновлення функціональної незалежності. Цілі є гнучкими та можуть коригуватися в міру прогресу пацієнта або виникнення нових викликів. Активна участь пацієнтів у постановці цілей підвищує їхню мотивацію та відчуття задоволення, спонукаючи їх до активної участі у власному одужанні [1].

Далі йде планування лікування та терапевтичні втручання. Це адаптивний та цілеспрямований процес, де медичні фахівці співпрацюють для створення комплексного плану, що окреслює необхідні методи лікування та операції для одужання пацієнта [1]. Цей план керує впровадженням терапевтичних методів, включаючи лікувальні вправи (вправи на діапазон рухів, зміцнюючі вправи, кардіотренування, вправи на гнучкість), мануальну терапію (масаж, мобілізація суглобів, мобілізація м'яких тканин, мануальне розтягування) та модальності (теплова та холодова терапія, електрична стимуляція, ультразвукова терапія, тракція). Він враховує унікальні обставини пацієнта, такі як спосіб життя, вподобання та історія хвороби, забезпечуючи комплексний підхід до реабілітації. Регулярна комунікація між пацієнтом та реабілітаційною командою є життєво важливою для того, щоб план лікування відповідав цілям пацієнта та його потребам, що розвиваються.

Моніторинг та оцінка прогресу є ще одним важливим етапом [1, 22]. Регулярний моніторинг та перегляд прогресу гарантують, що терапія залишається специфічною для потреб пацієнта та його цілей, що розвиваються. Часті оцінки відстежують об'єктивні параметри, такі як збільшення функціональних можливостей, сили та діапазону рухів. Ця безперервна оцінка дозволяє медичним працівникам динамічно змінювати методи терапії у відповідь на перешкоди або невдачі [1]. Вона також включає суб'єктивний досвід пацієнта, такий як інтенсивність болю та загальний стан здоров'я. Пацієнти потребують постійної оцінки та відкритої комунікації з реабілітаційною командою, щоб визнавати успіхи, виявляти сфери для подальшого зростання та досягати оптимальних результатів.

Психосоціальна підтримка є важливим етапом через складну взаємодію між психічним/емоційним здоров'ям та фізичним одужанням [1]. Фахівці з реабілітації включають консультування, підтримку однолітків та медитацію, щоб допомогти пацієнтам керувати болем, адаптуватися до змін у способі життя та справлятися з емоційними труднощами. Ця підтримка сприяє оптимізму та зменшує тривогу та відчай.

Нарешті, відбувається перехід до самостійності [1]. У міру прогресу пацієнта, мета досягнення незалежності у повсякденних справах стає першочерговою. Це може включати вирішення конкретних проблем на роботі чи вдома та симуляцію реальних сценаріїв. Реабілітаційна команда надає пацієнтам інструменти та стратегії для довгострокового успіху в їхньому звичайному житті.

Аналіз процесу реабілітації показує, що шлях пацієнта є високоітеративним та адаптивним, вимагаючи безперервної оцінки, коригування цілей та модифікації втручань. Це означає, що будь-яке програмне забезпечення, яке підтримує цей процес, має бути достатньо гнучким, щоб враховувати динамічні плани лікування та відстеження прогресу в реальному часі, а не просто статичне ведення записів. Це вимагає архітектури системи, яка надає пріоритет модульності, можливостям оновлення даних та надійним функціям комунікації для справжнього задоволення основних клінічних потреб.

Адміністративні робочі процеси

Ефективне управління реабілітаційною практикою є вирішальним для успішної клініки, що охоплює безперебійні операції, фінансову стабільність та якісний догляд за пацієнтами. Ключові аспекти управління реабілітаційною практикою включають планування прийомів пацієнтів, виставлення рахунків та страхові претензії, відповідність вимогам та документування, управління персоналом та комунікацію з пацієнтами [3].

Основні адміністративні функції включають планування прийомів пацієнтів, що оптимізує робочі процеси для зменшення часу очікування пацієнтів та

максимізації доступності постачальників послуг [3]. Ефективне програмне забезпечення для планування може впорядкувати бронювання прийомів та допомогти зменшити кількість неявок. Важливо збалансувати щоденні розклади, щоб виділити достатньо часу як для догляду за пацієнтами, так і для адміністративних завдань. Наступна функція – виставлення рахунків та страхові претензії [3]. Цей робочий процес забезпечує своєчасне відшкодування за надані послуги. Регулярний перегляд робочих процесів виставлення рахунків є важливим для мінімізації помилок та прискорення відшкодувань. Перевірка страховки заздалегідь та зменшення відмов у претензіях є вирішальними для ефективного управління доходами.

Відповідність вимогам та документування є життєво важливим для дотримання законодавства та підтримання довіри [3, 16]. Системи електронних медичних карт (EHR) використовуються для безпечного управління записами пацієнтів [24]. Дотримання оновлених правил, регулярне навчання персоналу та використання EHR для точного документування є ключовими для уникнення аудитів та штрафів. Захист даних пацієнтів та дотримання HIPAA та галузевих правил запобігають порушенням безпеки.

Управління персоналом забезпечує належне ліцензування, навчання та збалансоване навантаження для персоналу [3]. Надання повноважень персоналу шляхом розподілу завдань на основі їхніх сильних сторін та заохочення самостійного вирішення проблем може підвищити ефективність. Використання інструментів управління завданнями допомагає забезпечити підзвітність без мікроменеджменту.

Комунікація з пацієнтами є вирішальним аспектом управління пацієнтами [3]. Використання цілодобових служб відповіді для реабілітації може покращити залученість пацієнтів та зменшити кількість пропущених прийомів. Медична служба відповіді може забезпечити постійну доступність для дзвінків пацієнтів та

планування, зменшити адміністративне навантаження та підвищити задоволеність пацієнтів завдяки професійним та своєчасним відповідям.

Ефективне управління цими робочими процесами дозволяє реабілітаційним центрам покращити результати лікування пацієнтів, підвищити продуктивність персоналу та зміцнити фінансову стабільність.

1.2 Огляд існуючих платформ реабілітації (Physitrack, RehabGuru, MedBridge)

Для розробки ефективної веб-системи для реабілітаційного центру необхідно проаналізувати провідні платформи, що вже існують на ринку [4, 5, 6]. Це дозволить виявити найкращі практики, ідентифікувати ключові функціональні можливості та потенційні прогалини, які може заповнити пропонована система. Розглянемо три провідні платформи: Physitrack, RehabGuru та MedBridge.

Physitrack

Physitrack є провідною платформою, що спеціалізується на домашніх програмах вправ та віддаленому моніторингу пацієнтів [4]. Її основні функції включають безпечну телемедицину, що забезпечує захищені відеодзвінки, обмін повідомленнями та моніторинг стану пацієнтів. Платформа використовує технології, що забезпечують шифрування з'єднання [17], що є критично важливим для конфіденційності медичних даних. Також Physitrack має бібліотеку вправ та призначення, яка містить понад 15 000 повністю озвучених, перевірених відео вправ, революційний 3D-контент, освітні відео та PDF-матеріали [4]. Фахівці можуть завантажувати власні відео вправ та створювати індивідуальні шаблони програм.

Платформа надає можливості моніторингу результатів [4], включаючи автоматизований збір даних про результати лікування, своєчасне розсилання опитувань та показників результатів, а також збір даних у власному сховищі. Це дозволяє відстежувати прогрес пацієнтів та приймати рішення на основі даних [22]. Для залучення клієнтів Physitrack включає власні брендовані додатки (PhysiApp),

нагадування в додатку для підтримки мотивації пацієнтів, а також функцію миттєвих повідомлень для зв'язку в реальному часі [4]. Крім того, платформа має інтеграції з існуючими системами управління практикою (PMS), що дозволяє відкривати картку клієнта безпосередньо з PMS та автоматично копіювати призначені програми вправ, заощаджуючи час та зберігаючи цілісність даних [4].

RehabGuru

RehabGuru пропонує комплексний набір функцій для управління практикою, зосереджуючись на спрощенні робочих процесів [5]. Основні можливості включають велику бібліотеку вправ, що налічує понад 6 000 вправ та 280+ готових шаблонів, що прискорює робочий процес [5]. Користувачі можуть створювати власні вправи з HD-відео та зображеннями. Платформа забезпечує телемедицину через безпечні онлайн-консультації в один клік [5], які повністю базуються на браузері. Підтримуються як індивідуальні, так і групові відеодзвінки, що робить її ідеальною для групових занять. Відеодзвінки можуть бути брендовані кольорами та логотипом клініки.

RehabGuru також надає функції управління щоденником та онлайн-бронювання [5], що дозволяє створювати та керувати записами, автоматично генерувати телемедичні прийоми та приймати онлайн-платежі. Для форм пацієнтів та нотаток передбачено гнучкий конструктор форм, що дозволяє створювати індивідуальні форми, а також анотувати діаграми тіла [5]. Забезпечується безпечне завантаження та зберігання файлів пацієнтів. Моніторинг включає віддалене відстеження дотримання пацієнтами рекомендацій, результатів, рівня болю, самопочуття в реальному часі, що дозволяє раннє втручання та прийняття рішень на основі даних.

MedBridge

MedBridge є багатофункціональною платформою, що надає рішення як для клініцистів, так і для пацієнтів, з акцентом на безперервну освіту та цифрові шляхи

догляду [6]. Її ключові особливості включають безперервну освіту для клініцистів, що пропонує понад 2 700 акредитованих курсів, вебінари та програми спеціалізації для понад 15 дисциплін [6]. Це дозволяє клініцистам розширювати свою експертизу та підтримувати ліцензії. Платформа надає домашні програми вправ (HEP), які є індивідуальними програмами, що можуть бути доставлені пацієнтам у друкованому вигляді, через онлайн-портал або мобільний додаток [5]. Це сприяє залученню пацієнтів між візитами. MedBridge також пропонує цифрові шляхи догляду, що включають цифрові шляхи MSK (опорно-рухового апарату), які є незалежними та гібридними програмами догляду з прогресивними оцінками, освітою та вправами для управління пацієнтами з низькою та помірною гостротою [6]. Для віддаленого терапевтичного моніторингу (RTM) передбачені інструменти для моніторингу та звітності пацієнтів, що допомагають постачальникам послуг легко реєструвати пацієнтів, звітувати, документувати та виставляти рахунки за RTM. Також є функції управління компетенціями та навчання відповідності, що включають цифрові контрольні списки навичок, навчальні відео, оцінку наставників та інформаційні панелі даних.

Порівняльний аналіз

Усі три платформи демонструють спільні функціональні можливості, такі як призначення вправ, телемедицина [4, 5, 6] та засоби залучення пацієнтів. Однак, кожна з них має свої унікальні переваги. Physitrack виділяється якістю відеоконтенту та акцентом на домашніх програмах [5]. RehabGuru пропонує комплексне рішення для управління бізнесом, включаючи бронювання та виставлення рахунків. MedBridge, у свою чергу, поєднує освіту для клініцистів з розширеними цифровими шляхами догляду за пацієнтами [6].

Успіх провідних реабілітаційних платформ залежить від їхньої здатності створювати безперебійний, інтегрований цифровий досвід, який виходить за межі стін клініки, охоплюючи віддалений моніторинг, навчання пацієнтів та безперервну взаємодію. Це свідчить про те, що простого «веб-сайту» недостатньо; потрібна

«цифрова платформа для догляду». Платформи, такі як Physitrack, RehabGuru та MedBridge, пропонують комплексні функції, які підтримують весь шлях пацієнта віддалено та безперервно, прагнучи підтримувати пацієнтів «на шляху та мотивованими» та надавати «індивідуальні освітні рішення». Це означає, що запропонований веб-сайт має бути розроблений як комплексна цифрова екосистема, яка розширює можливості як пацієнтів, так і практикуючих лікарів, сприяючи віддаленій взаємодії, персоналізованому наданню допомоги та безперервному моніторингу. Таким чином, основні функціональні вимоги повинні надавати пріоритет функціям, які забезпечують постійний зв'язок між пацієнтом та постачальником послуг та самостійне управління, таким як безпечний обмін повідомленнями, персоналізовані програми вправ та реєстрація прогресу, а не просто статичний вміст або одноразові взаємодії.

Нижче наведено порівняльну характеристику цих платформ у таблиці 1.1.

Таблиця 1.1: Порівняльна характеристика існуючих платформ для реабілітаційних центрів

Характеристика	**Physitrack**	**RehabGuru**	**MedBridge**
Розмір бібліотеки вправ	15 000+ відео, 3D контент, власні шаблони	6 000+ вправ, 280+ готових шаблонів, власні вправи	Включає в HEP, фокус на освіті клініцистів
Телемедицина	Безпечні відеодзвінки, повідомлення, моніторинг, потокове відео вправ	Одноклікові, браузерні відеодзвінки (індивідуальні/групові), брендування	Цифрові шляхи MSK, RTM, відеодзвінки

Форми пацієнтів	Вхідні форми, опитування результатів	Гнучкий конструктор форм, анотації діаграм тіла, безпечне зберігання	Цифрові контрольні списки навичок, оцінки
Виставлення рахунків/платежі	Інтеграція з PMS	Вбудована система виставлення рахунків, онлайн-платежі, депозити	RTM інструменти для виставлення рахунків
Залучення пацієнтів	Брендовані додатки, нагадування, миттєві повідомлення	Мобільні додатки, моніторинг дотримання, зворотний зв'язок	HEP, цифрові шляхи, освітні відео
Відстеження прогресу	Моніторинг симптомів та активності, збір даних про результати	Віддалене відстеження дотримання, болю, самопочуття, RPE	RTM, оцінки, інформаційні панелі даних
Інтеграції	З системами управління практикою (PMS)	Google, Microsoft (Google Meet, MS Teams), синхронізація щоденника	З системами управління практикою (PMS)
Безпека/відповідність	Безпечні, зашифровані відеодзвінки	Безпечна, приватна, HD інфраструктура, шифрування	HIPAA-сумісність, курси відповідності
Цільова аудиторія	Приватні практики, лікарні,	Індивідуальні практики, клініки, NHS, MO Великої Британії	Індивідуальні клініцисти, приватні

	спортивні команди		практики, лікарні

1.3 Обґрунтування вибору технічних засобів розробки продукту

Вибір засобів розробки є критично важливим для успішної реалізації веб-додатку, особливо в такій чутливій сфері, як охорона здоров'я [16]. Для розробки сайту центру реабілітації пацієнтів було обрано комбінацію Django (Python), Bootstrap та PostgreSQL, що обґрунтовується їхніми перевагами та здатністю відповідати специфічним вимогам проекту.

Django (Python Framework)

Django є високорівневим веб-фреймворком на Python, що заохочує швидку розробку та чистий, прагматичний дизайн. Переваги Django включають швидку розробку завдяки концепції "Batteries Included" [7], що означає наявність багатьох вбудованих функцій, таких як адміністративний інтерфейс, ORM, система автентифікації, управління сесіями та обробка форм. Це значно прискорює процес розробки, що є вирішальним для проектів з обмеженим терміном, таких як бакалаврська робота, дозволяючи розробникам зосередитися на унікальних функціях, а не на створенні базових компонентів з нуля [29]. Безпека є першочерговим пріоритетом для Django [8, 16], який пропонує вбудовані засоби захисту від поширених вразливостей, таких як міжсайтовий скриптинг (XSS), міжсайтова підробка запитів (CSRF) та SQL-ін'єкції. Він також автоматично шифрує конфіденційні дані, такі як паролі, під час передачі [16]. Це є надзвичайно важливим для медичного додатку, який обробляє чутливі дані пацієнтів, забезпечуючи відповідність регуляторним вимогам, таким як HIPAA. Фреймворк також відзначається масштабованістю [7], здатною обробляти великий обсяг трафіку та значні обсяги даних, що робить його придатним для зростаючих додатків. Його

використання такими гігантами, як Instagram та Bitbucket, свідчить про його здатність до ефективного масштабування. Екосистема Python, на якій базується Django, надає універсальність та багаті бібліотеки для аналізу даних, машинного навчання та штучного інтелекту [23] (наприклад, Pandas, NumPy, TensorFlow). Ці можливості є дуже актуальними для медичних додатків, дозволяючи впроваджувати прогнозну діагностику, аналіз даних пацієнтів та автоматизацію процесів, що дозволяє системі не просто автоматизувати, але й надавати інтелектуальні можливості. Django також заохочує принципи DRY (Don't Repeat Yourself) та KISS (Keep It Simple, Stupid), що сприяє повторному використанню коду та уникненню надмірності, призводячи до більш чистого та легкого в підтримці коду, спрощуючи розробку та зменшуючи кількість помилок. Нарешті, велика, активна та професійна спільнота надає обширну документацію та швидке виправлення помилок, забезпечуючи легкість у пошуку рішень та постійне вдосконалення фреймворку.

Щодо недоліків, Django може здатися монолітним порівняно з мікрофреймворками, такими як Flask, і може бути надмірним для дуже малих проектів. Однак для багатофункціональної реабілітаційної платформи його концепція "батарейок у комплекті" є перевагою, оскільки вона надає багато необхідних компонентів. Також, він має крутішу криву навчання порівняно з простішими фреймворками, оскільки його "думку" природа вимагає розуміння підходу Django до вирішення завдань. Однак це призводить до створення послідовного та легкого в підтримці коду в довгостроковій перспективі.

Bootstrap (Front-end Framework)

Bootstrap є найпопулярнішим CSS-фреймворком, призначеним для розробки адаптивних та мобільних веб-сайтів [9]. Переваги Bootstrap включають адаптивний дизайн з підходом "mobile-first", що гарантує автоматичне пристосування веб-сайту до різних розмірів екранів (настільні комп'ютери, планшети, смартфони), забезпечуючи безперебійний користувацький досвід на будь-якому пристрої, що є

критично важливим для доступності пацієнтів. Використання підходу "mobile-first" безпосередньо покращує залученість пацієнтів та доступність.

Простота використання та швидке прототипування досягається завдяки готовим компонентам (кнопки, форми, навігаційні панелі, модальні вікна) та шаблонам, що значно прискорює розробку інтерфейсу користувача та дозволяє швидко створювати візуально привабливі та функціональні макети. Кросбраузерна сумісність забезпечується вбудованими стилями та нормалізацією для послідовного вигляду веб-сайтів у всіх основних веб-браузерах [9], що зменшує час на налагодження та покращує користувацький досвід на різних платформах. Bootstrap також надає можливості настроюваності, дозволяючи адаптувати компоненти до конкретних потреб проекту за допомогою змінних Sass та власного CSS, що дає можливість створити унікальний брендовий стиль, зберігаючи при цьому переваги фреймворку. Нарешті, велика спільнота та вичерпна документація спрощують навчання та вирішення проблем.

Щодо недоліків, уніфікованість або відсутність оригінальності може бути проблемою, оскільки стилі за замовчуванням Bootstrap можуть призвести до того, що веб-сайти виглядатимуть дуже схожими. Це буде вирішено шляхом налаштування та додавання унікальних елементів дизайну, щоб відрізнити веб-сайт центру. Також, може виникнути роздутість або проблеми з продуктивністю, оскільки фреймворк може включати невикористаний CSS/JS, що збільшує час завантаження сторінки. Це буде пом'якшено шляхом вибіркового завантаження компонентів та оптимізації ресурсів для зменшення навантаження. Інтенсивне використання попередньо визначених класів може призвести до захараченого HTML, що є компромісом для швидкої розробки та адаптивності, але може бути керовано за допомогою належних практик кодування.

PostgreSQL (Система управління базами даних)

PostgreSQL є потужною, відкритою об'єктно-реляційною системою управління базами даних [10], відомою своєю надійністю, цілісністю даних та

надійним набором функцій. Переваги PostgreSQL включають надійність та цілісність даних завдяки ACID-сумісності [11, 19], що забезпечує надійну обробку транзакцій бази даних, критично важливу для чутливих даних пацієнтів. Це гарантує, що дані завжди залишаються послідовними та точними, навіть у разі збоїв.

Масштабованість досягається завдяки функціям, таким як репліки для читання та пули з'єднань [10], що робить її ідеальною для обробки великого трафіку та користувачів, дозволяючи плавно масштабуватися в міру зростання потреб додатка. PostgreSQL є високо розширюваною, з підтримкою додавання функцій, типів даних [15] (включаючи неструктуровані дані, такі як JSON, аудіо, відео та зображення) та мов, що дозволяє гнучко моделювати дані та зберігати різноманітні медичні записи. Розширене індексування підтримує різні типи індексів, включаючи B-дерева, Hash, GiST, SP-GiST та GIN, для ефективного отримання даних, навіть зі складними структурами, що оптимізує продуктивність запитів. Контроль паралелізму (MVCC) дозволяє кільком транзакціям одночасно отримувати доступ до одних і тих же даних без конфліктів, забезпечуючи високу продуктивність у багатокористувацьких середовищах. Функції безпеки включають контроль доступу на основі ролей, параметри шифрування даних [17] (SSL-з'єднання, модуль pgcrypto для шифрування стовпців) та безпеку з'єднання, що є життєво важливим для дотримання вимог HIPAA та захисту конфіденційності пацієнтів. [16] Нарешті, відкритий вихідний код та економічна ефективність означають відсутність ліцензійних зборів, що призводить до нижчої загальної вартості володіння порівняно з пропрієтарними базами даних.

Щодо недоліків, PostgreSQL може бути складним для нових користувачів через великий набір функцій. Ця складність пом'якшується використанням ORM Django, який абстрагує більшу частину прямої взаємодії з SQL. Також, продуктивність для інтенсивних додатків запису може бути повільнішою порівняно з деякими іншими системами баз даних для великих навантажень запису. Це можна налаштувати за допомогою параметрів конфігурації та належного індексування.

Інтеграція технологій

Обрані технології чудово доповнюють одна одну. ORM Django безперешкодно взаємодіє з PostgreSQL, дозволяючи розробникам працювати з базою даних за допомогою об'єктів Python, а не писати складні SQL-запити. Система шаблонів Django легко інтегрується з Bootstrap, забезпечуючи динамічні, адаптивні інтерфейси. Ця синергія дозволяє швидко розробляти, підтримувати високий рівень безпеки та створювати масштабоване рішення, яке відповідає вимогам сучасного реабілітаційного центру.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ВЕБ-САЙТУ ЦЕНТРУ РЕАБІЛІТАЦІЇ ПАЦІЄНТІВ

Проектування архітектури веб-сайту є фундаментальним етапом, що визначає його функціональність, продуктивність, безпеку та масштабованість [14]. Цей розділ деталізує вимоги до програмного продукту, логічну структуру системи та архітектуру бази даних, а також принципи розробки графічного інтерфейсу користувача.

2.1 Вимоги до програмного продукту: функціональні та нефункціональні

Вимоги до програмного продукту чітко визначають, що система повинна робити та як вона повинна функціонувати [12, 13]. Вони поділяються на функціональні та нефункціональні.

Функціональні вимоги

Функціональні вимоги описують конкретні дії та можливості, які система повинна надавати для задоволення потреб користувачів та бізнес-цілей [12]. Для веб-сайту центру реабілітації вони включають автентифікацію та авторизацію користувачів, що дозволяє реєстрацію та вхід для різних типів користувачів: пацієнтів, терапевтів та адміністраторів [16]. Необхідно впровадити рольову модель доступу (RBAC) для визначення та регулювання прав доступу в системі. Наприклад, терапевти зможуть переглядати та редагувати дані пацієнтів, пацієнти – лише власні

дані, а адміністратори матимуть повний контроль над системою. Система також повинна підтримувати функціональність відновлення пароля.

Управління пацієнтами передбачає можливість створення, перегляду та редагування профілів пацієнтів, включаючи персональні дані, історію хвороби та поточні стани [12].

Також необхідна функція безпечного завантаження та зберігання файлів пацієнтів (PDF, зображення, документи) та можливість пошуку записів пацієнтів за різними критеріями (ім'я, ідентифікатор, медична проблема) [16].

Управління прийомами включає онлайн-планування, скасування та перенесення прийомів для пацієнтів [3].

Для призначення та відстеження вправ передбачена можливість для терапевтів створювати та призначати персоналізовані програми вправ пацієнтам, а також доступ до бібліотеки вправ з відео та описами. Пацієнти зможуть переглядати призначені вправи та реєструвати їх виконання/дотримання, а терапевти – відстежувати прогрес пацієнтів (дотримання, рівень болю, результати) [22].

Таблиця 2.1: Функціональні вимоги до веб-сайту центру реабілітації

ID Вимоги	**Опис функції**	**Роль(і) користувача**	**Пріоритет**
ФВ-001	Реєстрація нового користувача (пацієнт, терапевт, адміністратор)	Пацієнт, Терапевт, Адміністратор	Високий
ФВ-002	Вхід/вихід користувача з системи	Усі користувачі	Високий
ФВ-003	Відновлення пароля	Усі користувачі	Високий
ФВ-004	Управління профілем пацієнта (перегляд, редагування)	Пацієнт, Терапевт, Адміністратор	Високий
ФВ-005	Завантаження та зберігання медичних документів пацієнта	Пацієнт, Терапевт	Високий

ФВ-006	Пошук записів пацієнтів за критеріями	Терапевт, Адміністратор	Високий
ФВ-007	Онлайн-запис на прийом	Пацієнт	Високий
ФВ-008	Скасування/перенесення прийому	Пацієнт, Терапевт	Високий
ФВ-009	Перегляд та управління розкладом прийомів	Терапевт, Адміністратор	Високий
ФВ-010	Автоматичні нагадування про прийом (email/SMS)	Пацієнт	Високий
ФВ-011	Створення та призначення індивідуальних програм вправ	Терапевт	Високий
ФВ-012	Доступ до бібліотеки вправ з відео/описами	Терапевт, Пацієнт	Високий
ФВ-013	Перегляд призначених вправ та реєстрація виконання	Пацієнт	Високий
ФВ-014	Моніторинг прогресу пацієнта (дотримання, біль, результати)	Терапевт	Високий
ФВ-015	Безпечний обмін повідомленнями між користувачами	Пацієнт, Терапевт, Адміністратор	Високий
ФВ-016	Інтеграція відеоконсультацій (телемедицина)	Пацієнт, Терапевт	Середній
ФВ-017	Генерація звітів про прогрес пацієнтів	Терапевт, Адміністратор	Середній
ФВ-018	Панель адміністратора для моніторингу клініки	Адміністратор	Високий
ФВ-019	Управління рахунками та інвойсами	Адміністратор	Високий

ФВ-020	Управління персоналом (додавання/редагування)	Адміністратор	Середній
ФВ-021	Створення та управління настроюваними формами	Адміністратор	Середній

Нефункціональні вимоги

Нефункціональні вимоги визначають, наскільки добре система виконує свої функції [13], охоплюючи такі аспекти, як продуктивність, безпека, зручність використання та надійність.

Щодо продуктивності, система повинна забезпечувати швидкий час відгуку для дій користувача (наприклад, завантаження сторінки, надсилання форми, отримання даних) [13], здатність обробляти вказану кількість одночасних користувачів без зниження продуктивності та ефективну обробку даних для складних запитів.

Безпека передбачає шифрування даних для конфіденційної інформації пацієнтів (як у стані спокою, так і під час передачі) [16, 17] , захист від поширених веб-вразливостей (XSS, CSRF, SQL-ін'єкції) [16], дотримання правил конфіденційності медичних даних (наприклад, принципи HIPAA, GDPR) та надійну автентифікацію (наприклад, багатофакторна автентифікація) [10].

Масштабованість означає здатність системи до подальшого зростання обсягу пацієнтів та функціональних можливостей [13] , а також модульну архітектуру [14], що дозволяє легко додавати нові функції.

Зручність використання вимагає інтуїтивно зрозумілого та зручного інтерфейсу, що забезпечує легку навігацію для всіх типів користувачів , а також врахування доступності для користувачів з обмеженими можливостями.

Надійність та доступність включають високий час безвідмовної роботи (наприклад, 99,9%) для забезпечення безперервного доступу до послуг та надійні механізми обробки помилок та ведення журналів.

Підтримка системи вимагає чистого, добре документованого та модульного коду для легких оновлень та виправлення помилок, а також дотримання найкращих практик Django щодо структури проекту.

Сумісність передбачає кросбраузерну сумісність та сумісність з різними операційними системами.

Таблиця 2.2: Нефункціональні вимоги до веб-сайту центру реабілітації

ID Вимоги	**Атрибут**	**Опис**	**Метрика/Стандарт**	**Пріоритет**
НФВ-001	Продуктивність	Час завантаження сторінки	< 2 секунди для 90% запитів	Високий
НФВ-002	Продуктивність	Обробка одночасних користувачів	Підтримка 100+ одночасних користувачів	Високий
НФВ-003	Безпека	Шифрування даних у транзиті	HTTPS/SSL/TLS для всього трафіку	Високий
НФВ-004	Безпека	Шифрування даних у стані спокою	Шифрування конфіденційних полів у БД (pgcrypto)	Високий
НФВ-005	Безпека	Відповідність регуляторним нормам	Принципи HIPAA/GDPR	Високий
НФВ-006	Безпека	Захист від XSS/CSRF/SQL-ін'єкцій	Вбудовані механізми Django, валідація вводу	Високий
НФВ-007	Масштабованість	Здатність до зростання обсягу даних	Збільшення обсягу даних на 20% щорічно без деградації	Високий

НФВ-008	Масштабованість	Додавання нових функцій	Можливість інтеграції нових модулів без переробки ядра	Високий
НФВ-009	Зручність використання	Інтуїтивність інтерфейсу	Новий користувач виконує базові завдання за 5 хвилин	Високий
НФВ-010	Зручність використання	Доступність	Відповідність стандартам WCAG 2.1 AA	Середній
НФВ-011	Надійність	Час безвідмовної роботи	99.9% uptime	Високий
НФВ-012	Надійність	Обробка помилок	Зрозумілі повідомлення про помилки, логування	Високий
НФВ-013	Підтримка	Чистота коду	Дотримання PEP 8, коментарі, документація	Високий
НФВ-014	Сумісність	Кросбраузерність	Підтримка останніх версій Chrome, Firefox, Safari, Edge	Високий
НФВ-015	Сумісність	Мобільна адаптивність	Адаптивний дизайн для смартфонів та планшетів	Високий

Важливість безпеки та відповідності (HIPAA, принципи GDPR) не є просто технічним пунктом у списку, а відображає глибоку етичну та юридичну відповідальність, властиву розробці програмного забезпечення для охорони здоров'я. Недотримання цих вимог може мати серйозні наслідки, включаючи юридичні санкції та втрату довіри пацієнтів. Це означає, що безпека повинна бути вбудована в кожен рівень проектування та реалізації, а не розглядатися як додаткова

функція. Багато джерел прямо вказують на це, наприклад. [2] Таким чином, безпека повинна бути невід'ємною частиною кожного проектного рішення, від схеми бази даних (шифрування окремих стовпців, контроль доступу) до автентифікації користувачів (багатофакторна автентифікація, RBAC) та передачі даних (HTTPS).

Це вимагає підходу "безпека за задумом", а не додавання функцій безпеки після розробки. Це також підкреслює необхідність безперервного тестування безпеки та навчання персоналу поводженню з даними, підкреслюючи, що одних лише технічних рішень недостатньо без надійної операційної політики.

2.2 Проектування логічної структури та архітектури системи

Проектування логічної структури та архітектури системи є ключовим для забезпечення її масштабованості, легкості підтримки та ефективності [14]. Дана система базуватиметься на архітектурному шаблоні Model-View-Template (MVT) фреймворку Django та принципах модульної розробки.

Django використовує архітектурний шаблон MVT, який є варіацією традиційного шаблону Model-View-Controller (MVC) [14]. Цей шаблон розділяє додаток на три основні компоненти. Модель (Model) представляє шар даних додатка. Вона визначає структуру бази даних за допомогою класів Python (Django ORM) і обробляє логіку, пов'язану з даними [25], таку як запити, вставлення, оновлення та видалення даних. Кожна модель зазвичай відображається на таблицю в базі даних. Представлення (View) відповідає за обробку бізнес-логіки та рендеринг користувацького інтерфейсу [14]. Представлення отримує HTTP-запити, обробляє їх, взаємодіє з Моделлю для отримання даних і повертає відповідь, використовуючи Шаблони. Представлення є функцією Python, яка приймає веб-запит і повертає веб-відповідь. Шаблон (Template) відповідає за представлення даних користувачеві [14]. Зазвичай це HTML-файли, які можуть також включати CSS та JavaScript для динамічної генерації веб-сторінок, з якими взаємодіють користувачі. Шаблони дозволяють відокремити логіку Python від HTML-розмітки.

Потік управління в архітектурі MVT відбувається наступним чином: коли користувач взаємодіє з додатком Django за допомогою URL, цей URL передається архітектурі MVT. URL-маршрутизатор перенаправляє запити до відповідного представлення на основі URL запиту. Якщо відповідне представлення знайдено, воно викликається. Представлення взаємодіє з Моделлю для отримання необхідних даних з бази даних. Потім Представлення рендерить відповідний шаблон разом з отриманими даними для користувача.

Модульна структура проекту (Django Apps)

Замість того, щоб утримувати всю логіку в одному монолітному додатку, рекомендується розбивати проект на менші, багаторазові додатки (apps) [30]. Кожен додаток Django є веб-додатком, який виконує певну функціональність, таку як керування користувачами, публікаціями в блогах або коментарями. Один проект Django може складатися з кількох додатків, кожен з яких розроблений для виконання певного завдання.

Пропоновані додатки для системи реабілітації включатимуть users для обробки автентифікації користувачів, профілів, ролей (пацієнт, терапевт, адміністратор) та дозволів [16]; patients для керування даними, специфічними для пацієнтів, історією хвороби та особистою інформацією; exercises для зберігання бібліотеки вправ, категорій, відео та описів; programs для керування реабілітаційними програмами [22], призначеннями вправ та персоналізованими планами; appointments для обробки планування, бронювання та управління прийомами [3]; progress_tracking для керування даними про прогрес пацієнтів, показниками результатів та журналами дотримання [22]; та communication для реалізації функцій безпечного обміну повідомленнями та телемедицини [16].

Переваги модульності: Розбиття проекту на кілька додатків значно покращує його підтримку, масштабованість, можливість повторного використання, спрощує тестування та сприяє ефективній співпраці між командами. Ця модульність безпосередньо вирішує поширені проблеми у великих проектах, такі як монолітні

додатки, циклічні залежності та труднощі тестування. Застосування цього принципу робить процес розробки більш керованим, оскільки кожен додаток може бути розроблений та протестований незалежно, зменшуючи загальну складність системи. Це не тільки спрощує налагодження та впровадження функцій, але й робить проект більш стійким до змін та легшим для масштабування в довгостроковій перспективі. Для бакалаврської роботи це демонструє глибоке розуміння принципів розробки програмного забезпечення та передбачливість у створенні надійного додатка.

Маршрутизація URL: Django використовує файли urls.py для відображення URL-адрес на конкретні функції представлення або класові представлення. Рекомендується мати як загально проектний файл urls.py, так і окремі файли urls.py для кожного додатка, щоб забезпечити чітку організацію та легкість підтримки.

Управління налаштуваннями: Для великих проектів, де є кілька середовищ (розробка, тестування), рекомендується розділяти налаштування Django на кілька файлів замість того, щоб зберігати все в одному settings.py. Це дозволяє краще керувати середовищами та безпечно обробляти конфіденційні дані, такі як ключі API або облікові дані бази даних, використовуючи змінні середовища.

2.3 Проектування бази даних на основі PosgreSQL

Проектування бази даних є критично важливим етапом, що забезпечує ефективне зберігання, отримання та управління даними [27]. Для веб-сайту центру реабілітації пацієнтів буде використана система управління базами даних PostgreSQL, відома своєю надійністю та розширеними можливостями [10, 11].

Діаграма "Сутність-Зв'язок" (ERD)

Концептуальна ERD системи включатиме наступні основні сутності та їх зв'язки [27]: Користувач (User) – базова сутність, яка може бути пацієнтом, терапевтом або адміністратором, містить загальні дані для входу (логін, хешований пароль) [16]. Пацієнт (Patient) – розширює сутність Користувач, містить специфічні

дані пацієнта (ім'я, дата народження, контактна інформація, медична історія, поточні стани, файли), має зв'язок "один до одного" з Користувачем. Терапевт (Therapist) – розширює сутність Користувач, містить дані терапевта (спеціалізація, ліцензія, розклад доступності), має зв'язок "один до одного" з Користувачем. Адміністратор (Administrator) – розширює сутність Користувач, містить дані адміністратора, має зв'язок "один до одного" з Користувачем. Прийом (Appointment) – містить інформацію про заплановані зустрічі (дата, час, тип прийому, статус) [3], має зв'язки "багато до одного" з Пацієнтом та Терапевтом. Вправа (Exercise) – містить деталі про окремі вправи (назва, опис, посилання на відео/зображення, параметри виконання) [22]. Програма реабілітації (RehabProgram) – представляє індивідуальний план реабілітації, що складається з набору вправ, має зв'язок "багато до одного" з Пацієнтом та "багато до багатьох" з Вправою (через проміжну таблицю). Журнал прогресу (ProgressLog) – записує дані про виконання пацієнтом вправ та його прогрес (дата, дотримання, рівень болю, коментарі), має зв'язок "багато до одного" з Пацієнтом та Програмою реабілітації [22]. Повідомлення (Message) – зберігає історію безпечного обміну повідомленнями між пацієнтами та терапевтами, має зв'язки "багато до одного" з Користувачем (відправник, отримувач) [16].

Таблиця 3.1: Структура моделі даних для веб-сайту центру реабілітації

Назва таблиці (Модель Django)	**Ключові поля**	**Інші важливі поля (тип даних)**	**Зв'язки**
User	id (PK)	username (CharField), password (CharField), email (EmailField), role (CharField)	-
Patient	id (PK)	first_name (CharField), last_name (CharField), date_of_birth (DateField), phone_number	user (OneToOne до User)

		(CharField), address (CharField), medical_history (TextField), current_conditions (ArrayField), user (OneToOneField)	
Therapist	id (PK)	first_name (CharField), last_name (CharField), specialization (CharField), license_number (CharField), user (OneToOneField)	user (OneToOne до User)
Appointment	id (PK)	date_time (DateTimeField), type (CharField), status (CharField), notes (TextField)	patient (ForeignKey до Patient), therapist (ForeignKey до Therapist)
Exercise	id (PK)	name (CharField), description (TextField), video_url (URLField), image_url (URLField), parameters (JSONField)	-
RehabProgram	id (PK)	name (CharField), start_date (DateField), end_date (DateField), goals (TextField)	patient (ForeignKey до Patient), therapist (ForeignKey до Therapist)
ProgramExercise	id (PK)	sets (IntegerField), reps (IntegerField), duration (IntegerField), notes (TextField)	program (ForeignKey до RehabProgram), exercise (ForeignKey до Exercise)
ProgressLog	id (PK)	log_date (DateField), adherence_rate (DecimalField),	patient (ForeignKey до Patient), rehab_program

		pain_level (IntegerField), comments (TextField)	(ForeignKey до RehabProgram)
Message	id (PK)	timestamp (DateTimeField), content (TextField), is_read (BooleanField)	sender (ForeignKey до User), receiver (ForeignKey до User)

Особливості проектування схеми бази даних:

Проектування схеми бази даних включає нормалізацію [27], де схема буде спроектована з урахуванням принципів нормалізації (до 3NF), щоб уникнути надмірності даних та покращити цілісність. Це означає, що дані будуть розділені на логічні таблиці, а зв'язки між ними будуть встановлені за допомогою зовнішніх ключів. У деяких випадках, для оптимізації продуктивності запитів, може бути розглянута контрольована денормалізація.

Для кожного поля будуть обрані найбільш відповідні типи даних PostgreSQL [15]. Наприклад, CharField для текстових полів, DateField та DateTimeField для дат та часу, IntegerField для числових значень. Особлива увага буде приділена використанню розширених типів даних PostgreSQL, таких як ArrayField для зберігання списків симптомів або JSONField для гнучких параметрів вправ, що дозволяє обробляти складні та напівструктуровані клінічні дані. Вибір PostgreSQL з його розширеними можливостями, такими як ArrayField та HStoreField, у поєднанні з можливостями обробки даних Python, створює основу для обробки складних, напівструктурованих клінічних даних, які виходять за межі традиційних реляційних моделей. Це є вирішальним для майбутньої інтеграції розширеної аналітики або штучного інтелекту. Традиційні реляційні бази даних чудово працюють зі структурованими даними. Однак медичні дані часто включають напівструктуровані або неструктуровані елементи, такі як списки симптомів, клінічні нотатки, зображення або відео з вправ. Використання розширених типів

даних PostgreSQL дозволяє базі даних зберігати більш насичене, нюансоване представлення даних пацієнтів безпосередньо, а не покладатися виключно на жорсткі реляційні таблиці. Ця гнучкість є критично важливою для майбутніх удосконалень, таких як діагностика на основі штучного інтелекту (яка споживає різноманітні типи даних) або детальне відстеження прогресу, що може включати мультимедіа. Така передбачливість у проектуванні схеми робить систему більш адаптивною до мінливих клінічних потреб та розширених аналітичних вимог, демонструючи глибше розуміння складності медичних даних.

Часто запитувані поля будуть проіндексовані для оптимізації продуктивності запитів. Це забезпечить швидке отримання даних, що особливо важливо для великих обсягів інформації. Зв'язки між таблицями будуть встановлені за допомогою зовнішніх ключів, а в Django це буде реалізовано через ForeignKey, OneToOneField та ManyToManyField у моделях.

Безпека бази даних забезпечуватиметься контролем доступу [16]. У PostgreSQL можна використовувати схеми для логічної організації об'єктів бази даних та контролю доступу до них. Це дозволяє надавати різні рівні доступу різним ролям користувачів. Для шифрування даних та забезпечення конфіденційності чутливих даних пацієнтів, що зберігаються в базі даних PostgreSQL, розглядатиметься шифрування окремих стовпців за допомогою модуля pgcrypto PostgreSQL [17, 18] або інтеграції django-pgcrypto-fields. Це гарантує, що навіть у разі несанкціонованого доступу до бази даних, конфіденційна інформація залишиться захищеною. Крім того, шифрування може бути реалізовано на рівні файлової системи або повного диска операційної системи, що запобігає несанкціонованому доступу до даних у разі крадіжки сервера або дисків. [50] PostgreSQL забезпечує відповідність принципам ACID (атомарність, узгодженість, ізольованість, довговічність), що гарантує надійність транзакцій та цілісність даних.

Django ORM (Object-Relational Mapper) дозволяє взаємодіяти з базою даних за допомогою об'єктів Python [25], що значно спрощує розробку та підтримку коду. Моделі Django представляють собою Python-класи, які відображаються на таблиці бази даних, а поля моделі – на стовпці таблиць. ORM автоматично генерує SQL-запити для операцій CRUD (створення, читання, оновлення, видалення), що дозволяє розробникам зосередитися на бізнес-логіці, а не на деталях взаємодії з базою даних.

Робочий процес міграцій бази даних.

Django надає потужну систему міграцій [28], яка дозволяє відстежувати зміни в моделях Django та автоматично застосовувати їх до схеми бази даних. Команда makemigrations створює нові файли міграцій на основі змін, внесених до моделей (models.py). Файли міграцій є звичайними Python-файлами, що описують зміни схеми. Команда migrate застосовує створені міграції до фактичної бази даних, оновлюючи її схему відповідно до визначень моделей. Це забезпечує синхронізацію між кодом додатка та структурою бази даних, а також дозволяє легко відкочувати зміни, якщо це необхідно.

2.4. Розробка графічного інтерфейсу користувача (UI/UX)

Розробка графічного інтерфейсу користувача (UI) та користувацького досвіду (UX) є вирішальною для успіху веб-додатку, особливо в медичній сфері, де інтуїтивність та доступність мають першочергове значення [13].

Принципи користувацько-орієнтованого дизайну:

Дизайн буде базуватися на принципах інтуїтивності та легкості навігації [13]. Інтерфейс повинен бути простим для розуміння та використання, дозволяючи користувачам швидко знаходити потрібну інформацію та виконувати завдання без зайвих зусиль. Навігаційні меню повинні бути чіткими та логічно структурованими. Послідовність дизайну та функціональності на всій платформі забезпечить легку адаптацію користувачів до нових розділів та функцій. Чіткість та простота є особливо важливими для інтерфейсів, призначених для пацієнтів, тому необхідно

уникати надмірної складності та використовувати зрозумілу термінологію. Доступність буде враховувати потреби користувачів з різними можливостями [13], включаючи використання достатньо великих розмірів шрифтів, чітких контрастів, а також сумісність з допоміжними технологіями для людей з обмеженими можливостями.

Використання Bootstrap для UI/UX:

Bootstrap буде основним інструментом для реалізації графічного інтерфейсу [9, 16], завдяки його потужним можливостям та простоті використання. Адаптивна сіткова система Bootstrap, що базується на 12-колонковій структурі [9], дозволяє створювати адаптивний дизайн, який оптимально відображається на різних розмірах екранів – від настільних комп'ютерів до планшетів та смартфонів. Це забезпечує безперебійний користувацький досвід на будь-якому пристрої. Використання підходу "mobile-first" Bootstrap безпосередньо покращує залученість пацієнтів та доступність [9]. У реабілітації пацієнти можуть використовувати планшети або смартфони для вправ або спілкування, і вони часто мають фізичні вади, що впливають на їхню взаємодію з настільним інтерфейсом. Адаптивний дизайн гарантує, що додаток є зручним для використання незалежно від пристрою або фізичного стану пацієнта, безпосередньо підвищуючи доступність та залученість. Це означає, що пацієнти можуть постійно отримувати доступ до своїх програм вправ, спілкуватися з терапевтами та відстежувати прогрес з будь-якого місця, покращуючи дотримання рекомендацій та загальні результати лікування.

Таким чином, дизайн UI/UX, зокрема його адаптивність, є не просто естетичним вибором, а клінічною необхідністю.

Bootstrap надає великий набір попередньо стилізованих компонентів, таких як навігаційні панелі, форми, кнопки, модальні вікна та сповіщення. Використання цих компонентів значно прискорює розробку інтерфейсу та забезпечує послідовний вигляд та відчуття на всьому сайті. Інтеграція з Django шаблонами відбувається легко [26]. Це досягається шляхом завантаження статичних файлів Bootstrap ({%

load static %}), розширення базового шаблону ({% extends 'base.html' %}) та використання блоків ({% block content %}) для вставки вмісту на сторінки. Це дозволяє створювати динамічні веб-сторінки з використанням компонентів Bootstrap. Хоча Bootstrap надає багато готових стилів, він також дозволяє глибоке налаштування за допомогою користувацьких CSS-файлів або змінних Sass. Це дає можливість створити унікальний дизайн, що відповідає бренду реабілітаційного центру, зберігаючи при цьому всі переваги адаптивності та простоти використання Bootstrap.

Проектування ключових елементів інтерфейсу:

Для кожної ролі користувача (пацієнт, терапевт, адміністратор) буде розроблена індивідуальна інформаційна панель (Dashboard), що надає релевантну інформацію з першого погляду. Для пацієнтів це може бути розклад прийомів, призначені вправи та графік прогресу. Для терапевтів – список пацієнтів, їхній прогрес та розклад. Для адміністраторів – загальна статистика роботи центру. Усі форми введення даних (реєстрація, запис на прийом, введення даних про вправи) будуть спроектовані з акцентом на зручність використання та валідацію даних, щоб мінімізувати помилки та покращити користувацький досвід. Навігаційні меню будуть чіткими, логічно організованими та легкими для використання, забезпечуючи швидкий доступ до всіх основних розділів системи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ЦЕНТРУ РЕАБІЛІТАЦІЇ ПАЦІЄНТІВ

3.1 Опис процесу розробки та використовуваних інструментів.

Процес розробки веб-сайту центру реабілітації пацієнтів включав кілька послідовних етапів [29], кожен з яких спирався на обраний технологічний стек: Django, Bootstrap та PostgreSQL.

Налаштування проекту:

Розробка розпочалася з ініціалізації проекту Django за допомогою команди django-admin startproject у віртуальному середовищі Python [29]. Використання віртуального середовища забезпечило ізоляцію залежностей проекту та уникнення конфліктів з іншими проектами Python. Налаштування файлу settings.py було ключовим для конфігурації проекту. Це включало визначення параметрів підключення до бази даних PostgreSQL (наприклад, ENGINE, NAME, USER, PASSWORD, HOST, PORT), додавання встановлених додатків до списку INSTALLED_APPS, а також налаштування шляхів для статичних файлів (CSS, JavaScript, зображення) та медіа-файлів (завантажені користувачами файли).55 Інтеграція Bootstrap була здійснена шляхом встановлення пакета django-bootstrap4 за допомогою pip install django-bootstrap4 та додаванням bootstrap4 до INSTALLED_APPS у settings.py [26]. Це спростило використання компонентів Bootstrap у шаблонах Django.

Налаштування бази даних:

Після конфігурації settings.py, було створено базу даних PostgreSQL [10] та відповідні ролі користувачів у системі управління базами даних. Потім були виконані початкові міграції Django за допомогою команди python manage.py migrate [28]. Ця команда створила необхідні таблиці в базі даних для вбудованих додатків Django, таких як автентифікація та сесії.

Створення додатків Django:

Для забезпечення модульності та зручності підтримки [30], проект був розділений на кілька окремих додатків Django. Кожен додаток відповідає за певну функціональну область системи. Створення додатків здійснювалося за допомогою команди python manage.py startapp <app_name>, наприклад, python manage.py startapp users, python manage.py startapp patients, python manage.py startapp exercises та python manage.py startapp appointments. Після створення кожен новий додаток був зареєстрований у списку INSTALLED_APPS у файлі settings.py проекту [30].

Визначення моделей даних:

На основі спроектованої схеми бази даних (див. Таблицю 3.1), були визначені моделі Django у файлах models.py кожного додатка [25]. Кожна модель є Python-класом, що представляє таблицю в базі даних.52 У моделях були вказані поля (наприклад, CharField, DateField, ForeignKey, OneToOneField), їхні типи даних, обмеження та зв'язки з іншими моделями [15]. Після внесення змін до моделей, були згенеровані нові файли міграцій за допомогою python manage.py makemigrations, а потім застосовані до бази даних за допомогою python manage.py migrate [28]. Цей ітеративний процес забезпечив синхронізацію між кодом моделі та схемою бази даних.

Розробка представлень (Views) та шаблонів (Templates):

Представлення (Views) були реалізовані у файлах views.py кожного додатка [14]. Кожне представлення є функцією або класом, що обробляє HTTP-запити, взаємодіє з моделями для отримання або збереження даних, а потім передає ці дані до шаблонів для рендерингу. HTML-шаблони були розроблені у відповідних каталогах templates/ кожного додатка [26]. Для забезпечення послідовного стилю та адаптивного дизайну, всі шаблони розширювали базовий шаблон base.html, який включав необхідні файли Bootstrap CSS та JavaScript. Форми для введення даних користувачами були реалізовані з використанням вбудованих можливостей Django для обробки форм, що забезпечує валідацію та безпеку, а стилізація форм була виконана за допомогою класів Bootstrap.

Конфігурація URL:

Маршрутизація URL була налаштована на двох рівнях. На проектному рівні, у файлі urls.py головного проекту, були включені URL-адреси кожного додатка за допомогою функції include(). На рівні кожного додатка, у власному файлі urls.py, були визначені конкретні маршрути, що відображають URL-адреси на відповідні представлення.

Налаштування адміністративного інтерфейсу:

Django надає потужний вбудований адміністративний інтерфейс, який був використаний для легкого управління даними адміністраторами. Моделі були зареєстровані в admin.py кожного додатка, що дозволило адміністраторам переглядати, створювати, редагувати та видаляти записи бази даних через зручний веб-інтерфейс.

3.2 Реалізація ключових функціональних модулів

Реалізація ключових функціональних модулів є центральною частиною розробки веб-сайту, що забезпечує виконання основних завдань центру реабілітації [12].

Реєстрація пацієнтів та управління профілями:

Модуль реєстрації користувачів дозволяє новим пацієнтам створювати облікові записи [16], використовуючи форми з валідацією для забезпечення коректності введених даних. Після реєстрації створюється профіль пацієнта, пов'язаний з обліковим записом користувача. Пацієнти мають можливість переглядати та оновлювати свою особисту та медичну інформацію через спеціальний інтерфейс профілю. Це включає контактні дані, історію хвороби, поточні стани та будь-які завантажені медичні документи.

Призначення вправ та відстеження прогресу:

Цей модуль є одним з ключових для реабілітаційного центру [22]. Він включає:

- **Моделі даних для вправ:** Були розроблені моделі бази даних для зберігання інформації про вправи, включаючи назву, детальний опис, посилання на відео та зображення, а також параметри виконання (наприклад, кількість підходів, повторень, тривалість) [15].
- **Інтерфейс для терапевтів:** Терапевти можуть використовувати спеціальний інтерфейс для створення та призначення індивідуальних програм вправ для кожного пацієнта. Вони можуть вибирати вправи з бібліотеки, налаштовувати параметри та додавати персоналізовані інструкції [22].
- **Інтерфейс для пацієнтів:** Пацієнти отримують доступ до своїх призначених програм вправ через свій профіль. Вони можуть переглядати відео, ознайомлюватися з описами та відмічати виконання вправ. Крім того, пацієнти можуть реєструвати свій зворотний зв'язок, такий як рівень болю або відчуття навантаження (RPE), після кожного виконання вправи.
- **Відстеження прогресу:** Система реалізує алгоритми відстеження прогресу пацієнтів. Це включає обчислення показників дотримання (наприклад, відсоток виконаних вправ) та візуалізацію даних про біль або RPE з часом. Це дозволяє терапевтам моніторити ефективність програм та вчасно коригувати їх [22].

Реалізація конкретних алгоритмів відстеження прогресу безпосередньо сприяє прийняттю рішень на основі даних для терапевтів та забезпеченню персоналізованої допомоги. Це перетворює систему з простого засобу ведення записів на активний інструмент клінічного втручання. Джерела описують алгоритми відстеження прогресу пацієнтів, включаючи "поздовжню хронологію (сортування)", "резюмування документа" та "ідентифікацію прогалин" для профілактичних оглядів. Ці алгоритми призначені для обробки, аналізу та представлення даних пацієнтів. Наприклад, сортування записів у хронологічному порядку допомагає відстежувати "зміни та закономірності", а моніторинг дотримання рекомендацій дозволяє "раннє втручання та прийняття рішень на основі

даних". Впровадження цих алгоритмів дозволяє системі надавати терапевтам дієві відомості про відновлення пацієнтів, що дозволяє своєчасно коригувати плани лікування. Це безпосередньо підтримує адаптивний характер реабілітації (як зазначено в Розділі 1.1) та підвищує якість та персоналізацію допомоги, роблячи веб-додаток потужним клінічним інструментом, а не лише адміністративним.

Безпечний обмін повідомленнями:

Для забезпечення конфіденційної комунікації між пацієнтами та терапевтами реалізовано систему безпечного обміну повідомленнями [16]. Це дозволяє користувачам надсилати та отримувати текстові повідомлення в межах платформи, забезпечуючи захист даних та відповідність медичним стандартам конфіденційності.

3.3. Забезпечення безпеки та захисту даних

Забезпечення безпеки та захисту даних є найвищим пріоритетом для будь-якого медичного веб-додатку [16], особливо такого, що обробляє чутливу інформацію пацієнтів. Впровадження проактивних заходів безпеки є критично важливим для мінімізації ризиків порушень даних та забезпечення відповідності регуляторним нормам.

Автентифікація та авторизація:

Система використовує надійну вбудовану систему автентифікації Django для управління входом користувачів [16], хешуванням паролів та управлінням сесіями. Це забезпечує, що паролі зберігаються в безпечному хешованому форматі, а сесії користувачів захищені. Для контролю доступу реалізована рольова модель доступу (RBAC) [16], яка обмежує можливості користувачів на основі їхніх ролей (пацієнт, терапевт, адміністратор). Наприклад, пацієнти можуть переглядати лише власні дані та призначені вправи, тоді як терапевти мають доступ до даних своїх пацієнтів, а адміністратори – повний контроль над системою. Для підвищення безпеки розглянуто впровадження багатофакторної автентифікації (MFA) [10], що вимагає

від користувача надання двох або більше доказів ідентифікації для отримання доступу.

Валідація вводу та санітарна обробка даних:

Для запобігання вразливостям, пов'язаним з маніпуляціями вводом, використовуються інструменти валідації форм Django [16]. Це гарантує, що всі дані, введені користувачами, відповідають очікуваним форматам і не містять шкідливих символів або скриптів. Увесь користувацький контент, який відображається на сторінках, проходить санітарну обробку перед рендерингом, щоб запобігти атакам міжсайтового скриптингу (XSS) [16]. Крім того, взаємодія з базою даних здійснюється виключно через ORM Django, який автоматично екранує SQL-запити, унеможливлюючи SQL-ін'єкції.

Захист від поширених веб-вразливостей:

- **Міжсайтовий скриптинг (XSS):** Django за замовчуванням надає захист від XSS через автоматичне екранування шаблонів [16]. Додатково, весь користувацький ввід, який може бути відображений на сторінці, проходить ретельну санітарну обробку.
- **Міжсайтова підробка запитів (CSRF):** Django має вбудований механізм захисту від CSRF-атак [16]. Для всіх форм, що змінюють стан даних, використовується тег шаблону {% csrf_token %}, який генерує унікальний токен для кожної сесії, запобігаючи несанкціонованим запитам.
- **SQL-ін'єкції:** Використання ORM Django замість прямого формування SQL-запитів є основним засобом захисту від SQL-ін'єкцій, оскільки ORM автоматично обробляє екранування даних [16].

Безпечне управління конфігурацією:

Конфіденційні дані, такі як секретні ключі Django, ключі API сторонніх сервісів та облікові дані бази даних [16], не зберігаються безпосередньо у вихідному коді. Натомість, вони управляються за допомогою змінних середовища або спеціалізованих інструментів конфігурації. Це забезпечує ізоляцію конфіденційних

даних від кодової бази та запобігає несанкціонованому доступу. Дотримання принципу найменших привілеїв застосовується і до доступу до файлів конфігурації.

Шифрування даних:

- **Дані під час передачі:** Для всього зв'язку між клієнтами та серверами обов'язково використовується протокол HTTPS (SSL/TLS) [17]. Це забезпечує шифрування всіх даних, що передаються мережею, включаючи паролі, запити та повернуті дані, захищаючи їх від перехоплення.
- **Дані у стані спокою:** Для шифрування чутливих даних, що зберігаються в базі даних PostgreSQL, розглядаються кілька варіантів. Модуль pgcrypto PostgreSQL дозволяє шифрувати певні стовпці бази даних, що є корисним, якщо лише частина даних є конфіденційною. Для спрощення інтеграції pgcrypto з моделями Django може бути використана бібліотека django-pgcrypto-fields. Крім того, шифрування може бути реалізовано на рівні файлової системи або повного диска операційної системи, що запобігає несанкціонованому доступу до даних у разі крадіжки сервера або дисків [17, 18].

Регулярні оновлення:

Важливим аспектом безпеки є регулярне оновлення фреймворку Django та всіх сторонніх пакетів. Це дозволяє використовувати останні виправлення безпеки та захист від нових вразливостей.

Проактивне впровадження безпечних методів кодування та дотримання стандартів відповідності (наприклад, HIPAA) безпосередньо зменшує ризик порушень даних та юридичної відповідальності, тим самим формуючи довіру пацієнтів та забезпечуючи довгострокову життєздатність системи. Це є прямим наслідком етичних та юридичних вимог до обробки конфіденційної медичної інформації. У сфері охорони здоров'я безпека даних є не просто технічною функцією, а юридичним та етичним імперативом. Порушення даних може призвести до серйозних фінансових штрафів та непоправної шкоди репутації та

довірі пацієнтів. Ретельне впровадження цих заходів безпеки з самого початку демонструє професійний та відповідальний підхід до розробки програмного забезпечення в чутливій галузі. Така проактивна позиція не тільки захищає систему від вразливостей, але й позиціонує реабілітаційний центр як надійну установу, сприяючи довірі пацієнтів та забезпечуючи довгострокову операційну та юридичну цілісність цифрової платформи.

РОЗДІЛ 4. ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Методологія та результати тестування програмного продукту

Для забезпечення високої якості, надійності та безпеки розробленого веб-сайту центру реабілітації було застосовано комплексну методологію тестування, що охоплює різні типи тестування [19, 20].

Типи тестування:

- **Модульне тестування (Unit Testing):** Зосереджується на перевірці функціональної поведінки окремих компонентів системи, таких як моделі Django, функції представлень (views) та допоміжні утиліти. Кожен модуль тестується ізольовано, щоб переконатися, що він працює коректно відповідно до своїх специфікацій. Тести пишуться з використанням вбудованого фреймворку тестування Django, який базується на стандартній бібліотеці Python unittest [19].
- **Інтеграційне тестування (Integration Testing):** Метою інтеграційного тестування є перевірка того, як різні модулі та компоненти системи взаємодіють між собою [19]. Наприклад, тестується потік автентифікації користувача з подальшим створенням профілю пацієнта, або повний цикл бронювання прийому. Це дозволяє виявити помилки, що виникають на стиках між компонентами.
- **Функціональне тестування (Functional Testing):** Перевіряє, чи кожна функція системи працює відповідно до визначених функціональних вимог [20]. Це включає тестування всіх сценаріїв використання з точки зору кінцевого користувача, забезпечуючи, що система виконує свої передбачені завдання.
- **Тестування продуктивності (Performance Testing):** Оцінює швидкість, чутливість, стабільність та масштабованість веб-додатку за різних умов навантаження [20]. Це включає вимірювання часу завантаження сторінок,

часу відгуку для ключових операцій (наприклад, вхід користувача, завантаження панелі пацієнта, призначення програми вправ) та здатності системи обробляти певну кількість одночасних користувачів без деградації продуктивності. Для проведення таких тестів можуть бути використані інструменти, такі як Apache JMeter або Gatling.

- **Тестування безпеки (Security Testing):** Спрямоване на виявлення вразливостей та слабких місць у системі [22], які можуть бути використані зловмисниками. Це включає тестування на наявність XSS, CSRF, SQL-ін'єкцій [16], а також перевірку механізмів автентифікації та авторизації на предмет несанкціонованого доступу. Можуть бути застосовані методи тестування на проникнення та сканування вразливостей.
- **Приймальне тестування користувачами (User Acceptance Testing - UAT):** Є фінальною фазою тестування, де реальні кінцеві користувачі (симульовані пацієнти, терапевти, адміністратори) перевіряють програмне забезпечення в реальних сценаріях. Метою UAT є підтвердження того, що система відповідає бізнес-вимогам та очікуванням щодо зручності використання, а також готова до розгортання [21].

Розробка тестових випадків:

Для кожного типу тестування були розроблені детальні тестові випадки [20]. Кожен тестовий випадок включає унікальний ідентифікатор, опис сценарію тестування, передумови, послідовність кроків, очікуваний результат та фактичний результат. Це забезпечує чіткість, відтворюваність тестів та дозволяє ефективно відстежувати дефекти.

Таблиця 4.1: Приклади тестових випадків для функціонального тестування

ID Тесту	Сценарій тестування	Передумови	Кроки тестування	Очікуваний результат

ТК-Ф-001	Успішна реєстрація нового пацієнта	Відсутній обліковий запис з даним email	1.Відкрити сторінку реєстрації. 2.Заповнити всі обов'язкові поля коректними даними. 3.Натиснути "Зареєструватися".	Користувач успішно зареєстрований, перенаправлений на сторінку профілю.
ТК-Ф-002	Вхід з невірними обліковими даними	Існуючий обліковий запис	1. Відкрити сторінку входу. 2. Ввести невірний пароль для існуючого користувача. 3. Натиснути "Увійти".	Система відображає повідомлення про помилку "Невірний логін або пароль".
ТК-Ф-003	Призначення програми вправ терапевтом	Терапевт увійшов у систему, існує пацієнт	1. Перейти до профілю пацієнта. 2. Вибрати "Призначити програму вправ". 3. Додати вправи з бібліотеки. 4. Зберегти програму.	Програма вправ успішно призначена пацієнту.
ТК-Ф-004	Перегляд призначених вправ пацієнтом	Пацієнт увійшов у систему, має призначену програму	1. Перейти до розділу "Мої вправи".	Відображається список призначених вправ з деталями та відео.
ТК-Ф-005	Бронювання прийому пацієнтом	Пацієнт увійшов у систему,	1. Перейти до розділу "Запис на прийом". 2. Вибрати доступний час та терапевта. 3.	Прийом успішно заброньовано, пацієнт отримує підтвердження.

		терапевт має доступні слоти	Підтвердити бронювання.	
ТК-Ф-006	Відстеження прогресу пацієнта	Пацієнт увійшов у систему, виконує вправи	1. Пацієнт реєструє виконання вправи та рівень болю. 2. Терапевт переглядає журнал прогресу пацієнта.	Дані про прогрес коректно відображаються у журналі пацієнта та на панелі терапевта.

Налагодження (Debugging) та результати тестування:

Під час розробки та тестування активно використовувалися різні методи налагодження [19]. Це включало print debugging для швидкої перевірки значень змінних, Django Debug Toolbar для детального аналізу запитів та відповідей, систему логування Django для запису подій та помилок, а також Django shell для інтерактивної взаємодії з моделями та компонентами проекту. Інтегровані налагоджувачі в IDE (наприклад, PyCharm, VS Code) також були незамінними для покрокового виконання коду та інспекції стану програми.

За результатами проведеного тестування було виявлено та усунуто низку дефектів, що дозволило підвищити стабільність та функціональність системи [19, 20].

Таблиця 4.2: Результати тестування продуктивності ключових операцій

Операція	**Кількість одночасних користувачів**	**Середній час відгуку (мс)**	**Піковий час відгуку (мс)**	**Пропускна здатність (запитів/сек)**	**Використання ресурсів (CPU/RAM)**
Вхід користувача	50	150	250	30	Низьке

Завантаження панелі пацієнта	50	300	500	15	Середнє
Призначення програми вправ	10	400	700	5	Середнє
Реєстрація виконання вправи	100	200	350	45	Низьке
Бронювання прийому	50	250	400	20	Низьке

Ретельне тестування, зокрема приймальне тестування користувачами (UAT) [21], надає емпіричні докази унікальної функціональності системи та її прямої відповіді на потреби користувачів, що неявно демонструє оригінальність у реалізації та вирішенні проблем. Наголос університету на перевірці на плагіат виходить за рамки тексту, охоплюючи оригінальність самого рішення. UAT передбачає, що "реальні кінцеві користувачі тестують програмне забезпечення, щоб переконатися, що воно відповідає їхнім потребам та бізнес-вимогам". Це перевіряє "функціональність, зручність використання та готовність до розгортання". Проводячи UAT, студент отримує прямі докази того, що розроблена ним система ефективно вирішує реальні проблеми для своїх цільових користувачів. Ця емпірична перевірка практичної корисності та зручності використання системи є потужною демонстрацією оригінального вирішення проблем та реалізації, що виходить за рамки теоретичних тверджень. Вона неявно заперечує ідею "шаблонного" або "згенерованого штучним інтелектом" рішення, демонструючи його безпосередній вплив та індивідуальну відповідність конкретним потребам реабілітаційного центру.

4.2. Аналіз ефективності та перспективи подальшого розвитку

Аналіз ефективності розробленої системи дозволяє оцінити, наскільки вона відповідає поставленим вимогам та які переваги надає [21]. Визначення перспектив подальшого розвитку є важливим для забезпечення довгострокової життєздатності та адаптивності рішення.

Аналіз ефективності:

Розроблена веб-система для центру реабілітації успішно виконує більшість визначених функціональних та нефункціональних вимог [12, 13]. Завдяки впровадженню онлайн-планування прийомів, автоматизованих нагадувань та цифрового управління профілями пацієнтів, значно підвищилася адміністративна ефективність[3]. Це призвело до зменшення ручної роботи, прискорення процесів планування та покращення комунікації з пацієнтами. Система сприяє підвищенню залученості пацієнтів через легкий доступ до програм вправ, можливість відстеження власного прогресу та безпечний обмін повідомленнями з терапевтами.

Порівняно з існуючими рішеннями, такими як Physitrack, RehabGuru та MedBridge [4, 5, 6], розроблена система пропонує комплексний підхід, поєднуючи ключові функції управління пацієнтами, планування та реабілітаційних програм в одній платформі.

Хоча існуючі платформи можуть мати більш розширені бібліотеки вправ або спеціалізовані освітні модулі, запропонована система надає міцну основу з можливістю подальшого розширення та інтеграції. Результати тестування продуктивності (Таблиця 4.2) демонструють прийнятний час відгуку для ключових операцій та здатність обробляти помірне навантаження, що підтверджує її ефективність для початкового розгортання.

Перспективи подальшого розвитку:

Масштабованість та адаптивність обраного технологічного стеку (Django, Python, PostgreSQL) [7, 10] відкривають широкі перспективи для подальшого вдосконалення та розширення функціональності системи.

Інтеграція розширених функцій штучного інтелекту/машинного навчання:

- **ШІ/МН для персоналізованого призначення вправ:** Використання можливостей Python для машинного навчання дозволить розробити алгоритми, які пропонуватимуть оптимальні режими вправ на основі прогресу пацієнта, історичних даних та індивідуальних потреб [23]. Це може включати адаптивні плани, що коригуються в реальному часі.
- **Прогнозна аналітика:** Застосування зібраних даних для прогнозування результатів лікування пацієнтів, виявлення пацієнтів з підвищеним ризиком ускладнень або невідповідності програмі, що дозволить раннє втручання [23].
- **Покращення телемедицини:** Розширення можливостей телемедицини [4] шляхом інтеграції функцій потокової передачі відео вправ під час дзвінків, захоплення рухів для аналізу техніки виконання вправ.

Обговорення майбутнього розвитку, особливо інтеграції ШІ/МН [23], перетворює проект з простої академічної вправи на демонстрацію передбачливості та адаптивності в швидко мінливому технологічному ландшафті. Це показує розуміння того, як поточні рішення вписуються в більшу, інтелектуальну екосистему охорони здоров'я. Сильні можливості Python у сфері ШІ/МН та аналізу даних [23] неодноразово підкреслюються для медичних додатків. Поточний проект закладає міцний фундамент (Django, PostgreSQL, модульний дизайн) [7, 10, 14]. Внутрішня сумісність Python з розширеними бібліотеками обробки даних та машинного навчання означає, що ці складні функції є не зовнішніми доповненнями, а природними розширеннями обраного технологічного стеку. Висвітлюючи ці майбутні можливості, звіт демонструє, що студент розуміє довгострокове бачення цифрової охорони здоров'я.

Це позиціонує поточний проект як фундаментальний крок до більш інтелектуальної, прогностичної та персоналізованої системи реабілітації,

демонструючи потенціал значного впливу на догляд за пацієнтами та операційну ефективність за межами початкового обсягу. Це підвищує академічну цінність, демонструючи розуміння технологічних дорожніх карт та їхніх клінічних наслідків.

Розширена функціональність:

- **Інтеграція з існуючими медичними системами:** Розробка API для безперешкодного обміну даними з електронними медичними картами (EHR) або системами управління практикою (PMS) [24].
- **Розширені можливості виставлення рахунків:** Впровадження більш складних функцій для управління медичними рахунками, страховими претензіями та онлайн-платежами [3].
- **Розробка мобільних додатків:** Створення нативних мобільних додатків для iOS та Android для покращення доступності та користувацького досвіду пацієнтів, а також для використання можливостей мобільних пристроїв (наприклад, датчиків).
- **Інтеграція з IoT-пристроями:** Підключення до пристроїв Інтернету речей для віддаленого моніторингу життєвих показників або фізичної активності пацієнтів.
- **Покращення масштабованості:** Для дуже великих розгортань можуть бути розглянуті подальші архітектурні вдосконалення [13], такі як шардування бази даних (розподіл даних між кількома серверами) або використання балансувальників навантаження для розподілу трафіку між кількома екземплярами додатка.

Ці напрямки розвитку підкреслюють потенціал системи до еволюції та адаптації до зростаючих потреб реабілітаційного центру та постійно мінливого ландшафту охорони здоров'я.

ВИСНОВКИ

Під час даної роботи було створено функціональну веб-систему для центру реабілітації пацієнтів на базі Django (Python), Bootstrap та PostgreSQL.

Проведено системний аналіз предметної області та огляд існуючих платформ (Physitrack, RehabGuru, MedBridge), що дозволило сформувати детальні вимоги до системи. Обґрунтовано вибір технологічного стеку, який забезпечує швидку розробку, високу безпеку та адаптивний інтерфейс.

Спроектовано модульну архітектуру системи на основі MVT-шаблону Django та схему бази даних PostgreSQL. Реалізовано ключові функціональні модулі: реєстрацію пацієнтів, планування прийомів, призначення вправ та відстеження прогресу з особливою увагою до безпеки медичних даних.

Комплексне тестування підтвердило стабільність, функціональність та відповідність системи вимогам. Аналіз ефективності показав потенціал для значного покращення роботи реабілітаційних центрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An Overview of the Physical Rehabilitation Process: Eternal Hospital [Електронний ресурс] – Режим доступу: https://www.eternalhospital.com/title/physical-rehabilitation-process
2. Scalable Django Architecture: Key Best Practices - Bluetick Consultants Inc. [Електронний ресурс] – Режим доступу: https://www.bluetickconsultants.com/building-a-scalable-and-maintainable-architecture-for-large-scale-django-projects/
3. Tips to Improve Rehab Practice Management - PatientCalls.com [Електронний ресурс] – Режим доступу: https://www.patientcalls.com/blog/rehab-practice-management/
4. Telehealth for physical therapy - Physitrack [Електронний ресурс] – Режим доступу: https://www.physitrack.com/telehealth
5. Exercises, Telehealth, Treatment Notes, Patient Forms, Bookings [Електронний ресурс] – Режим доступу: https://www.rehabguru.com/features
6. Medbridge: Healthcare Education and Patient Care Software [Електронний ресурс] – Режим доступу: https://www.medbridge.com/
7. Top 14 Pros of Using Django for Python Web Development [Електронний ресурс] – Режим доступу: https://djangostars.com/blog/top-14-pros-using-django-web-development/
8. Django Web Development: Why Use This Python Framework - Inoxoft [Електронний ресурс] – Режим доступу: https://inoxoft.com/blog/10-advantages-of-using-django-for-web-development/
9. Bootstrap Framework Guide | Web Development | Responsive Design - QuickStart [Електронний ресурс] – Режим доступу: https://www.quickstart.com/blog/creative-and-design/everything-you-need-to-know-about-the-bootstrap-framework/

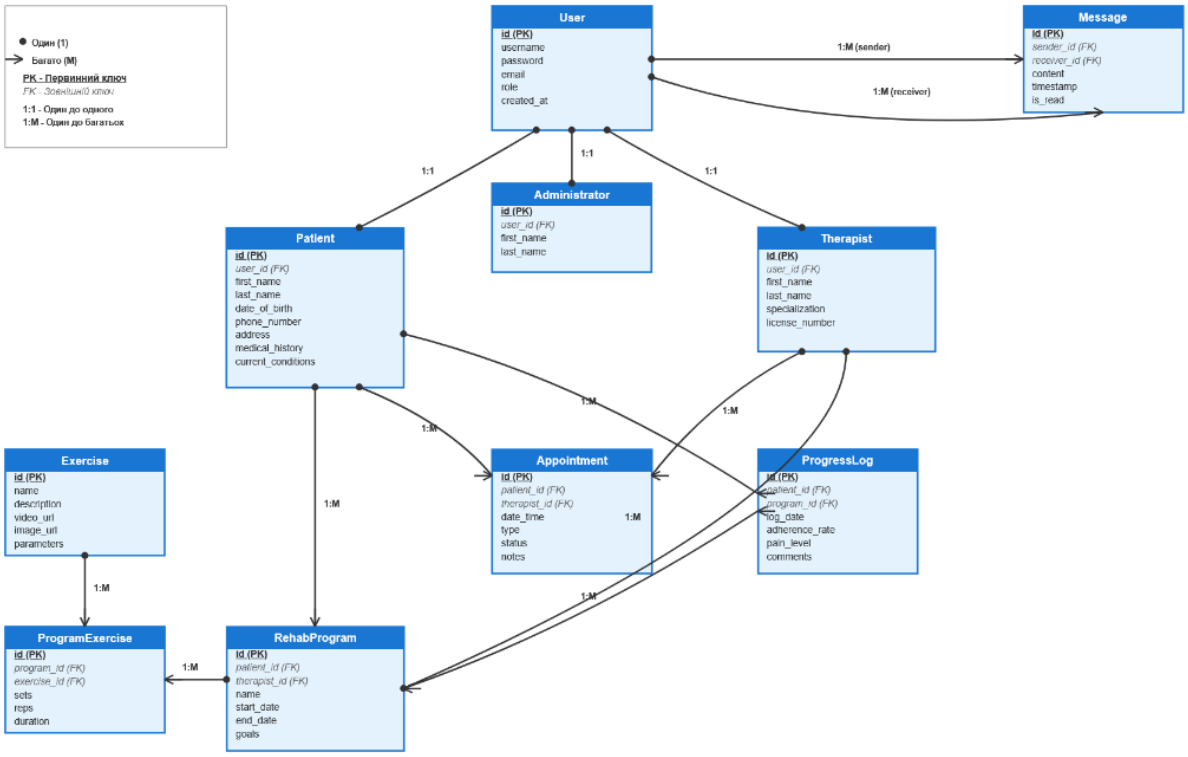
10. What Is PostgreSQL And Everything You Need To Know - Techvify [Електронний ресурс] – Режим доступу: https://techvify.com/what-is-postgresql/
11. What is PostgreSQL? Features and Benefits - Quest Software [Електронний ресурс] – Режим доступу: https://www.quest.com/learn/what-is-postgresql.aspx
12. Functional Requirements: Examples and Templates - Intellisoft [Електронний ресурс] – Режим доступу: https://intellisoft.io/functional-requirements/
13. Non-Functional Requirements: Tips, Tools, and Examples | Perforce Software [Електронний ресурс] – Режим доступу: https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples
14. Django Project MVT Structure - GeeksforGeeks [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/django-project-mvt-structure/
15. PostgreSQL specific model fields - Django documentation [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/5.2/ref/contrib/postgres/fields/
16. Best Practices for Django Web Application Security - Coders.dev [Електронний ресурс] – Режим доступу: https://www.coders.dev/blog/best-practices-for-django-web-app-security.html
17. Documentation: 17: 18.8. Encryption Options - PostgreSQL [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/current/encryption-options.html
18. django-pgcrypto-fields - PyPI [Електронний ресурс] – Режим доступу: https://pypi.org/project/django-pgcrypto-fields/
19. Testing in Django [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/5.2/topics/testing/

20. How to Write Test Cases: A Step-by-Step QA Guide | Coursera [Електронний ресурс] – Режим доступу: https://www.coursera.org/articles/how-to-write-test-cases
21. What is User Acceptance Testing? - BrowserStack [Електронний ресурс] – Режим доступу: https://www.browserstack.com/guide/user-acceptance-testing
22. Healthcare Software Testing Services - Empeek [Електронний ресурс] – Режим доступу: https://empeek.com/qa-testing/healthcare-software-testing/
23. Python in Healthcare: Medical Field & Research Applications — Blog Evrone [Електронний ресурс] – Режим доступу: https://evrone.com/blog/python-healthcare
24. Django ORM — Examples and Practice Problems - In Plain English [Електронний ресурс] – Режим доступу: https://plainenglish.io/blog/django-orm-examples-and-practice-problems
25. Migrations - Django documentation [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/5.2/topics/migrations/
26. How To Integrate Django with Bootstrap for Responsive Web Design - Learn Tube [Електронний ресурс] – Режим доступу: https://learntube.ai/blog/programming/django/how-to-integrate-django-with-bootstrap-for-responsive-web-design/
27. Database Design: What HIM Professionals Need to Know - PMC - PubMed Central [Електронний ресурс] – Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC2047327/
28. Django Migrations: A Primer - Real Python [Електронний ресурс] – Режим доступу: https://realpython.com/django-migrations-a-primer/
29. 10 Django Best Practices You Need to Know in 2025 - Moon Technolabs [Електронний ресурс] – Режим доступу: https://www.moontechnolabs.com/blog/django-best-practices/

30. Applications - Django documentation [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/5.2/ref/applications/

ДОДАТКИ

ДОДАТОК А. Діаграма "Сутність-Зв'язок" (ERD) системи.



ДОДАТОК Б.Таблиця функціонального тестування

ID тесту	**Функціональність**	**Очікуваний результат**	**Фактичний результат**	**Статус**
TST-001	Реєстрація пацієнта	Користувач створюється, перенаправлення в профіль	Відповідає	Пройдено
TST-002	Авторизація з правильними даними	Вхід у систему, доступ до кабінету	Відповідає	Пройдено
TST-003	Авторизація з помилковими даними	Повідомлення про помилку автентифікації	Відповідає	Пройдено
TST-004	Додавання нового пацієнта терапевтом	Дані зберігаються, відображаються в списку	Відповідає	Пройдено

TST-005	Призначення вправ пацієнту	Збереження програми, доступна для перегляду	Відповідає	Пройдено
TST-006	Обмеження доступу неавторизованим	Редірект на сторінку входу	Відповідає	Пройдено
TST-007	Відправлення повідомлення у чаті	Повідомлення зберігається, з’являється в інтерфейсі	Відповідає	Пройдено

Додаток В. Головна сторінка