Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Веб-орієнтована програмна система для медсестер будинку літніх людей
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-6

Романенко П.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Голян Н.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Романенко Поліні Валентинівні
(прізвище, ім’я, по батькові)

1. Тема роботи Веб-орієнтована програмна система для медсестер будинку літніх людей

Затверджена наказом по університету від 20.05.2024 №471Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи Веб-орієнтована програмна система для медсестер будинку літніх людей спрощує ведення пацієнтів, введення ліків та координацію роботи медсестер. Розроблена з використанням стеку MERN, вона дозволяє лікарям реєструвати медсестер, керувати записами пацієнтів та контролювати процедури.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	09.04.2024	*виконано*
2	Створення специфікації ПЗ	15.04.2024	*виконано*
3	Проектування ПЗ	20.04.2024	*виконано*
4	Розробка ПЗ	02.06.2024	*виконано*
5	Тестування ПЗ	03.06.2024	*виконано*
6	Оформлення пояснювальної записки	05.06.2024	*виконано*
7	Підготовка презентації та доповіді	15.06.2024	*виконано*
8	Попередній захист	16.06.2024	*виконано*
9	Нормоконтроль, рецензування	16.06.2024	*виконано*
10	Здача роботи у електронний архів	17.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	18.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка) _____ Романенко П.В.
(підпис)

Керівник роботи _____ доц. Голян Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 95 стор., 19 рис., 17 джерел.

БУДИНОК ЛІТНІХ ЛЮДЕЙ, БУДИНОК ПРЕСТАРІЛИХ, ВЕБ-ОРІЄНТОВАНА ПРОГРАМНА СИСТЕМА, ДОКТОР, ЛІКИ, МЕДСЕСТРИ, ПАЦІЄНТИ, ПРОГРАМНА СИСТЕМА, ПРОЦЕДУРИ, EXPRESSJS, MONGODB, NODEJS, REACTJS.

Об’єкт розробки – веб-орієнтована програмна система для медсестер будинку літніх людей. Ця система спрямована на оптимізацію управління пацієнтами, медсестрами, ліками та процедурами в умовах будинку престарілих. Вона надає адміністратору (лікарю) функціональні можливості для реєстрації нових медсестер, додавання ліків і процедур, управління записами пацієнтів і призначення медсестер до пацієнтів. Медсестри можуть входити в систему для перегляду інформації про закріплених за ними пацієнтів, а також мають доступ до повного списку всіх пацієнтів, ліків і процедур.

Мета розробки – створення ефективної та зручної веб-системи, яка покращить управління медсестрами та пацієнтами в будинку престарілих. Система повинна забезпечити легку реєстрацію та управління медсестрами, а також спрощений доступ до інформації про пацієнтів, ліків та процедур. Це має підвищити загальну ефективність роботи будинку для людей похилого віку та сприяти кращому догляду за пацієнтами.

Метод рішення – стек MERN, який включає MongoDB, Express.js, React.js та Node.js, а також середовище розробки Visual Studio Code. MongoDB використовується як база даних для зберігання інформації про пацієнтів, медсестер, ліки та процедури. Express.js використовується як фреймворк веб-додатків для обробки HTTP-запитів та маршрутів. React.js використовується для розробки фронтенду для створення зручного інтерфейсу для взаємодії з системою.

Node.js використовується як внутрішнє середовище виконання для запуску серверної логіки та управління загальним робочим процесом додатку.

У результаті розробки було створено комплексну веб-орієнтовану програмну систему для медсестер будинку літніх людей. Ця система надає адміністратору можливість ефективно управляти медсестрами, пацієнтами, ліками та процедурами. Медсестри мають доступ до зручного інтерфейсу, де вони можуть переглядати інформацію про закріплених за ними пацієнтів, а також мають доступ до повного списку всіх пацієнтів, ліків та процедур. Загалом, система покращує управління медсестрами та пацієнтами в будинку для літніх людей, підвищує операційну ефективність та якість догляду за пацієнтами.

DOCTOR, EXPRESSJS, MEDICINES, MONGODB, NODEJS, NURSES, NURSING HOME, PATIENTS, PROCEDURES, REACTJS, SOFTWARE SYSTEM, WEB-ORIENTED SOFTWARE SYSTEM.

The object of development is a web-based software system for nurses in a nursing home. This system is aimed at optimizing the management of patients, nurses, medications and procedures in a nursing home. It provides the administrator (doctor) with the functionality to register new nurses, add medications and procedures, manage patient records, and assign nurses to patients. Nurses can log in to view information about their assigned patients and have access to a complete list of all patients, medications and procedures.

The goal of the development is to create a convenient and efficient web-based system that will improve the management of nurses and patients in a nursing home. The system should provide easy registration and management of nurses, as well as simplified access to information about patients, medications and procedures. This should increase the overall efficiency of the nursing home and contribute to better patient care.

The solution is based on the MERN stack, which includes MongoDB, Express.js, React.js, and Node.js, as well as the Visual Studio Code development environment. MongoDB is used as a database to store information about patients, nurses, medications,

and procedures. Express.js is used as a web application framework for processing HTTP requests and routes. React.js is used for frontend development to create a user-friendly interface for interacting with the system. Node.js is used as an internal runtime environment to run server logic and manage the overall workflow of the application.

As a result of the development, a comprehensive web-oriented software system for nurses of a nursing home was created. This system provides the administrator with the ability to effectively manage nurses, patients, medications, and procedures. Nurses have access to a user-friendly interface where they can view information about the patients assigned to them, and have access to a complete list of all patients, medications, and procedures. Overall, the system improves nursing and patient management in the nursing home, increasing operational efficiency and the quality of patient care.

Я, Романенко Поліна Валентинівна, студентка гр. ПЗПІ-20-6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Веб-орієнтована програмна система для медсестер будинку літніх людей», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасних системах охорони здоров’я ефективне управління даними пацієнтів, медикаментами та процедурами має вирішальне значення для забезпечення якісного надання медичної допомоги, особливо в таких установах, як будинки престарілих, де необхідно ретельно керувати безліччю факторів для задоволення різноманітних потреб пацієнтів. Тому розробка веб-орієнтованої програмної системи, створеної спеціально для медсестер в будинках для людей похилого віку, стає першочерговою потребою. Ця бакалаврська робота має на меті заглибитися в концептуалізацію, проектування та реалізацію такої системи, визнаючи її потенціал для зміни в управлінні доглядом за пацієнтами в будинках літніх людей.

Будинки для людей похилого віку слугують основним місцем проживання для осіб, які потребують спеціалізованого догляду через похилий вік, хронічні захворювання, інвалідність або інші стани здоров’я. Оскільки населення світу продовжує старіти, попит на послуги будинків престарілих зростає, що збільшує навантаження на медичні заклади, які повинні надавати комплексний, персоналізований догляд мешканцям. Однак ручне управління даними про пацієнтів, ліками та процедурами в таких закладах часто призводить до неефективності, помилок та упущень у наданні допомоги. Впроваджуючи веб-орієнтовану програмну систему, пристосовану до конкретних потреб медсестер у будинках для людей похилого віку, ця кваліфікаційна робота прагне вирішити ці проблеми та впорядкувати надання допомоги, підвищуючи якість життя мешканців.

Основна мета цієї роботи – спроектувати та розробити веб-орієнтовану програмну систему, яка дозволить медсестрам у будинках для людей похилого віку ефективно управляти даними пацієнтів, ліками та процедурами. Використовуючи сучасні веб-технології, запропонована система має на меті покращити комунікацію, координацію та прийняття рішень серед медичних працівників, тим самим оптимізуючи надання допомоги та покращуючи результати лікування пацієнтів. Крім того, система має на меті зменшити адміністративне навантаження

на медперсонал, що дозволить йому зосередити більше часу та уваги на безпосередньому догляді за пацієнтами.

Для цієї мети, проектування та розробка веб-орієнтованої програмної системи буде спрямована на досягнення наступних цілей:

проведення комплексної оцінки потреб і проблем, з якими стикаються медсестри в будинках престарілих щодо управління даними про пацієнтів, введення ліків і процедурної документації;

використання інформації, яка отримана в результаті аналізу потреб, для розробки зручної, інтуїтивно зрозумілої веб-орієнтованої програмної системи, яка відповідає виявленим потребам і викликам, забезпечуючи масштабованість, безпеку та інтероперабельність;

розробка програмної системи відповідно до встановлених проектних специфікацій з використанням надійних мов програмування, фреймворків та методологій розробки для забезпечення функціональності та надійності;

проведення тестування, щоб підтвердити продуктивність, зручність та ефективність розробленої системи.

Результати цієї роботи мають широке застосування в галузі медичної інформатики, зокрема в управлінні закладами довготривалого догляду, такими як будинки престарілих. Розроблена веб-орієнтована програмна система пропонує рішення для ефективного управління даними пацієнтів, адміністрування ліків, процедурної документації та координації роботи персоналу. Покращуючи комунікацію, оптимізуючи робочі процеси та підвищуючи якість обслуговування, система має потенціал змінити роботу будинків для людей похилого віку та покращити результати для пацієнтів.

Отже, враховуючи специфічні потреби та проблеми цих закладів, веб-орієнтована програмна система для медсестер будинку літніх людей покращує комунікацію, оптимізує робочі процеси та підвищує якість догляду.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Предметна галузь розробки веб-орієнтованої програмної системи для медсестер в будинку престарілих передбачає комплексний аналіз різних аспектів, включаючи потреби медичних працівників і пацієнтів, існуючі проблеми в закладах довгострокового догляду, нормативні вимоги, технологічні міркування і потенційний вплив на надання медичної допомоги. Нижче наведено детальний аналіз кожного аспекту.

Потреби медичних працівників та пацієнтів:

- медсестри потребують системи, яка полегшує ефективне управління даними про пацієнтів, включаючи запис історії хвороби, ліків, лікування та планів догляду;
- лікарям, як адміністраторам, потрібні інструменти для контролю та управління персоналом, пацієнтами, ліками та процедурами, забезпечення дотримання нормативних вимог та оптимізації надання медичної допомоги;
- пацієнти та їхні сім’ї потребують персоналізованих планів догляду, своєчасного прийому ліків та ефективної комунікації з медичними працівниками.

Існуючі проблеми у закладах довготривалого догляду:

- ручний паперовий документообіг призводить до неефективності, помилок і затримок у документуванні, комунікації та координації між медичними працівниками;
- відсутність централізованих інформаційних систем призводить до фрагментарності записів про пацієнтів, що перешкоджає безперервності догляду та прийняттю рішень;
- нормативні вимоги щодо конфіденційності даних, управління медикаментами та забезпечення якості накладають додатковий тягар на персонал будинків для людей похилого віку.

Нормативні вимоги:

- дотримання нормативно-правових актів у сфері охорони здоров’я, що мають важливе значення для забезпечення конфіденційності та безпеки даних пацієнтів;
- практика управління медикаментозним забезпеченням повинна відповідати регуляторним стандартам, щоб мінімізувати помилки, несприятливі події та юридичні зобов’язання;
- документування процедур і планів лікування необхідне для регуляторних аудитів, акредитації та ініціатив з покращення якості.

Технологічні вимоги:

- веб-орієнтована архітектура забезпечує доступ до програмної системи з будь-якого пристрою з доступом до Інтернету, що полегшує віддалене та мобільне використання медсестрами та лікарями;
- масштабованість та інтероперабельність є критично важливими для задоволення зростаючих потреб будинків престарілих, інтеграції з існуючою IT-інфраструктурою охорони здоров’я та обміну даними із зовнішніми системами;
- дизайн користувацького інтерфейсу повинен надавати пріоритет зручності, доступності та інтуїтивно зрозумілій навігації для користувачів з різним рівнем технічної підготовки.

Потенційний вплив на надання медичної допомоги:

- впровадження веб-орієнтованої програмної системи може спростити адміністративні завдання, зменшити кількість помилок у документації та покращити комунікацію та координацію між медичними працівниками;
- доступ у режимі реального часу до даних пацієнтів, ліків та процедур покращує прийняття клінічних рішень, безпеку ліків та безперервність надання медичної допомоги, що призводить до покращення результатів лікування пацієнтів;
- система дає можливість медсестрам і лікарям надавати особистісно-орієнтовану допомогу, сприяти залученню пацієнтів, будувати довіру і

взаєморозуміння з пацієнтами та їхніми сім’ями.

В епоху цифрової трансформації заклади охорони здоров’я мають доступ до безлічі програмних рішень, розроблених для задоволення їхніх конкретних потреб і вимог. Від оптимізації управління персоналом до покращення клінічної допомоги та забезпечення відповідності нормативним вимогам, такі передові платформи управління, як NurseCare, PointClickCare, MatrixCare, SmartLinx та AL Advantage дають змогу медичним організаціям легше та ефективніше орієнтуватися в складнощах галузі. Кожна з цих платформ пропонує унікальні функції та можливості, пристосовані до різноманітних потреб постачальників медичних послуг у різних умовах, включаючи лікарні, будинки престарілих, будинки для людей похилого віку та агенції з догляду на дому.

NurseCare поєднує систему виклику медсестер з функціями управління охороною здоров’я та веденням медсестринської документації в одному універсальному пристрої (див. рис. 1.1).

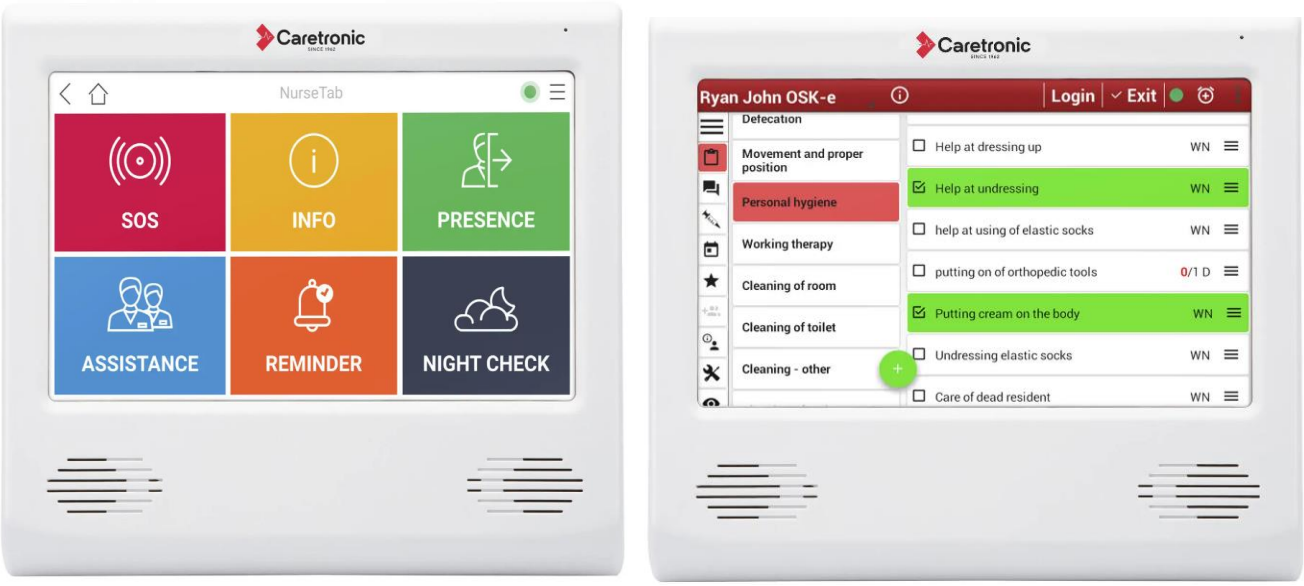


Рисунок 1.1 – NurseCare

Переваги NurseCare:

– NurseCare забезпечує безпеку як медичного персоналу, так і пацієнтів, надаючи надійну та оперативну систему виклику медсестер. Завдяки

таким функціям, як екстрені сповіщення та можливості зв’язку в режимі реального часу, NurseCare сприяє швидкому реагуванню на потреби пацієнтів та надзвичайні ситуації, мінімізуючи ризики та сприяючи створенню більш безпечного медичного середовища;

- автоматизуючи рутинні адміністративні завдання та впорядковуючи робочі процеси, NurseCare зменшує навантаження на медсестер та доглядальниць, дозволяючи їм зосередити більше часу та уваги на безпосередньому догляді за пацієнтами. Це не лише знижує рівень стресу та вигорання серед медичного персоналу, але й підвищує рівень задоволеності роботою та утримання кадрів;
- NurseCare оптимізує робочі процеси в медичних установах шляхом оцифрування та автоматизації різних завдань, таких як ведення документації пацієнтів та розподіл ресурсів. Це призводить до підвищення ефективності, продуктивності та використання ресурсів, що в кінцевому підсумку покращує загальну операційну діяльність медичних закладів;
- інтегруючи функціональність системи виклику медсестер з управлінням охороною здоров’я та медсестринською документацією в одному пристрої, NurseCare пропонує економічно ефективне рішення для медичних установ. Система допомагає оптимізувати витрати, зменшуючи потребу в декількох автономних системах і ручних процесах, що призводить до довгострокової економії коштів і поліпшення фінансової стійкості;
- завдяки передовим IP-технологіям та широкому функціоналу NurseCare сприяє підвищенню якості та ефективності охорони здоров’я. Оцифровуючи робочі процеси, оптимізуючи розподіл ресурсів і сприяючи комунікації та співпраці між медичними працівниками в режимі реального часу, NurseCare покращує результати лікування пацієнтів, рівень їхньої задоволеності та загальний рівень надання медичних послуг.

Недоліки NurseCare:

- розгортання NurseCare може викликати певні труднощі на початковому

етапі впровадження, включаючи інтеграцію системи з існуючою інфраструктурою, вимоги до навчання персоналу та коригування робочого процесу. Заклади охорони здоров'я повинні інвестувати час і ресурси, щоб забезпечити плавний перехід на нову систему і пом'якшити потенційні перебої в роботі;

– передові IP-технології та багатофункціональні можливості NurseCare можуть становити технічну складність для деяких користувачів, особливо для тих, хто має обмежені технологічні навички. Належне навчання та постійна технічна підтримка мають важливе значення для забезпечення ефективного використання системи та мінімізації проблем, пов'язаних з користувачами.

Отже, NurseCare є значним досягненням в галузі технологій охорони здоров'я, пропонуючи комплексне рішення для цифровізації процесів адміністрування та управління охороною здоров'я.

PointClickCare – це комплексна хмарна платформа, яка призначена для оптимізації управління медичними послугами для постачальників кваліфікованої допомоги, включаючи будинки престарілих, будинки для людей з обмеженими можливостями та центри догляду за людьми похилого віку (див. рис. 1.2).

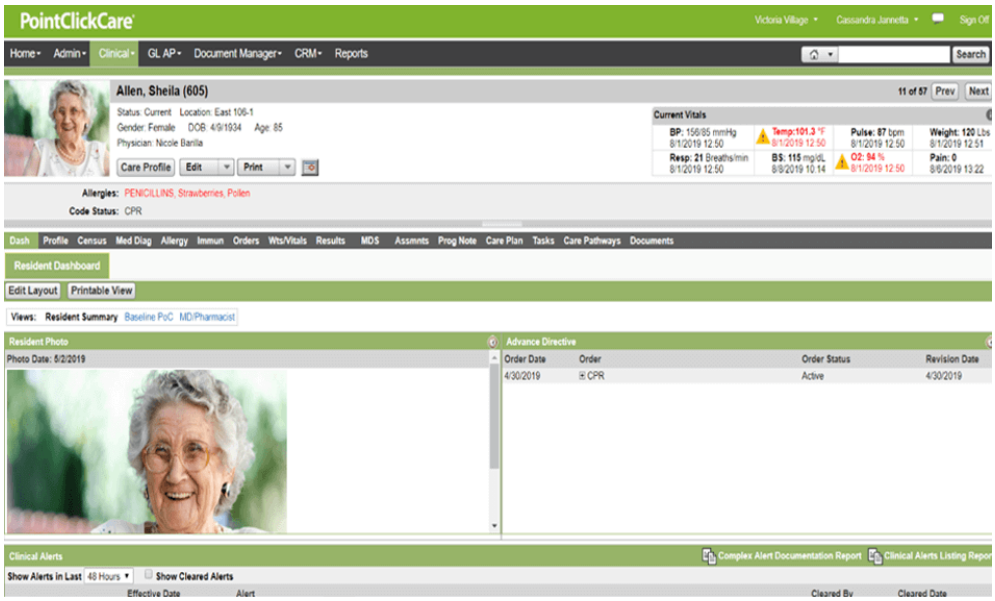


Рисунок 1.2 – PointClickCare

Переваги PointClickCare:

- PointClickCare допомагає постачальникам послуг оптимізувати цикли отримання доходів, покращити якість перепису та підвищити операційну ефективність. Завдяки таким функціям, як спрощення процесів виставлення рахунків, точне відстеження відшкодування та фінансова аналітика, постачальники можуть максимізувати доходи та мінімізувати фінансові ризики;
- забезпечуючи співпрацю між командами та оптимізуючи клінічні процеси, PointClickCare дає змогу первинному персоналу надавати послідовну та якісну медичну допомогу. Співробітники отримують доступ до глибокої клінічної інформації, протоколів, заснованих на доказах, та найкращих робочих процесів;
- PointClickCare сприяє впровадженню протоколів, заснованих на доказах, та найкращих робочих процесів для підтримки послідовної, всебічної та відповідної вимогам документації. Постачальники можуть забезпечити дотримання регуляторних вимог, зменшити ризики невідповідності та підвищити якість надання медичної допомоги;
- PointClickCare дозволяє постачальникам послуг узгоджувати клінічні потреби пацієнтів з можливостями закладу ще до госпіталізації, забезпечуючи бажані результати та перевершуючи очікування від лікування. Використовуючи дані та вподобання пацієнтів, лікарі можуть персоналізувати плани догляду, підвищити рівень задоволеності пацієнтів та покращити загальний досвід лікування;
- за допомогою PointClickCare лікарі можуть ефективно збирати та використовувати дані для покращення координації медичної допомоги в межах своєї мережі. Демонструючи якість результатів і тісно співпрацюючи з лікарнями, організаціями підзвітної медичної допомоги та іншими мережевими партнерами, постачальники можуть стати кращим вибором для направлень і партнерства.

Недоліки PointClickCare:

– впровадження PointClickCare може спричинити проблеми, пов’язані з інтеграцією системи, міграцією даних та навчанням персоналу. Постачальники медичних послуг повинні інвестувати час і ресурси, щоб забезпечити плавний перехід на нову платформу і пом’якшити потенційні перебої в роботі;
– PointClickCare покладається на хмарні технології, що може спричинити залежність від підключення до Інтернету та підтримки постачальника. Постачальники повинні забезпечити надійний доступ до Інтернету та мати плани на випадок непередбачуваних ситуацій на випадок простою системи або технічних проблем.

Отже, PointClickCare є потужним інструментом для кваліфікованих медичних працівників, які прагнуть оптимізувати управління охороною здоров’я та покращити результати лікування пацієнтів.

MatrixCare – провідний постачальник інтуїтивно зрозумілих програмних рішень, розроблених для підвищення ефективності, залучення персоналу та покращення якості обслуговування в різних медичних установах. Від закладів довгострокового догляду до спільнот для людей похилого віку, спільнот життєвого плану, агентств домашнього догляду, хоспісів та приватних постачальників послуг з догляду, MatrixCare пропонує комплексні хмарні рішення, розроблені з урахуванням унікальних потреб кожного сегмента охорони здоров’я (див. рис. 1.3).

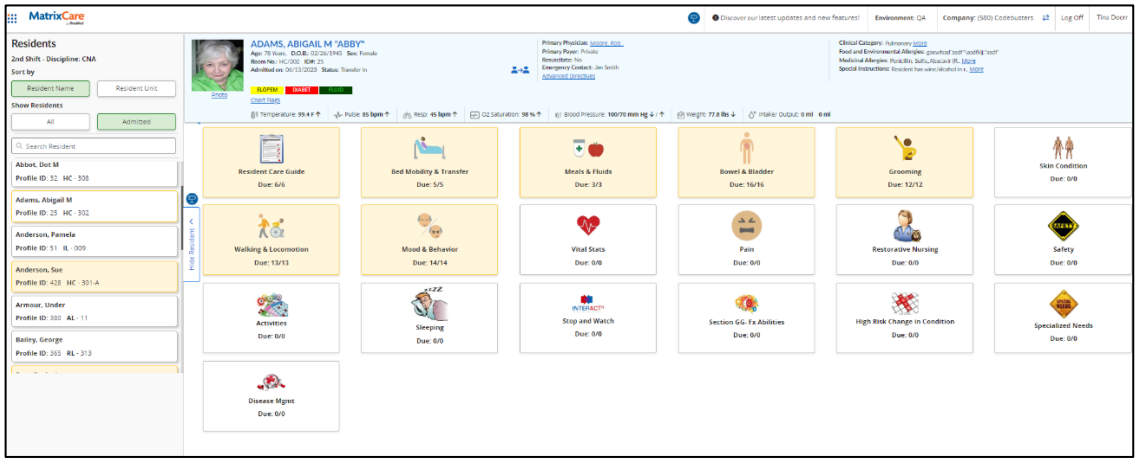


Рисунок 1.3 – MatrixCare

Переваги MatrixCare:

- MatrixCare надає постачальникам медичних послуг інноваційні інструменти та інтуїтивно зрозумілі програмні рішення, які оптимізують робочі процеси та підвищують залученість персоналу. Автоматизуючи адміністративні завдання, спрощуючи процедури документування та надаючи доступ до необхідної інформації в режимі реального часу, MatrixCare дозволяє персоналу працювати більш ефективно та зосередитися на наданні високоякісної допомоги пацієнтам;
- у закладах довготривалого догляду та спільнотах для людей похилого віку MatrixCare допомагає оптимізувати відшкодування та управління фінансами, надаючи комплексні рішення для виставлення рахунків, управління циклом надходжень та дотримання нормативних вимог. Завдяки таким функціям, як точна документація, автоматизована обробка заяв та аналітика доходів, MatrixCare дозволяє постачальникам послуг максимізувати доходи та забезпечити фінансову стійкість;
- MatrixCare підтримує досконаліть клінічної допомоги, пропонуючи інструменти та функціональні можливості, які покращують прийняття клінічних рішень, координацію допомоги та вимірювання результатів. Від електронних медичних записів до планування лікування та управління медикаментами, MatrixCare надає медичним працівникам можливість надавати персоналізовану, науково обґрунтовану медичну допомогу, яка відповідає унікальним потребам кожного пацієнта;
- у домашніх умовах, хоспісах та приватних службах догляду за хворими MatrixCare забезпечує безперебійну координацію догляду на всьому шляху пацієнта. Завдяки інтегрованим рішенням для планування, документації візитів, дистанційного моніторингу та комунікації, MatrixCare гарантує, що пацієнти отримують своєчасну та скоординовану допомогу від мультидисциплінарних команд, що призводить до покращення результатів та задоволеності пацієнтів;
- MatrixCare пропонує гнучкі та масштабовані рішення, які адаптуються до

мінливих потреб медичних працівників, також може бути адаптована до конкретних вимог кожного сегмента охорони здоров'я, що дозволяє постачальникам зростати в динамічному медичному середовищі.

Недоліки MatrixCare:

– впровадження MatrixCare може передбачати авансові витрати, пов'язані з ліцензуванням програмного забезпечення, впровадженням та навчанням. Постачальники медичних послуг повинні оцінити свої бюджетні обмеження та визначити довгострокову рентабельність інвестицій від впровадження рішень MatrixCare;
– інтеграція MatrixCare з існуючою IT-інфраструктурою та системами охорони здоров'я може створювати технічні проблеми, включаючи міграцію даних, проблеми інтероперабельності та сумісності систем. Постачальники медичних послуг повинні планувати та реалізовувати надійну стратегію інтеграції, щоб забезпечити безперебійне підключення та обмін даними між системами.

Отже, MatrixCare є універсальним і комплексним рішенням для постачальників медичних послуг, які прагнуть підвищити ефективність, поліпшити якість обслуговування та досягти кращих результатів у різних медичних установах.

SmartLinx – провідний постачальник рішень для управління персоналом, спеціально розроблених спеціально для закладів довготривалого догляду, допомагаючи з плануванням, відстеженням часу та відвідування, а також дотриманням нормативних вимог (див. рис. 1.4).

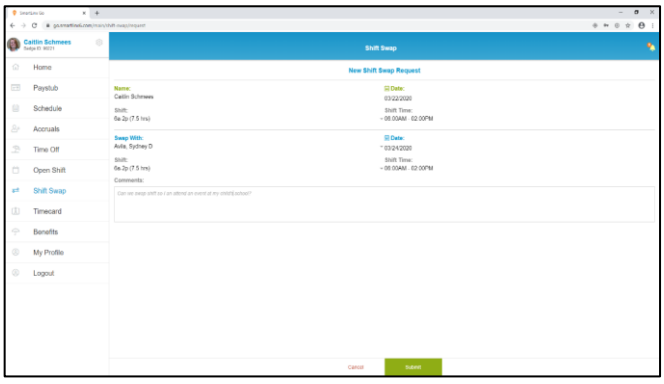


Рисунок 1.4 – SmartLinx

Переваги SmartLinx:

- SmartLinx дозволяє організаціям охорони здоров’я оптимізувати процеси планування та укомплектування штату завдяки розширеним функціям управління персоналом. Завдяки таким функціям, як автоматизоване планування, управління змінами та відстеження доступності та кваліфікації персоналу в режимі реального часу, SmartLinx допомагає забезпечити достатній рівень укомплектованості штату, мінімізуючи витрати на понаднормову роботу та дефіцит робочої сили;
- SmartLinx допомагає медичним організаціям підтримувати відповідність галузевим нормам і трудовому законодавству шляхом автоматизації моніторингу та звітності. Від відстеження сертифікатів та облікових даних співробітників до управління обліком робочого часу та відвідування, SmartLinx забезпечує дотримання нормативних вимог, зменшує ризики дотримання вимог та мінімізує потенційні штрафи та пені;
- надаючи працівникам доступ до інструментів самообслуговування, мобільних додатків та каналів зв’язку, SmartLinx підвищує залученість та задоволеність працівників. Співробітники можуть переглядати свій графік, запитувати відгули та безперешкодно спілкуватися з менеджерами та колегами, що призводить до підвищення морального духу, продуктивності та рівня утримання персоналу;
- SmartLinx спрощує процеси розрахунку заробітної плати та виставлення рахунків шляхом інтеграції даних управління персоналом з системами розрахунку заробітної плати та програмним забезпеченням для виставлення рахунків. Ця інтеграція забезпечує точний розрахунок заробітної плати, своєчасне подання рахунків та ефективне управління циклом надходжень, зменшуючи адміністративні накладні витрати та покращуючи фінансові показники;
- SmartLinx пропонує надійні можливості звітності та аналітики, які дозволяють медичним організаціям приймати рішення на основі даних та оптимізувати стратегії управління персоналом. Аналізуючи ключові

показники ефективності, такі як витрати на оплату праці, показники продуктивності та коефіцієнти укомплектованості штату, організації можуть виявляти тенденції, прогнозувати кадрові потреби та ефективно розподіляти ресурси.

Недоліки SmartLinx:

– впровадження SmartLinx може вимагати значного часу, ресурсів та зусиль з управління організаційними змінами. Організаціям охорони здоров'я необхідно інвестувати в навчання та підтримку, щоб забезпечити безперешкодне впровадження програмного забезпечення та пом'якшити потенційні перебої в роботі;
– інтеграція SmartLinx з існуючими IT-системами охорони здоров'я, такими як електронні медичні картки та білінгові системи, може створювати технічні проблеми. Організаціям охорони здоров'я необхідно спланувати та реалізувати комплексну стратегію інтеграції, щоб забезпечити безперебійний обмін даними та функціональну сумісність між системами.

Отже, SmartLinx пропонує комплексне рішення для організацій охорони здоров'я, які прагнуть оптимізувати управління персоналом і підвищити операційну ефективність.

AL Advantage – це комплексна програмна платформа, спеціально розроблена для будинків для людей з обмеженими фізичними можливостями, AL Advantage надає інструменти для оцінки стану мешканців, планування догляду та управління медикаментозним лікуванням (див. рис. 1.5).

Переваги AL Advantage:

– AL Advantage надає пансіонатам централізовану платформу для управління інформацією про підопічних, планами догляду, графіками прийому ліків і розкладами активностей. Консолідуючи дані про пацієнтів в одній системі, AL Advantage дозволяє персоналу швидко отримувати доступ до важливої інформації, приймати обґрунтовані рішення щодо догляду та надавати підопічним персоналізований догляд;
– завдяки таким функціям, як планування змін, призначення персоналу та

управління завданнями, AL Advantage допомагає установам з підтриманого проживання оптимізувати кількість персоналу та підвищити його ефективність. Автоматизуючи рутинні адміністративні завдання та забезпечуючи видимість розкладу та навантаження персоналу в режимі реального часу, AL Advantage дозволяє керівникам ефективно розподіляти ресурси та забезпечувати адекватне покриття в будь-який час;

- AL Advantage полегшує комунікацію та співпрацю між співробітниками, мешканцями та сім'ями завдяки захищеному обміну повідомленнями, спільним календарям та цифровій документації. Надаючи централізовану платформу для обміну інформацією та оновленнями, AL Advantage сприяє прозорості, підзвітності та залученню зацікавлених сторін, що в кінцевому підсумку підвищує задоволеність мешканців та залученість сімей;
- AL Advantage підтримує установи з підтриманим проживанням у дотриманні відповідності державним і федеральним нормам, галузевим стандартам і найкращим практикам. Від документування медичних втручань та прийому ліків до проведення оцінок і складання звітів, AL Advantage допомагає забезпечити відповідність закладів нормативним вимогам і продемонструвати відповідальність за надання якісних послуг;
- пропонуючи вбудовані інструменти забезпечення якості, показники ефективності та аналітичні можливості, AL Advantage дозволяє установам з підтриманого проживання відстежувати ключові показники ефективності, виявляти тенденції та впроваджувати ініціативи з безперервного вдосконалення. Аналізуючи дані, пов'язані з результатами діяльності підопічних, продуктивністю персоналу та роботою закладу, AL Advantage дає можливість установам оптимізувати процеси та підвищити загальну продуктивність.

Недоліки AL Advantage:

- впровадження AL Advantage може передбачати авансові витрати, пов'язані з ліцензуванням програмного забезпечення, впровадженням і навчанням. Заклади з підтриманим проживанням повинні оцінити свої

бюджетні обмеження та оцінити рентабельність інвестицій від впровадження AL Advantage з точки зору підвищення операційної ефективності, задоволеності мешканців та дотримання нормативних вимог;

– навчання ефективному використанню AL Advantage може потребувати часу та ресурсів для навчання та адаптації персоналу. Заклади з підтриманим проживанням повинні виділити достатньо часу та підтримки для того, щоб співробітники навчилися користуватися програмним забезпеченням і безперешкодно інтегрувати його у свої щоденні робочі процеси.

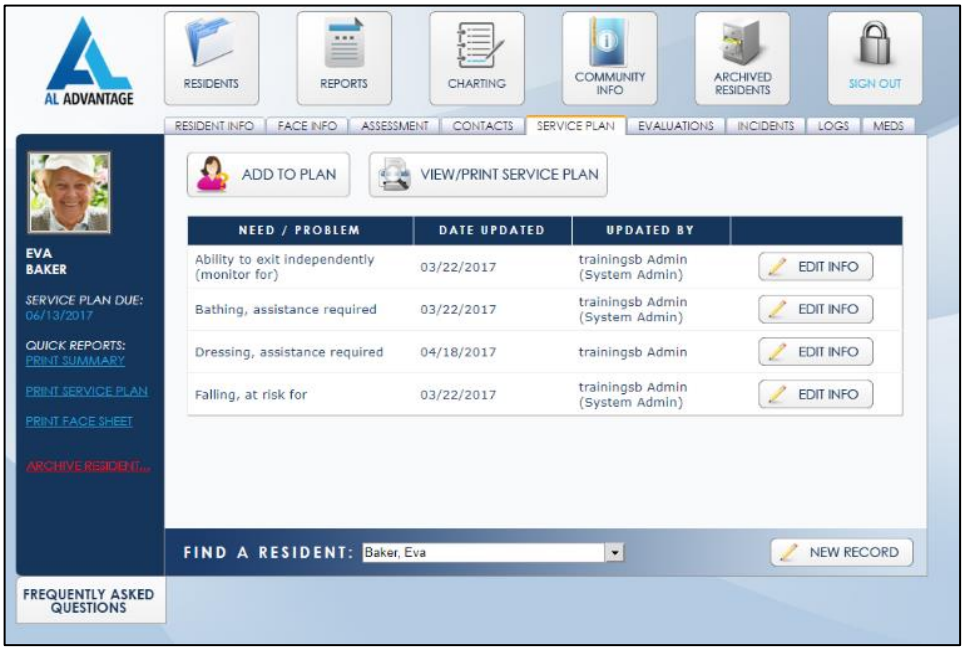


Рисунок 1.5 – AL Advantage

Отже, AL Advantage пропонує комплексне рішення для будинків для людей з обмеженими можливостями, які прагнуть покращити процеси управління, покращити догляд за пацієнтами та забезпечити відповідність нормативним вимогам.

Таким чином, аналіз предметної області підкреслює багатогранну природу розробки веб-орієнтованої програмної системи для медсестер в будинку для людей похилого віку. Враховуючи різноманітні потреби, виклики, нормативні та

технологічні вимоги, запропонована система має потенціал для трансформації практики надання допомоги, покращення результатів лікування пацієнтів та підвищення загальної якості життя в закладах довготривалого догляду. Використовуючи можливості цих передових платформ, організації охорони здоров’я можуть досягти більшої ефективності, результативності та досконалості в наданні медичної допомоги, що в кінцевому підсумку покращить благополуччя пацієнтів та мешканців у всьому континуумі надання медичної допомоги. Стає очевидним, що майбутнє управління охороною здоров’я полягає у впровадженні інновацій та технологій. Впроваджуючи ці передові рішення, медичні заклади можуть залишатися на крок попереду, адаптуватися до мінливих галузевих тенденцій і продовжувати надавати винятковий догляд тим, кого вони обслуговують. З NurseCare, PointClickCare, MatrixCare, SmartLinx та AL Advantage, які є лідерами, шлях до досконалості в управлінні охороною здоров’я вже розпочався.

1.2 Виявлення та вирішення проблем

Виявлення та вирішення проблем при розробці веб-орієнтованої програмної системи для медсестер будинку літніх людей передбачає всебічний аналіз потенційних викликів і застосування рішень, запозичених з аналогічних систем. Спираючись на такі аналоги, як NurseCare, PointClickCare, MatrixCare, SmartLinx і AL Advantage, можна визначити ключові проблемні області та ефективні стратегії для їх вирішення.

Проблема аутентифікації користувачів та контроль доступу: забезпечення безпечного доступу до системи для медсестер та адміністраторів при одночасному захисті інформації про пацієнтів від несанкціонованого доступу. Рішенням є впровадити надійні механізми автентифікації користувачів, такі як безпечні облікові дані для входу, багатофакторна автентифікація та контроль доступу на основі ролей. Використання протоколів шифрування для захисту конфіденційних даних та дотримання правил конфіденційності у сфері охорони здоров’я.

Проблема управління медикаментами та процедурами: ефективне управління медикаментами та процедурами, включаючи додавання, оновлення та видалення записів, зберігаючи при цьому точність і послідовність. Рішенням є розробити інтуїтивно зрозумілі інтерфейси для адміністраторів для додавання, редагування та видалення записів про ліки та процедури. Впровадити перевірки валідності даних для запобігання помилкам і забезпечення узгодженості в назвах та описах. Увімкнути контроль версій для відстеження змін та редагувань.

Проблема управління інформацією про пацієнтів: ефективне управління інформацією про пацієнта, включаючи демографічні дані, історію хвороби, призначену медсестру та контактні дані родичів, зберігаючи при цьому приватність і конфіденційність пацієнта. Рішенням є розробити комплексні профілі пацієнтів з полями для імені, дати народження, віку, стану здоров’я, ліків, процедур, призначеної медсестри та контактів для екстреної допомоги. Впровадити суворий контроль доступу, щоб обмежити доступ до інформації про пацієнта лише для уповноваженого персоналу. Додати функції для оновлення інформації про пацієнта та ведення точних записів.

Проблема комунікації та співпраці: сприяння комунікації та співпраці між медсестрами, адміністраторами та іншими медичними працівниками для забезпечення ефективної координації медичної допомоги. Рішенням є інтегрувати комунікаційні інструменти, такі як системи обміну повідомленнями та сповіщеннями в програмну платформу. Надати медсестрам можливість безпечно спілкуватися одна з одною та з адміністраторами щодо оновлень у догляді за пацієнтами, замовлень на ліки та процедурних інструкцій. Впровадити функції для розподілу завдань та відстеження статусу для полегшення командної роботи та підзвітності.

Проблема дотримання нормативних вимог: забезпечення відповідності нормам охорони здоров’я та галузевим стандартам в управлінні даними пацієнтів та медичними операціями. Рішенням є проводити регулярні аудити та оцінки для забезпечення дотримання регуляторних вимог. Впровадити шифрування даних,

контроль доступу та аудиторські сліди для захисту конфіденційності пацієнтів та збереження цілісності даних.

Проблема масштабованості та продуктивність системи: забезпечення можливості масштабування програмної системи відповідно до зростаючої кількості пацієнтів, медсестер, ліків та процедур, зберігаючи при цьому оптимальну продуктивність. Рішенням є розробити систему з урахуванням можливості масштабування, використовуючи модульну архітектуру та хмарну інфраструктуру для розширення. Впровадити інструменти моніторингу продуктивності для виявлення вузьких місць та оптимізації роботи системи. Регулярно оновлювати та підтримувати систему для забезпечення сумісності з новими технологіями та стандартами.

Отже, виявляючи потенційні проблеми та застосовуючи рішення, розробка веб-орієнтованої програмної системи для медсестер в будинку престарілих може забезпечувати ефективне управління доглядом за пацієнтами і медичними операціями, зберігаючи при цьому відповідність нормативним вимогам і кращим галузевим практикам.

1.3 Постановка задачі

Задача передбачає розробку веб-орієнтованої програмної системи, розробленої спеціально для медсестер, які працюють у будинках престарілих. Ця система спрямована на оптимізацію різних аспектів управління охороною здоров’я, включаючи догляд за пацієнтами, введення ліків, управління процедурами та адміністрування медсестер. Основною метою є створення зручної та ефективної платформи, яка покращує робочий процес медсестер, забезпечуючи при цьому безперебійну координацію з іншими медичними працівниками. Нижче наведені ключові особливості та функціональні можливості.

Роль адміністратора:

– адміністратор, який є лікарем, має повний контроль над системою та може виконувати такі завдання, як реєстрація нових медсестер, додавання ліків і процедур, управління записами пацієнтів і нагляд за всією роботою;

– адміністратор може додавати, редагувати або видаляти медсестер, ліки, процедури та записи пацієнтів за потреби. Він має право змінювати будь-який аспект системи, щоб пристосувати його до змін у кадровому складі, методах лікування чи потребах пацієнтів.

Управління медсестрами:

– медсестри мають персоналізовані облікові записи з обліковими даними для безпечного доступу до системи;
– медсестри можуть переглядати закріплених за ними пацієнтів, оновлювати записи про пацієнтів, документувати введення ліків і процедури;
– система дозволяє медсестрам керувати своїм графіком, переглядати призначення для пацієнтів та отримувати доступ до інформації про пацієнтів, включаючи історію хвороби, поточні ліки та необхідні процедури.

Управління пацієнтами:

– система підтримує повні профілі пацієнтів, що містять важливу інформацію, таку як ім’я, дата народження, вік, інвалідність, хронічні та невиліковні захворювання;
– медичні картки пацієнтів містять інформацію про призначені ліки, необхідні процедури, призначену медсестру та контакти родичів для екстрених випадків;
– медсестри можуть оновлювати записи пацієнтів, документувати медичні втручання та відстежувати прогрес пацієнта в системі.

Управління медикаментами та процедурами:

– ліки та процедури каталогізуються в системі з назвою та полем для опису;
– медсестри можуть отримати доступ до бази даних ліків і процедур, переглянути описи, а також вводити ліки або виконувати процедури, призначені медичними працівниками;
– система надає сповіщення та нагадування про графік прийому ліків і забезпечує дотримання протоколів прийому ліків та інструкцій з безпеки.

Комунікація та співпраця:

- система полегшує комунікацію та співпрацю між медсестрами, дозволяючи їм обмінюватися інформацією та ефективно координувати догляд;
- медсестри можуть надсилати безпечні повідомлення та співпрацювати над планами догляду в межах платформи;
- система підтримує комунікацію в режимі реального часу, що сприяє швидкому прийняттю рішень і своєчасному втручанню в разі надзвичайних ситуацій або змін у стані пацієнта.

Отже, основною метою є розробка комплексної веб-орієнтованої програмної системи, яка відповідає унікальним потребам медсестер, що працюють в умовах будинку престарілих. Завдяки зручному інтерфейсу, надійній функціональності та безперешкодній інтеграції з існуючими робочими процесами, програмне забезпечення має на меті оптимізувати практику медсестринства та сприяти покращенню результатів лікування пацієнтів у будинках для людей похилого віку. Завдяки зручному інтерфейсу, який забезпечує інтуїтивно зрозумілу навігацію та швидкий доступ до необхідної інформації, програмне забезпечення значно полегшує роботу медсестер. Важливим аспектом є також надійна функціональність, яка гарантує безперебійне виконання усіх завдань, пов'язаних з медсестринською практикою, включаючи автоматизацію рутинних процесів, що дозволяє медсестрам більше часу приділяти безпосередньому догляду за пацієнтами. Головною метою розробки такого програмного забезпечення є оптимізація практики медсестринства, що включає підвищення ефективності роботи, зменшення ймовірності помилок, покращення якості обслуговування пацієнтів та забезпечення більш персоналізованого підходу до догляду. Все це, в кінцевому рахунку, сприяє покращенню результатів лікування пацієнтів у будинках для людей похилого віку, підвищенню їх якості життя та задоволеності доглядом.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Для того, щоб розробити комплексну веб-орієнтовану програмну систему, розроблену спеціально для для медсестер будинку літніх людей, необхідно встановити чіткі та детальні вимоги. Ці вимоги, що охоплюють як функціональні, так і нефункціональні аспекти, слугуватимуть основою для концептуалізації, проектування та розробки програмної системи. Нижче наведено опис функціональних вимог до програмної системи.

Управління користувачами:

- система повинна дозволяти адміністратору (лікарю) реєструвати нових медсестер, призначаючи їм унікальні облікові дані для входу в систему;
- медсестри повинні мати можливість входити в систему, використовуючи призначені їм облікові дані для безпечного доступу до системи.

Управління пацієнтами:

- система повинна дозволяти адміністратору додавати нових пацієнтів, фіксуючи такі дані, як ім’я, дата народження, вік, інвалідність, хронічні та невиліковні захворювання;
- медсестри повинні мати можливість переглядати профілі пацієнтів, включаючи призначені ліки, необхідні процедури та контактну інформацію для родичів;
- медичні сестри повинні мати можливість оновлювати записи про пацієнтів, документувати втручання та плани лікування.

Управління медикаментами та процедурами:

- система повинна надавати адміністратору можливість додавати, редагувати та видаляти ліки та процедури;
- медичні сестри повинні мати доступ до бази даних ліків і процедур, переглядати описи та вводити ліки або виконувати процедури відповідно до призначень.

Комунікація та співпраця:

- система повинна сприяти комунікації між медсестрами, дозволяючи їм обмінюватися повідомленнями, оновлювати інформацію про стан пацієнта та

співпрацювати над планами догляду;

– медсестри повинні мати доступ до функції обміну повідомленнями для спілкування з колегами та адміністратором, забезпечуючи ефективну координацію догляду за пацієнтами.

Адміністративні функції:

– адміністратор повинен мати повний контроль над системою з можливістю змінювати будь-який її аспект, включаючи додавання, редагування або видалення медсестер, ліків, процедур і записів про пацієнтів;
– система повинна забезпечувати аудиторські сліди для відстеження змін, внесених адміністратором, і вести облік дій користувачів для цілей підзвітності та аудиту.

Нижче наведено опис нефункціональних вимог до програмної системи.

Безпека:

– система повинна реалізовувати надійні механізми автентифікації, включаючи безпечні облікові дані для входу в систему та протоколи шифрування, для захисту інформації про пацієнта від несанкціонованого доступу;
– контроль доступу повинен застосовуватися для обмеження привілеїв користувачів на основі ролей та дозволів, гарантуючи, що конфіденційні дані будуть доступні лише уповноваженому персоналу.

Зручність використання:

– інтерфейс користувача повинен бути інтуїтивно зрозумілим та зручним, що дозволить медсестрам ефективно орієнтуватися в системі та виконувати завдання з мінімальною підготовкою;
– система повинна надавати чіткі інструкції та вказівки щодо виконання завдань, мінімізуючи помилки та максимізуючи продуктивність користувача.

Надійність:

– система повинна бути надійною та доступною для використання в будь-який час, з мінімальним часом простою та перервами в обслуговуванні;
– механізми резервного копіювання та відновлення мають бути впроваджені для забезпечення цілісності даних та захисту від втрати критично важливої

інформації.

Продуктивність:

- система повинна бути швидкою та продуктивною, здатною працювати з декількома користувачами одночасно без зниження швидкості та функціональності;
- час відгуку для доступу до записів пацієнтів, введення ліків та виконання інших завдань повинен відповідати прийнятним стандартам продуктивності.

Масштабованість:

- система повинна бути масштабованою, щоб відповідати зростанню кількості пацієнтів, медсестер, ліків та процедур з плином часу;
- масштабованість досягається за рахунок гнучкої архітектури та хмарної інфраструктури, що дозволяє безперешкодно розширювати систему в міру розвитку потреб будинку престарілих.

Отже, визначивши ці функціональні та нефункціональні вимоги, можна приступити до проектування та розробки веб-орієнтованої програмної системи з чітким розумінням цілей, завдань та обмежень проекту. Ці вимоги спрямовуватимуть процес розробки, гарантуючи, що отримана програмна система відповідатиме потребам медсестер в умовах будинку престарілих, дотримуючись при цьому галузевих стандартів і найкращих практик.

Таким чином, чітке визначення функціональних та нефункціональних вимог дозволить ефективно спрямувати процес проектування та розробки веб-орієнтованої програмної системи, яка максимально відповідатиме потребам медсестер у будинках престарілих. Визначення цих вимог створює міцну основу для подальших етапів розробки, включаючи планування, реалізацію та тестування системи. Це, у свою чергу, сприятиме підвищенню якості догляду за пацієнтами, оптимізації робочих процесів та загальному покращенню результатів роботи медичного персоналу.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

UML забезпечує комплексну основу для візуалізації, специфікації, конструювання та документування артефактів програмної системи [1]. Завдяки різноманітним діаграмам, UML дозволяє розуміти структуру, поведінку та взаємодію системи.

Діаграма класів для веб-орієнтованої програмної системи для медсестер будинку літніх людей слугує візуальним представленням різних класів, їхніх атрибутів, методів та взаємозв'язків у системі (див. рис. 3.1).

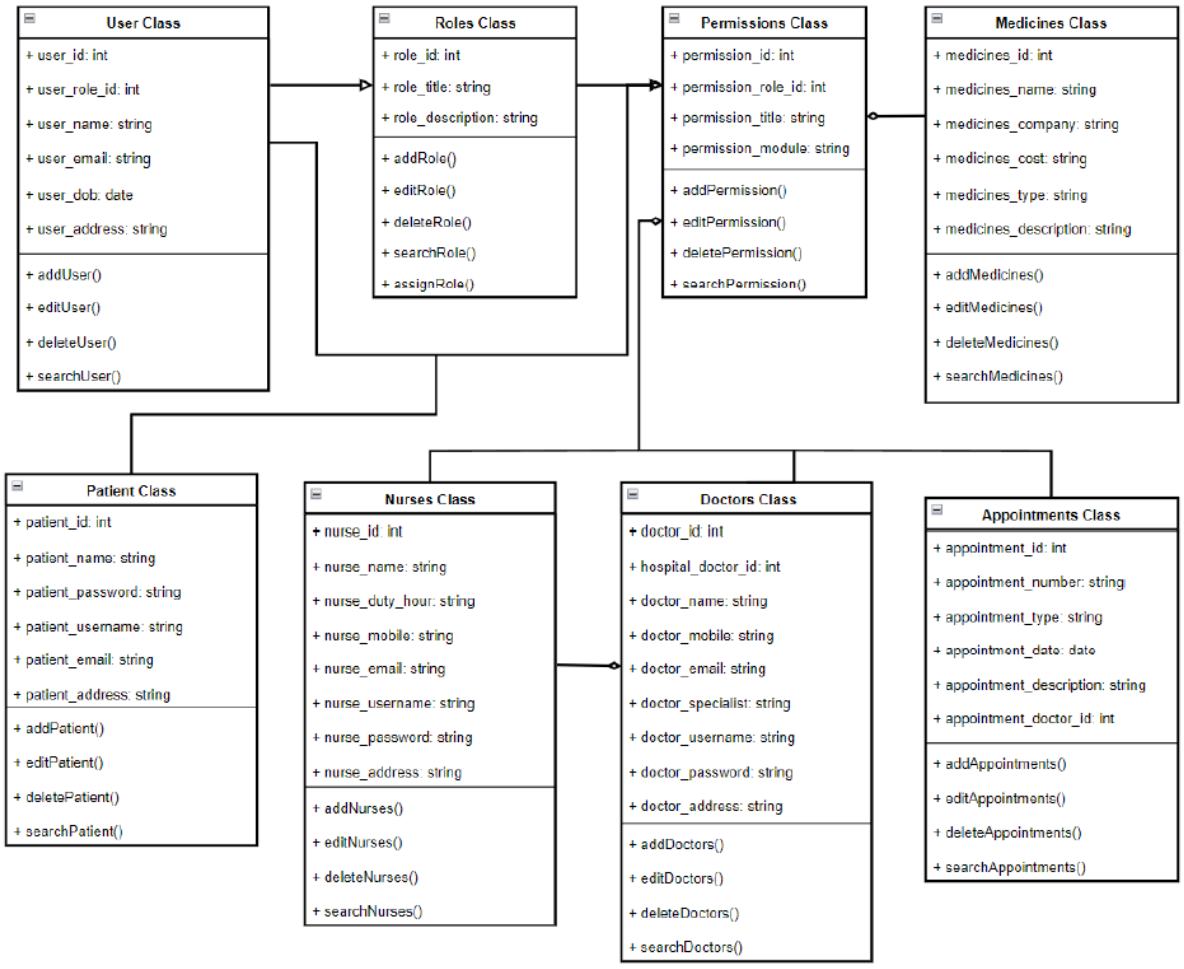


Рисунок 3.1 – Діаграма класів

Клас Roles керує ролями, доступними у системі. Він включає такі атрибути, як ідентифікатор ролі, назва та опис. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти та шукати ролі, а також призначати ролі користувачам.

Клас Permissions обробляє дозволи, призначені різним ролям у системі. Він включає такі атрибути, як ідентифікатор дозволу, ідентифікатор ролі, назва і модуль. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти та шукати дозволи.

Клас Medicines керує ліками, доступними у будинку престарілих. Він включає такі атрибути, як ідентифікатор ліків, назва, компанія, вартість, тип та опис. Методи, пов’язані з цим класом, полегшують додавання, редагування, видалення та пошук ліків.

Клас User представляє користувачів, зареєстрованих у системі. Він включає такі атрибути, як ідентифікатор користувача, ідентифікатор ролі, ім’я, електронна пошта, дата народження та адреса. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти та шукати користувачів.

Клас Appointments обробляє зустрічі, заплановані в системі будинку престарілих. Він включає такі атрибути, як ідентифікатор зустрічі, номер зустрічі, тип, дата, опис та ідентифікатор лікаря. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти і шукати зустрічі.

Клас Doctors представляє лікарів, пов’язаних з будинком престарілих. Він включає такі атрибути, як ідентифікатор лікаря, ідентифікатор лікаря, ім’я, номер мобільного телефону, електронна пошта, спеціалізація, облікові дані для входу (ім’я користувача, пароль) та адреса. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти та шукати лікарів.

Клас Nurses обробляє інформацію про медсестер, які працюють у будинку престарілих. Він включає такі атрибути, як ідентифікатор медсестри, ім’я, графік роботи, номер мобільного телефону, електронна пошта, облікові дані для входу (ім’я користувача, пароль) та адреса. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти та шукати медсестер.

Клас Patient інкапсулює дані про пацієнтів, зареєстрованих у системі будинку престарілих. Він містить основні атрибути, такі як ідентифікатор пацієнта, ім’я, облікові

дані для входу (ім’я користувача та пароль), електронну пошту та адресу. Ці атрибути полегшують унікальну ідентифікацію та комунікацію з пацієнтами. Методи, пов’язані з цим класом, дозволяють додавати, редагувати, видаляти та шукати пацієнтів.

Отже, діаграма класів надає структурований огляд компонентів системи та їхньої взаємодії, що полегшує проектування, розробку та підтримку веб-орієнтованої програмної системи [2].

Діаграма компонентів представляє веб-орієнтовану програмну систему для медсестер будинку літніх людей (див. рис. 3.2). Вона демонструє компоненти, інтерфейси, порти та взаємозв’язки між різними модулями, полегшуючи розуміння архітектури та функціональності системи [3].

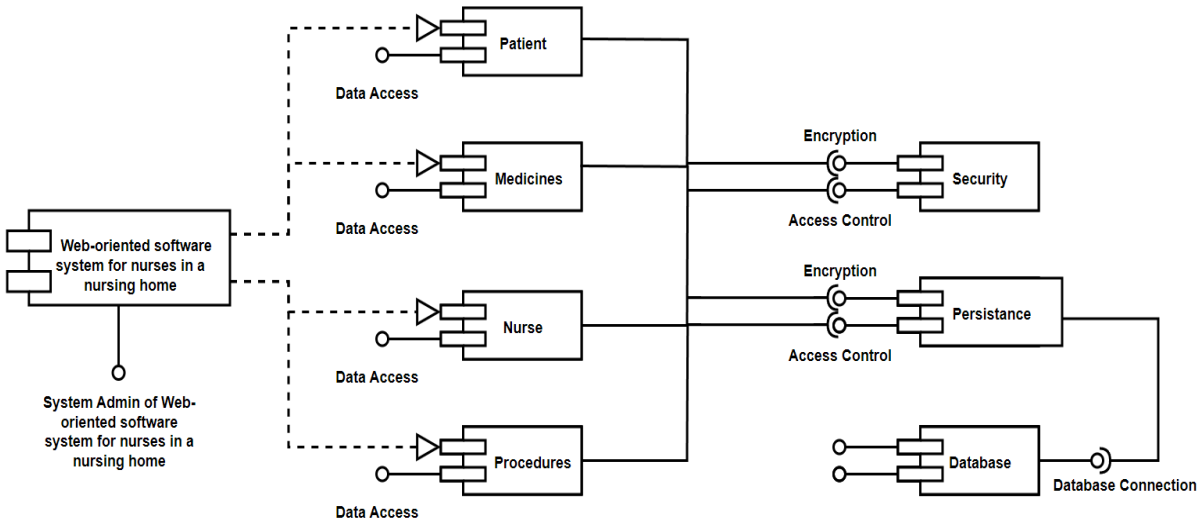


Рисунок 3.2 – Діаграма компонентів

Компонент "Пацієнти" відіграє вирішальну роль у наданні пацієнт-орієнтованої допомоги в системі будинків престарілих. Він дає змогу медичним працівникам вести повну медичну документацію пацієнтів, відстежувати показники здоров’я та ефективно організовувати зустрічі з пацієнтами. Централізуючи інформацію про пацієнта та засоби комунікації, цей компонент дозволяє медичним працівникам надавати персоналізовану допомогу, адаптовану до потреб кожного пацієнта, одночасно сприяючи залученню та задоволенню пацієнтів.

Компонент "Ліки" є фундаментальною основою управління медикаментозним забезпеченням в умовах будинку престарілих. Він надає медичним працівникам надійну платформу для відстеження запасів медикаментів, призначення ліків і контролю за дотриманням пацієнтами режиму прийому. Завдяки таким функціям, як планування прийому ліків, сповіщення та нагадування, цей компонент підвищує безпеку та дотримання пацієнтами режиму прийому ліків, що в кінцевому підсумку сприяє покращенню результатів лікування та якості медичної допомоги.

Компонент "Медсестри" слугує ключовим аспектом цифрової інфраструктури будинку престарілих, полегшуючи безперебійну координацію та управління медсестринським персоналом. Він дозволяє адміністраторам реєструвати, контролювати та організовувати профілі медсестер, забезпечуючи ефективне розгортання та розподіл медсестринських ресурсів для виконання різних завдань з догляду за пацієнтами. Крім того, цей компонент покращує канали комунікації між медсестрами, оптимізуючи співпрацю та обмін інформацією в межах будинку престарілих.

Компонент "Процедури" впорядковує управління медичними процедурами та втручаннями в умовах будинку престарілих. Він полегшує планування, документування та координацію різних медичних процедур, забезпечуючи ефективне використання ресурсів і своєчасне виконання заходів з догляду за пацієнтами. Пропонуючи в реальному часі видимість розкладу процедур і оновлення статусів, цей компонент дозволяє медичним працівникам оптимізувати ефективність робочого процесу і надавати високоякісну допомогу пацієнтам, які проходять медичні втручання.

Компонент автентифікації захищає цілісність та безпеку цифрової інфраструктури будинку престарілих, впроваджуючи надійні механізми автентифікації користувачів і контролю доступу. Він перевіряє ідентичність користувачів, авторизує доступ на основі заздалегідь визначених ролей і дозволів, а також зберігає аудиторські сліди дій користувачів. Завдяки суворим протоколам автентифікації та заходам безпеки цей компонент зменшує ризик несанкціонованого доступу та витоку даних, захищаючи конфіденційну інформацію про пацієнтів і дотримуючись нормативних стандартів.

Отже, діаграма компонентів допомагає в проектуванні, розробці та підтримці веб-орієнтованої програмної системи, забезпечуючи чіткість та узгодженість її архітектури та реалізації.

Діаграма діяльності входу в систему слугує візуальним представленням послідовності кроків, пов’язаних з процесом входу в систему у веб-орієнтованому програмному забезпеченні, призначеному для медсестер будинку літніх людей (див. рис. 3.3).

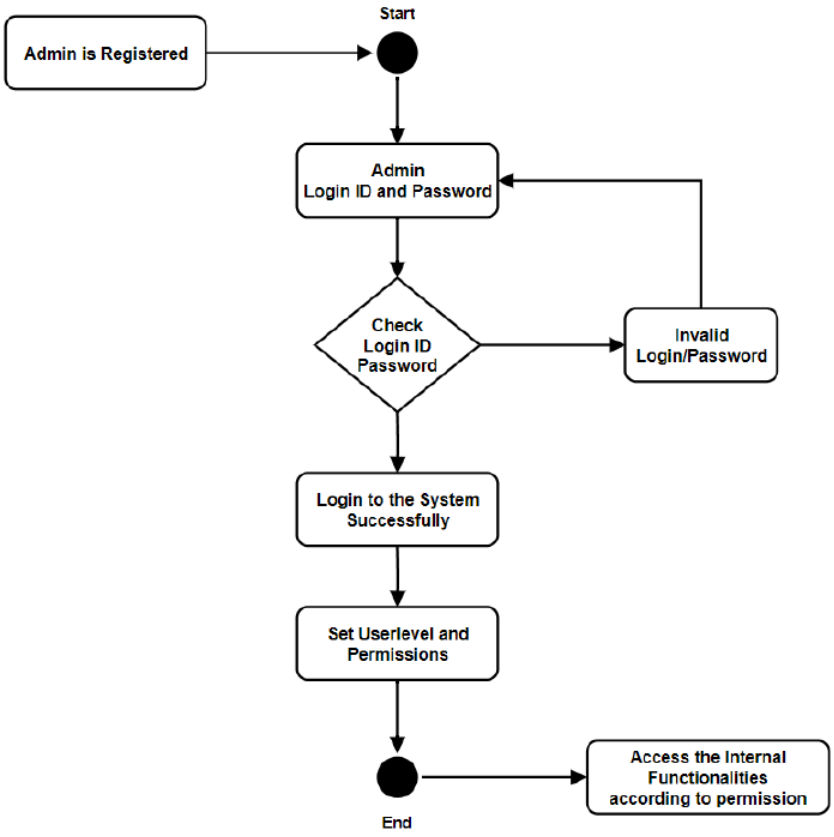


Рисунок 3.3 – Діаграма діяльності входу в систему управління будинком престарілих

Діаграма діяльності входу в систему ілюструє послідовність дій, пов’язаних з процесом входу в систему управління будинком для людей похилого віку. Після входу на сторінку входу адміністратору пропонується ввести своє ім’я користувача та пароль. Після успішної автентифікації адміністратор отримує доступ до різних функцій, пов’язаних з управлінням медсестрами, прийомом ліків, доглядом за пацієнтами та координацією роботи медсестер. Ці функції безпечно доступні тільки після того, як

адміністратор увійшов в систему, що забезпечує конфіденційність даних і безпеку системи. Діаграма підкреслює взаємодію між різними компонентами, такими як медсестри, ліки, пацієнти та лікарі, наголошуючи на важливості перевірки особи перед доступом до конфіденційної інформації або виконанням системних завдань.

Отже, діаграма діяльності входу описує покроковий процес, за допомогою якого адміністратор отримує доступ до системи управління будинком престарілих, підкреслюючи важливість безпеки, автентифікації та досвіду користувача для забезпечення безперебійної та безпечної роботи системи [4].

Діаграма діяльності системи управління будинком престарілих ілюструє послідовність дій і переходів між різними компонентами, включаючи ліки, пацієнтів, медсестер і лікарів (див. рис. 3.4).

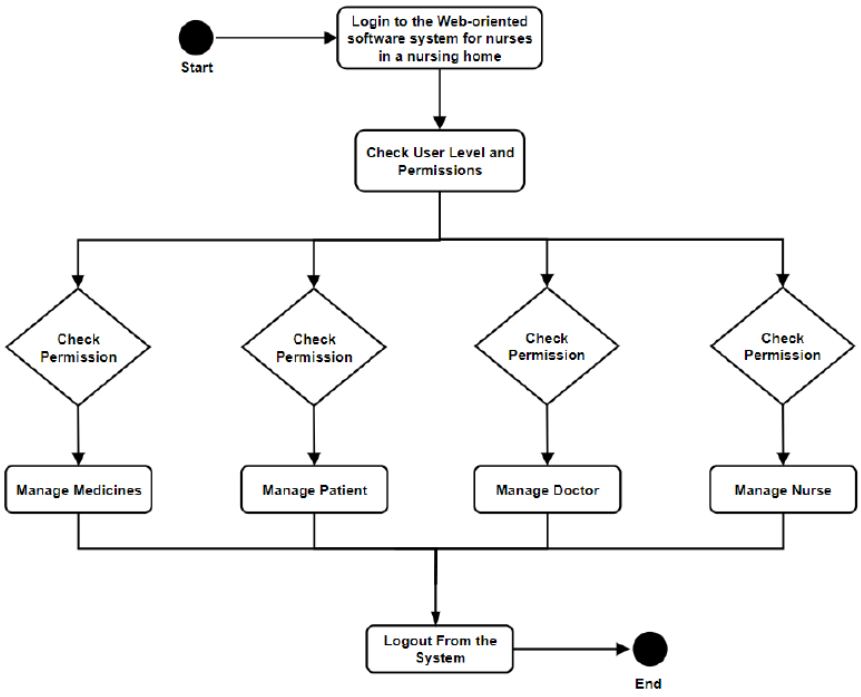


Рисунок 3.4 –Діаграма діяльності системи управління будинком престарілих

Користувачі-адміністратори можуть виконувати такі дії, як пошук, перегляд описів, додавання, оновлення та видалення ліків в системі.

На діаграмі показано потік дій, пов’язаних з редагуванням, додаванням та оновленням записів про пацієнта.

Користувачі мають можливість шукати і створювати звіти, пов’язані з медсестрами і лікарями в системі.

Всі об’єкти, включаючи ліки, пацієнтів, лікарів і медсестер, взаємопов’язані між собою, демонструючи цілісний потік діяльності в системі.

Діаграма надає комплексне зображення діяльності, описів і потоків, що відносяться до ліків, лікарів, медсестер і пацієнтів в системі управління будинком престарілих.

Отже, діаграма діяльності системи управління будинком престарілих візуально представляє діяльність і взаємодію в системі, пропонуючи розуміння управління різними компонентами і потоком операцій, що виконуються користувачами-адміністраторами та іншими зацікавленими сторонами [5].

Діаграма "сутність-зв’язок" (ER-діаграма) слугує для візуального представлення структур баз даних і зв’язків у системі [6]. ER-діаграма на рисунку 3.5 ілюструє взаємозв’язки між різними сутностями.

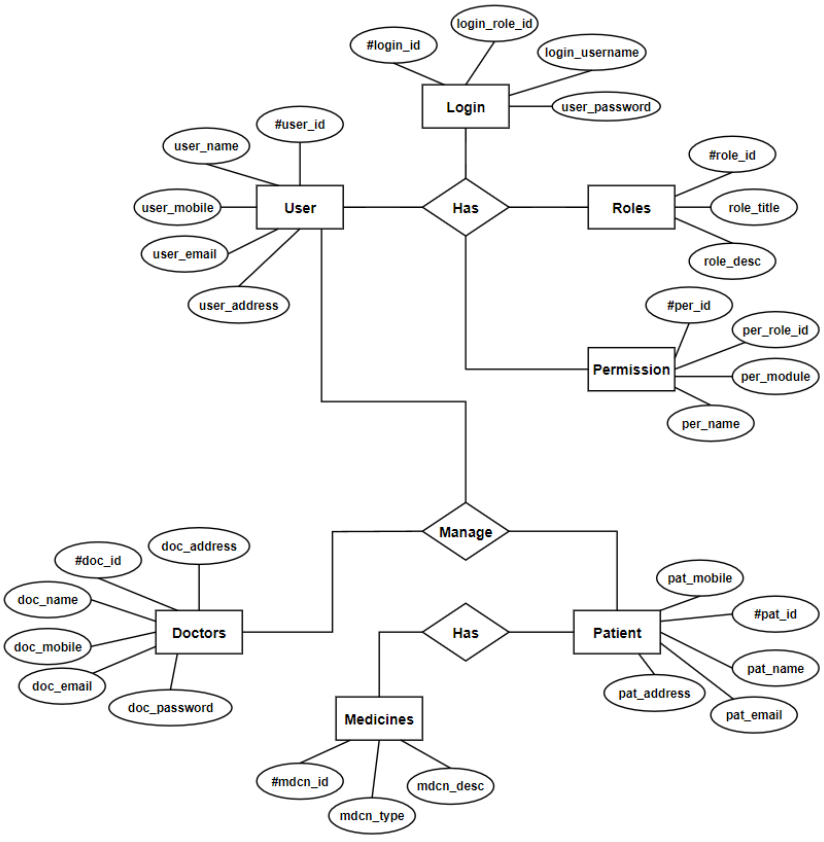


Рисунок 3.5 – ER-діаграма

ER-діаграма представляє модель системи управління будинком літніх людей, зображуючи структуру таблиць бази даних і зв'язки між ними. Вона допомагає зрозуміти організацію даних і взаємозв'язки, які визначають функціональність системи управління будинком літніх людей.

Кожна сутність включає первинні та унікальні ключі для забезпечення цілісності та унікальності даних.

Отже, ER-діаграма системи управління будинком престарілих дає уявлення про структуру бази даних, взаємозв'язки та оптимізації, впроваджені для ефективної підтримки функціональності системи.

Таким чином, UML проектування програмного забезпечення є цінним інструментом, що дозволяє створювати добре спроектовані, підтримувані та масштабовані програмні системи, які задовольняють потреби зацікавлених сторін та кінцевих користувачів.

3.2 Проєктування архітектури ПЗ

При розробці веб-орієнтованої програмної системи для медсестер будинку літніх людей архітектура відіграє вирішальну роль у забезпеченні масштабованості, надійності та ефективності. Однією з широко використовуваних архітектур для веб-додатків є трирівнева архітектура, яка розділяє додаток на три окремі рівні: рівень клієнта, рівень бізнес-логіки та рівень бази даних (див. рис. 3.6). При розробці цієї системи використовувався стек MERN, який складається з MongoDB, Express.js, React.js та Node.js.

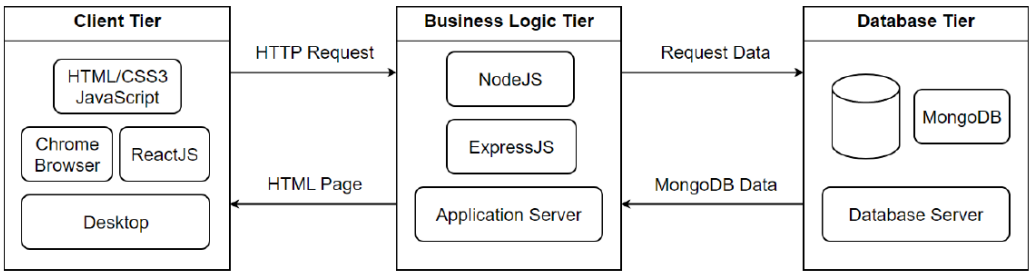


Рисунок 3.6 – Трирівнева архітектура MERN

Клієнтський рівень являє собою інтерфейсний рівень програми, з яким користувачі взаємодіють безпосередньо. Він відповідає за представлення даних і полегшення взаємодії з користувачем за допомогою інтуїтивно зрозумілого і чуйного інтерфейсу [7].

Клієнтський рівень реалізовано за допомогою JavaScript, HTML та CSS на фреймворку React.js. JavaScript, HTML та CSS – це фундаментальні технології, що використовуються для створення веб-сторінок та інтерактивних користувацьких інтерфейсів [8].

React.js – це бібліотека JavaScript для побудови користувацьких інтерфейсів, яка відома своєю архітектурою на основі компонентів та віртуальним рендерингом DOM, що дозволяє ефективно оновлювати інтерфейс [9].

Клієнтський рівень рендерить різні компоненти користувацького інтерфейсу, такі як профілі пацієнтів, списки ліків та розклад процедур, на основі даних, отриманих з сервера. Він обробляє взаємодію з користувачем, наприклад, надсилання форм, натискання кнопок та навігацію між різними поданнями, забезпечуючи безперебійний та інтерактивний досвід для користувачів. React.js дозволяє модулювати компоненти інтерфейсу користувача, що полегшує управління та підтримку кодової бази, а також сприяє повторному використанню та масштабуванню.

Рівень бізнес-логіки служить посередником між клієнтським рівнем і рівнем бази даних. Він містить логіку додатку та обробляє дані і бізнес-правила.

Рівень бізнес-логіки побудований з використанням Node.js та Express.js. Node.js – це середовище виконання JavaScript, яке дозволяє програмувати на стороні сервера, в той час як Express.js – це фреймворк веб-додатків для Node.js, що надає надійний набір функцій для побудови веб-серверів та API [10].

Рівень бізнес-логіки отримує запити від клієнтського рівня та обробляє їх, виконуючи необхідну бізнес-логіку для виконання запитів. Він взаємодіє з рівнем бази даних для отримання або маніпулювання даними, виконуючи CRUD-операції (створення, читання, оновлення, видалення) за необхідності. Express.js полегшує

створення RESTful API, забезпечуючи зв’язок між клієнтським рівнем і рівнем бази даних за допомогою HTTP-запитів і відповідей [11]. Аутентифікація, авторизація та валідація користувацьких даних обробляються на рівні бізнес-логіки, щоб забезпечити цілісність, безпеку та відповідність нормативним стандартам.

Рівень бази даних відповідає за зберігання та управління даними програми. Він відповідає за зберігання та управління даними програми у структурованому форматі.

Рівень бази даних використовує MongoDB як систему управління базами даних. MongoDB – це документно-орієнтована база даних NoSQL, яка зберігає дані у гнучких JSON-подібних документах, що робить її придатною для зберігання неструктурованих або напівструктурованих даних [12].

Рівень бази даних зберігає всі основні дані, необхідні для роботи програми, включаючи записи пацієнтів, профілі медсестер, інформацію про ліки та процедури, у вигляді колекцій або документів. Він обробляє CRUD-операції, ініційовані рівнем бізнес-логіки, виконуючи запити та команди для отримання, вставки, оновлення або видалення даних з бази даних.

Гнучка схема та можливості масштабування MongoDB дозволяють ефективно зберігати та отримувати дані, враховуючи динамічний характер медичних даних та мінливі потреби додатку. Цілісність та узгодженість даних забезпечується за допомогою таких механізмів, як транзакції, індекси та правила перевірки даних, що підтримують надійність та точність збережених даних.

Отже, завдяки використанню стеку MERN та реалізації трирівневої архітектури веб-орієнтована програмна система для медсестер будинку літніх людей має модульний, масштабований та ефективний дизайн. Клієнтський рівень забезпечує зручний інтерфейс для взаємодії з додатком, рівень бізнес-логіки відповідає за обробку даних та бізнес-правил, а рівень бази даних забезпечує стійкість та цілісність даних додатку. Разом ці рівні формують міцну та надійну архітектуру, яка відповідає потребам медичних працівників та покращує надання допомоги пацієнтам в умовах будинку для людей похилого віку.

3.3 Проєктування структури зберігання даних

MongoDB, документно-орієнтована база даних з відкритим вихідним кодом, є ключовим компонентом у розробці структури зберігання даних для веб-орієнтованої програмної системи для медсестер будинку літніх людей. Гнучка схема, висока продуктивність та масштабованість MongoDB роблять її ідеальним вибором для ефективного зберігання та управління медичними даними.

MongoDB працює як сервер баз даних, забезпечуючи платформу для зберігання та управління даними в документно-орієнтованому форматі. Дані організовані в бази даних, колекції та документи (див. рис. 3.7).

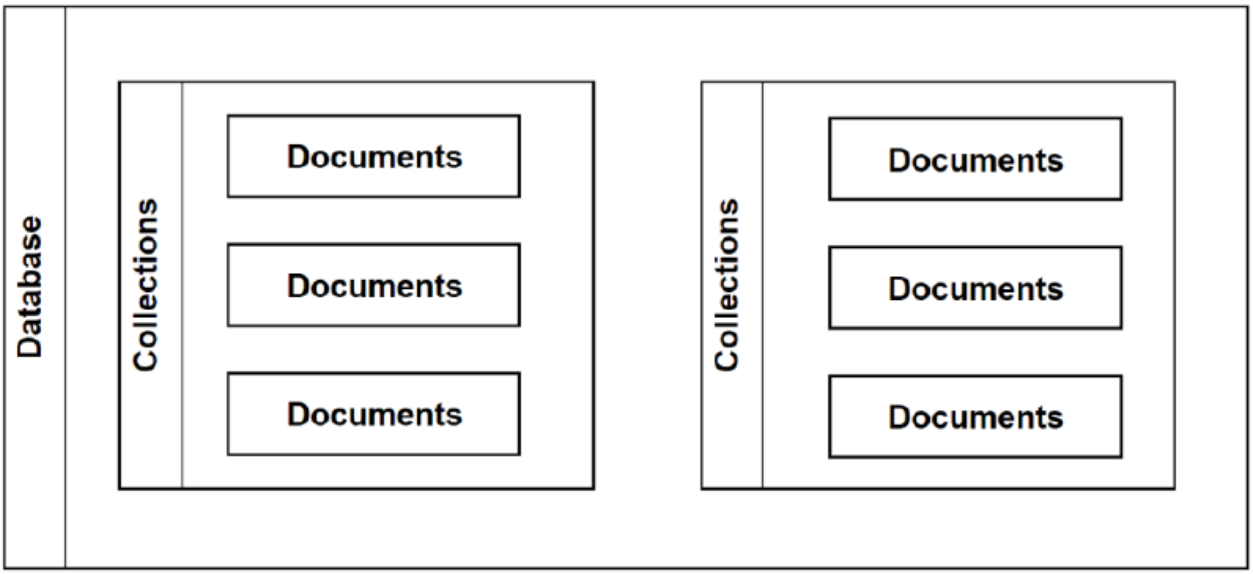


Рисунок 3.7 – Взаємозв’язок між базами даних, колекціями та документами

MongoDB дозволяє створювати кілька баз даних, кожна з яких містить одну або кілька колекцій. Бази даних слугують контейнерами для колекцій та забезпечують ізоляцію та управління даними на найвищому рівні.

Колекції є аналогом таблиць у реляційних базах даних, але не мають схеми. Вони зберігають кілька документів, кожен з яких представляє запис або сутність, і можуть містити різні типи даних в межах однієї колекції.

Документи є основною одиницею зберігання даних в MongoDB і представлені у форматі BSON (Binary JSON) [13]. Вони складаються з пар ключ-значення, де ключами

є назви полів, а значеннями можуть бути різні типи даних, такі як рядки, числа, масиви або вкладені документи.

Особливості MongoDB:

- безсхемна природа MongoDB дозволяє колекціям зберігати різні типи документів з різною структурою, забезпечуючи гнучкість у моделюванні та управлінні даними;
- дані в MongoDB зберігаються в документах, а не в таблицях, що робить її більш інтуїтивно зрозумілою та гнучкою для представлення складних структур даних;
- MongoDB підтримує індексування полів у документах, що забезпечує ефективний пошук даних і виконання запитів. Індексування покращує продуктивність запитів, зменшуючи кількість документів, що скануються під час пошуку;
- MongoDB пропонує горизонтальну масштабованість за допомогою шардингу, що дозволяє розподіляти дані між декількома серверами. Це гарантує, що система може обробляти великі обсяги даних і пристосовуватися до зростання з часом;
- MongoDB забезпечує високу доступність та відмовостійкість завдяки реплікації, коли кілька копій даних зберігаються на різних серверах. У разі збою сервера MongoDB автоматично переходить на резервну копію, забезпечуючи безперервну роботу сервісу;
- MongoDB підтримує операції агрегації, що дозволяє обробляти та аналізувати дані в межах колекцій. Конвеєри агрегації, функції картографічної редукції та одноцільові методи агрегації дозволяють виконувати складні перетворення даних та обчислення;
- завдяки таким функціям, як індексування, шардінг та реплікація, MongoDB пропонує високу продуктивність та масштабованість для роботи з великими наборами даних та великими обсягами транзакцій.

Переваги MongoDB для програмної системи:

- безсхемна природа MongoDB усуває необхідність у попередньо визначених

схемах, що дозволяє гнучко представляти медичні дані без жорстких структур;

- можливості MongoDB з індексування та оптимізації запитів забезпечують швидкий пошук і обробку даних, підвищуючи швидкість реагування програмної системи для медсестер;
- оскільки ця програмна система обробляє більше даних, можливості шардингу MongoDB дозволяють здійснювати горизонтальне масштабування для розподілу даних між декількома серверами, забезпечуючи постійну продуктивність та доступність;
- документно-орієнтована природа MongoDB добре узгоджується зі складною та різноманітною природою медичних даних, дозволяючи легко зберігати та знаходити записи пацієнтів, дані про ліки та інформацію про процедури;
- MongoDB легко інтегрується з іншими технологіями та фреймворками, що полегшує сумісність з існуючими системами та майбутніми вдосконаленнями.

Отже, MongoDB слугує основою структури зберігання даних для веб-орієнтованої програмної системи для медсестер будинку літніх людей. Її гнучка схема, висока продуктивність, масштабованість і багатофункціональні можливості роблять її ідеальним вибором для ефективного та раціонального управління медичними даними, що дозволяє створити надійне програмне рішення, яке відповідає потребам медичних працівників і покращує якість надання допомоги пацієнтам.

3.4 Приклади найцікавіших алгоритмів та методів

У сфері веб-орієнтованих програмних систем для медсестер будинку літніх людей ефективне управління такими завданнями, як розподіл пацієнтів, планування прийому ліків і процедур, має першорядне значення для забезпечення оптимального надання медичної допомоги.

Використання складних алгоритмів має вирішальне значення для автоматизації цих завдань та оптимізації використання ресурсів з дотриманням різних обмежень і вимог. Тому нижче наведено три найцікавіші алгоритми та методи, розроблені для цієї програмної системи: алгоритм розподілу пацієнтів, алгоритм управління медикаментами та алгоритм планування процедур. Кожен алгоритм стосується певного аспекту

управління будинком престарілих, пропонуючи рішення для оптимізації операцій, покращення догляду за пацієнтами та підвищення загальної ефективності. Завдяки поєднанню обчислювальних методів, таких як динамічне програмування та генетичний алгоритм, ці алгоритми надають цінну інформацію та практичні інструменти для розробки надійних програмних систем, пристосованих до унікальних потреб медсестер і пацієнтів в умовах будинку для людей похилого віку.

Алгоритм розподілу пацієнтів розроблений для ефективного розподілу медсестер між пацієнтами на основі різних факторів, таких як навантаження медсестер, потреби пацієнтів та справедливість розподілу. Алгоритм має на меті рівномірно розподілити пацієнтів між доступними медсестрами, враховуючи балансування навантаження та вимоги до догляду за пацієнтами.

У будинку престарілих є кілька пацієнтів з різними медичними потребами, а кількість медсестер, які можуть їх обслуговувати, обмежена. Тому передбачено справедливий та ефективний розподіл медсестер між пацієнтами для забезпечення оптимального надання допомоги.

Алгоритм планування Round Robin (RR) є поширеним методом розподілу завдань, коли завдання призначаються ресурсам по колу, гарантуючи, що кожен ресурс отримує рівну частку завдань протягом певного часу [14].

Однак в умовах будинку престарілих простого алгоритму RR може бути недостатньо, оскільки медсестри можуть мати різне робоче навантаження або вимоги до догляду за пацієнтами.

Щоб вирішити цю проблему, було вдосконалено алгоритм RR за допомогою зваженої справедливої черги, яка призначає вагу медсестрам на основі їх робочого навантаження або інших відповідних факторів.

Медсестри з більшим навантаженням або призначені до пацієнтів з більш критичними потребами отримують більшу вагу, забезпечуючи відповідний розподіл пацієнтів.

На рисунку 3.8 можна побачити програмний код реалізації цього алгоритму розподілу пацієнтів.

```
function allocateNurseToPatient(nurses, patients) {
    let nurseIndex = 0;

    patients.forEach((patient) => {
        if (nurseIndex >= nurses.length) {
            nurseIndex = 0; // Повернутися до першої медсестри, коли всі медсестри пройдені
        }

        const nurse = nurses[nurseIndex];
        nurse.assignPatient(patient); // Призначити пацієнта до медсестри
        nurseIndex++; // Перейти до наступної медсестри для наступного пацієнта
    });
}
```

Рисунок 3.8 – Алгоритм розподілу пацієнтів

Функція allocateNurseToPatient приймає два параметри: масив об’єктів медсестер і масив об’єктів пацієнтів. Вона ітераційно проходить через кожного пацієнта в масиві пацієнтів. Для кожного пацієнта вона призначає пацієнта до медсестри з поточним індексом nurseIndex. Якщо nurseIndex перевищує кількість доступних медсестер (nurses.length), він скидає nurseIndex на 0, щоб почати призначати пацієнтів з початку списку медсестер. Цей процес триває до тих пір, поки всі пацієнти не будуть призначені медсестрам на основі алгоритму RR із зваженою справедливою чергою.

Алгоритм розподілу пацієнтів пропонує системний підхід до розподілу медсестер між пацієнтами, ефективно балансуючи навантаження та потреби пацієнтів. Використовуючи Round Robin зі зваженою справедливою чергою, цей алгоритм забезпечує справедливий розподіл медсестринських ресурсів, одночасно сприяючи наданню допомоги, орієнтованої на пацієнта.

Алгоритм управління прийомом ліків відповідає за створення графіків прийому ліків для пацієнтів у будинку для людей похилого віку. Його мета – створити оптимізований графік, який гарантує, що пацієнти отримають свої ліки в потрібний час, враховуючи такі фактори, як час прийому ліків, взаємодія та індивідуальні потреби пацієнта.

В умовах будинку престарілих пацієнти часто потребують прийому декількох ліків через певні проміжки часу протягом дня. Тому передбачено створення персоналізованих графіків прийому ліків для кожного пацієнта на основі призначених ліків і режимів дозування.

Динамічне програмування (ДП) – це потужний алгоритмічний метод, який використовується для вирішення оптимізаційних задач шляхом розбиття їх на простіші підзадачі та зберігання їх розв'язків, щоб уникнути надлишкових обчислень [15]. У контексті планування прийому ліків, ДП можна використовувати для створення оптимальних розкладів, дотримуючись обмежень, таких як час прийому ліків, взаємодія та межі дозування.

Функція generateMedicationSchedule призначена для ефективної генерації розкладу прийому ліків для пацієнтів в умовах будинку престарілих:

```
function generateMedicationSchedule(patient) {
    let schedule = [];
```

Функція починається з ініціалізації порожнього масиву schedule для зберігання згенерованого розкладу прийому ліків.

Вона витягує відповідну інформацію з об'єкта пацієнта, включаючи список ліків (medications) та обмеження (constraints), такі як максимальна добова доза, загальна доза та тривалість лікування:

```
  const medications = patient.medications;
    const constraints = patient.constraints;
```

Динамічне програмування використовується для розрахунку оптимального графіка прийому ліків з урахуванням обмежень, визначених для кожного препарату.

Таблиця динамічного програмування dp ініціалізується для зберігання розрахункових значень для кожної комбінації препаратів і днів, що залишилися:

```
        const dp = new Array(medications.length +
1).fill(null).map(() => new Array(constraints.maxDays +
1).fill(null));
```

Функція computeSchedule рекурсивно обчислює максимальну користь (ефективність) від прийому ліків з урахуванням поточного індексу ліків і днів, що залишилися.

```
function computeSchedule(index, daysLeft) {
```

На кожному кроці вона розглядає два варіанти: або прийняти поточний препарат, або пропустити його, виходячи з таких обмежень, як максимальна добова доза та загальна доза:

```
            if (index === medications.length || daysLeft === 
0) {
                return 0;  }
            if (dp[index][daysLeft] !== null) {
                return dp[index][daysLeft];}
            let maxBenefit = computeSchedule(index + 1, 
daysLeft);
            if (medications[index].dailyDose <= 
constraints.maxDailyDose && medications[index].totalDose <= 
constraints.maxTotalDose) {
                const benefit = medications[index].benefit + 
computeSchedule(index + 1, daysLeft - 1);
                maxBenefit = Math.max(maxBenefit, benefit);}
```

Функція обчислює максимальну користь і зберігає її в таблиці DP, щоб уникнути зайвих обчислень:

```
            dp[index][daysLeft] = maxBenefit;
            return maxBenefit;}
        computeSchedule(0, constraints.maxDays);
```

Після того, як таблиця DP заповнена оптимальними значеннями, функція виконує ретроспективний аналіз, щоб відновити оптимальний графік прийому ліків:

```
        let remainingDays = constraints.maxDays;
        for (let i = 0; i < medications.length; i++) {
            if (computeSchedule(i, remainingDays) !==
computeSchedule(i + 1, remainingDays)) {
                schedule.push(medications[i]);
                remainingDays -= 1;}}
        return schedule; }
```

Починаючи з початку списку ліків, вона ітераційно переглядає кожен препарат і перевіряє, чи дає його прийом більшу користь порівняно з пропуском. Якщо прийом ліків є корисним, він додається до розкладу, а дні, що залишилися, відповідно оновлюються. Цей процес триває доти, доки не буде побудовано оптимальний розклад для заданої тривалості. Функція повертає згенерований графік прийому ліків у вигляді масиву, що містить послідовність препаратів, які потрібно приймати протягом заданого періоду часу.

Алгоритм управління прийомом ліків демонструє можливості динамічного програмування у створенні оптимальних графіків прийому ліків для пацієнтів. Враховуючи час прийому ліків, взаємодію та індивідуальні обмеження пацієнта, цей алгоритм сприяє безпечному та ефективному застосуванню ліків, тим самим покращуючи результати лікування та знижуючи ризик виникнення побічних ефектів.

Алгоритм планування процедур розроблений для ефективного планування медичних процедур для пацієнтів і медсестер у будинку для людей похилого віку. Він має на меті оптимізувати розподіл процедур для пацієнтів, враховуючи доступність медсестер, потреби пацієнтів та інші відповідні обмеження.

В умовах будинку для людей похилого віку пацієнтам необхідно запланувати різні медичні процедури, такі як огляди, лікування та терапевтичні сеанси. Крім того, необхідно призначити медсестер для нагляду та допомоги у виконанні цих процедур.

Тому передбачено створення оптимізованого розкладу, який максимізує використання ресурсів, мінімізує час очікування та забезпечує своєчасне надання допомоги.

Генетичний алгоритм – це потужний метод оптимізації, натхненний процесом природного відбору та еволюції [16]. Він імітує процес природного відбору, ітеративно генеруючи популяцію потенційних рішень (хромосом), оцінюючи їх придатність та розвиваючи їх протягом поколінь за допомогою операцій відбору, кросинговеру та мутації для знаходження оптимального рішення.

На рисунку 3.9 наведено алгоритм планування процедур на основі генетичного алгоритму.

```
function scheduleProcedures(patients, nurses) {
    let population = initializePopulation();
    let bestSchedule = null;
    while (!terminationConditionMet()) {
        population = evolvePopulation(population);
        bestSchedule = getBestSchedule(population);
    }
    return bestSchedule;
}
```

Рисунок 3.9 – Алгоритм планування процедур

Функція scheduleProcedures приймає два параметри: масив об’єктів пацієнтів і масив об’єктів медсестер. Вона ініціалізує популяцію потенційних розкладів за допомогою функції initializePopulation. Кожен розклад являє собою потенційне рішення (хромосому) для задачі планування процедур. Функція входить в цикл, який продовжується до тих пір, поки не буде виконана умова завершення, наприклад, досягнення максимальної кількості поколінь або знаходження задовільного розв’язку. На кожній ітерації циклу популяція еволюціонує за допомогою генетичних операторів, таких як селекція, кросинговер і мутація, за допомогою функції evolvePopulation. Після кожного кроку еволюції отримується найкращий розклад (розв’язок) з популяції за

допомогою функції getBestSchedule. Потім функція повертає найкращий розклад, знайдений після процесу оптимізації.

Алгоритм планування процедур використовує генетичний алгоритм для оптимізації планування медичних процедур для пацієнтів і медсестер. Завдяки ітеративній еволюції та процесам відбору цей алгоритм дозволяє створювати оптимізовані розклади, які максимізують використання ресурсів і мінімізують час очікування, що в кінцевому підсумку підвищує загальну ефективність надання медичної допомоги.

Отже, використовуючи ці алгоритми у веб-орієнтовані програмні системи, медичні працівники можуть оптимізувати роботу, покращити результати лікування пацієнтів і підвищити рівень догляду в будинках для людей похилого віку, тим самим покращуючи якість життя як пацієнтів, так і персоналу.

3.5 Створення UI / UX дизайну системи

Веб-орієнтована програмна система для медсестер будинку літніх людей розроблена з комплексним та інтуїтивно зрозумілим користувацьким інтерфейсом (UI) для спрощення завдань, пов’язаних з веденням пацієнтів, координацією роботи лікарів, створенням медичних довідок та відстеженням даних IMT. Дизайн UI/UX надає пріоритет простоті, ефективності та доступності, щоб забезпечити безперешкодну навігацію та оптимальний користувацький досвід для медичних працівників [17].

Сторінка "Пацієнти" пропонує зручний інтерфейс для додавання нових пацієнтів та ефективного управління їхньою особистою та медичною інформацією (див. рис. 3.10).

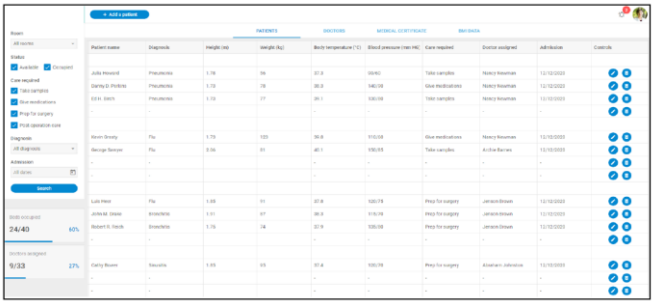


Рисунок 3.10 – Сторінка "Пацієнти"

Особлива увага приділяється полям для введення даних, які мають чіткі позначення та випадаючі меню для легкого вибору та введення.

Різні розділи організовані логічно, що дозволяє користувачам легко вводити особисті дані, інформацію для зв’язку, контакти для екстрених випадків, страхові дані та налаштування розподілу пацієнтів (див. рис. 3.11).

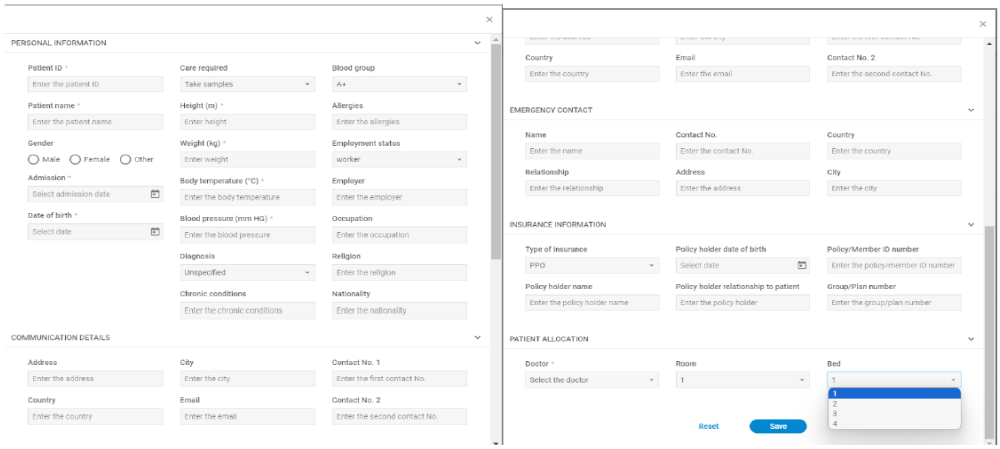


Рисунок 3.11 – Додавання користувача

Елементи інтерфейсу розроблені з урахуванням адаптивності, забезпечуючи сумісність з різними пристроями та розмірами екранів для зручності та гнучкості.

Сторінка "Лікарі" сприяє ефективній комунікації та координації між медичним персоналом (див. рис. 3.12).

Рисунок 3.12 – Сторінка "Лікарі"

Тут представлений повний список медичних працівників, а також їхні відповідні ролі, контактні дані, статус зайнятості та пацієнти. Користувачі можуть швидко шукати,

фільтрувати та сортувати список, щоб знайти конкретних лікарів відповідно до своїх уподобань чи вимог. Дизайн інтерфейсу сприяє наочності та доступності, дозволяючи користувачам отримувати доступ до відповідної інформації та виконувати необхідні дії з мінімальними зусиллями.

Сторінка "Медична довідка" пропонує зручний інтерфейс для створення медичних довідок для пацієнтів (див. рис. 3.13).

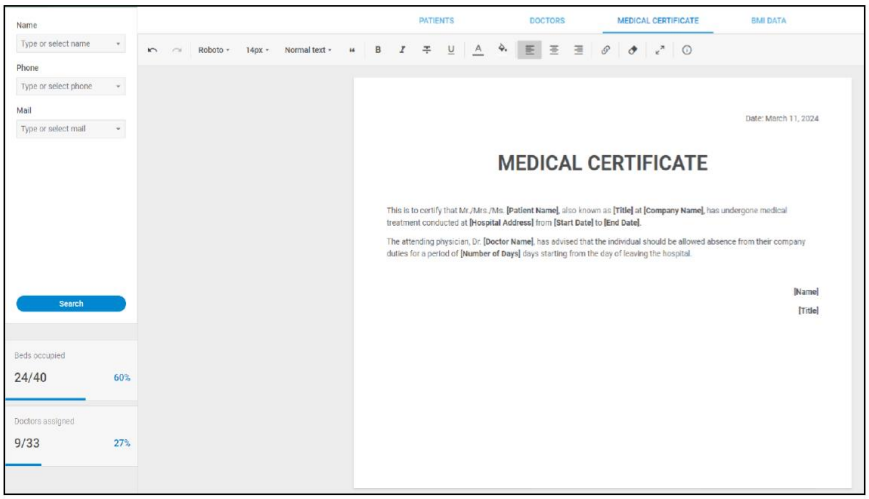


Рисунок 3.13 – Сторінка "Медична довідка"

Надається стандартний шаблон, що дозволяє користувачам легко вводити дані про пацієнта, інформацію про будинок престарілих, рекомендації лікуючого лікаря та термін дії довідки. Інтуїтивно зрозумілі інструменти редагування дозволяють користувачам налаштовувати зміст, формат і макет довідки відповідно до конкретних вимог. Дизайн інтерфейсу підкреслює ясність і простоту, гарантуючи, що користувачі можуть легко створювати точні та професійні медичні довідки.

Сторінка даних ІМТ забезпечує централізовану платформу для відстеження та управління інформацією, пов’язаною з ІМТ пацієнтів (див. рис. 3.14).

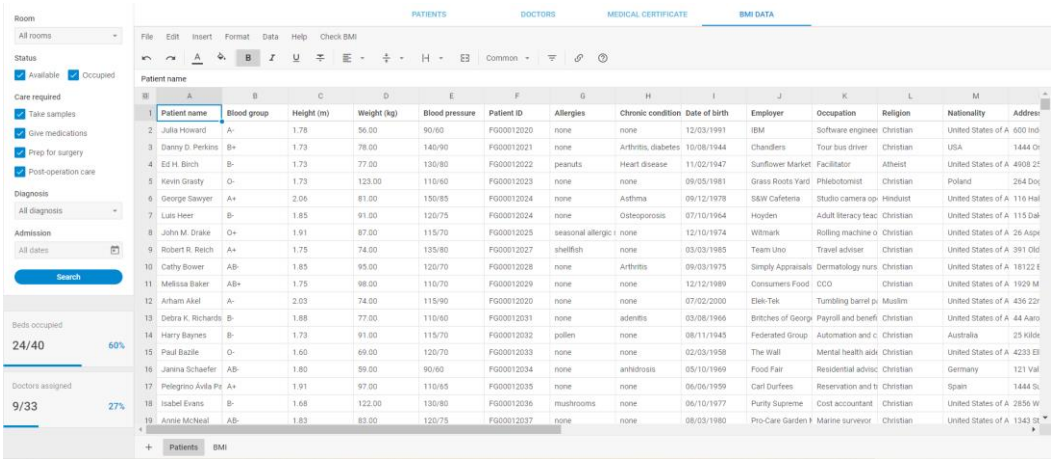


Рисунок 3.14 – Сторінка даних ІМТ

Для представлення даних пацієнта використовується табличний формат, що включає розрахунок ІМТ, групу крові, зріст, вагу, артеріальний тиск та інші відповідні дані. Інтегровані розширені інструменти редагування, що дозволяють користувачам імпортувати та експортувати дані ІМТ, виконувати аналіз даних та створювати звіти без особливих зусиль. Дизайн інтерфейсу зосереджений на візуалізації даних та зручності використання, що дозволяє користувачам ефективно інтерпретувати дані ІМТ та приймати обґрунтовані рішення щодо догляду за пацієнтами та ведення їхнього здоров'я.

Переваги UI/UX дизайну:

- орієнтований на користувача підхід до дизайну гарантує, що медичні працівники можуть легко орієнтуватися в системі, що призводить до більш приємного та ефективного користувацького досвіду;
- інтуїтивно зрозумілий макет та організована структура користувацького інтерфейсу спрощують такі складні завдання, як ведення пацієнтів, координація роботи лікарів, генерація медичних довідок та відстеження даних ІМТ, тим самим впорядковуючи робочі процеси;
- дизайн інтерфейсу включає чіткі написи, випадаючі меню та логічне групування інформації, що дозволяє користувачам легко знаходити та отримувати доступ до потрібних функцій та можливостей без плутанини;
- адаптивний дизайн забезпечує сумісність з різними пристроями та розмірами екранів, дозволяючи медичним працівникам безперешкодно працювати з

системою зі стаціонарних комп’ютерів, планшетів або мобільних пристроїв, що підвищує доступність та гнучкість;

- дизайн інтерфейсу оптимізує введення даних за допомогою зручних полів введення та інтуїтивно зрозумілих форм, що дозволяє медичним працівникам ефективно вводити, оновлювати та керувати інформацією;
- важлива інформація, наприклад, як дані про пацієнта, представлені чітко та на видному місці, що дозволяє медичним працівникам швидко та легко отримати доступ до важливих даних та інтерпретувати їх;
- дизайн полегшує аналіз даних і функції звітності, дозволяючи медичним працівникам аналізувати тенденції ІМТ, створювати змістовні звіти та приймати обґрунтовані рішення щодо догляду за пацієнтами та ведення їхнього здоров’я;
- поєднуючи зручні інтерфейси, оптимізовані робочі процеси та ефективні інструменти управління даними, UI/UX дизайн підвищує загальну ефективність медичних працівників, дозволяючи їм зосередитися на наданні якісної допомоги в умовах будинку для людей похилого віку.

Отже, створений UI/UX дизайн ставить на перше місце функціональність, ефективність та задоволеність користувачів, гарантуючи, що медичні працівники можуть легко орієнтуватися в системі та ефективно виконувати свої обов’язки в умовах будинку для людей похилого віку.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Веб-орієнтована програмна система для медсестер будинку літніх людей призначена для полегшення управління пацієнтами, медсестрами, ліками та процедурами. Система використовує стек MERN, який забезпечує надійне та масштабоване рішення для створення динамічних веб-додатків та складається з MongoDB, Express.js, React.js та Node.js.

MongoDB – це база даних NoSQL, яка зберігає дані у гнучкому JSON-подібному форматі BSON. Вона розроблена для масштабованості та продуктивності, що робить її ідеальною для обробки великих обсягів даних зі складними взаємозв'язками [18]. У програмній системі будинку літніх людей MongoDB використовується для зберігання всіх даних, пов'язаних з пацієнтами, медсестрами, ліками та процедурами. Колекції включають дані користувачів (як адміністраторів, так і медсестер), записи пацієнтів, інформацію про ліки та описи процедур. Ключові особливості:

- дизайн без схем забезпечує гнучкість у моделюванні даних;
- висока доступність та можливості горизонтального масштабування;
- ефективні запити та індексація для швидкого пошуку даних.

Express.js – це мінімальний та гнучкий фреймворк веб-додатків Node.js, який надає надійний набір функцій для створення веб- і мобільних додатків. Він полегшує створення серверної логіки та API [19]. Express.js слугує бекенд-фреймворком, обробляючи HTTP-запити, маршрутизацію та функції проміжного програмного забезпечення. Він надає API, які взаємодіють з базою даних MongoDB для виконання CRUD-операцій (створення, читання, оновлення, видалення) над даними. Ключові особливості:

- підтримка проміжного програмного забезпечення для обробки запитів, відповідей та маршрутизації;
- спрощена розробка API з вбудованими методами та утилітами;
- інтеграція з іншими модулями та інструментами Node.js.

React.js – це JavaScript бібліотека для побудови користувацьких інтерфейсів, зокрема односторінкових додатків, де дані динамічно змінюються з часом. Вона дозволяє створювати великі веб-додатки, які можуть ефективно оновлюватися та рендеритися у відповідь на зміни даних [20]. React.js використовується для створення фронтенду системи для будинку престарілих. Вона створює адаптивні та інтерактивні користувацькі інтерфейси як для адміністраторів, так і для медсестер. Компоненти включають форми для входу, інформаційні панелі, профілі пацієнтів, сторінки управління ліками та списки процедур. Ключові особливості:

- компонентна архітектура для багаторазового використання елементів інтерфейсу;
- віртуальний DOM для ефективного оновлення та рендерингу;
- одностороння прив’язка даних для передбачуваного потоку даних.

Node.js – це середовище виконання JavaScript, побудоване на JavaScript-движку Chrome V8. Він дозволяє використовувати JavaScript для написання серверного коду, що дає змогу створювати швидкі та масштабовані мережеві додатки [21]. Node.js надає середовище виконання для серверного коду, написаного на Express.js. Він обробляє асинхронні операції вводу/виводу, гарантуючи, що додаток залишається чуйним при високому навантаженні. Ключові особливості:

- неблокуюча, керована подіями архітектура для високої продуктивності;
- крос-платформна підтримка;
- широка екосистема модулів, доступних через npm (Node Package Manager).

Користувачі взаємодіють з додатком через веб-браузер. Фронтенд React.js взаємодіє з внутрішніми API через HTTP-запити. Сервер обробляє вхідні запити, виконує необхідні операції (наприклад, запити до бази даних) і надсилає відповіді назад на фронтенд. Бекенд взаємодіє з MongoDB для отримання, зберігання та оновлення даних. Кожен тип даних (пацієнти, медсестри, ліки, процедури) зберігається у відповідній колекції. Приклад функціональних можливостей:

- панель адміністратора надає інтерфейси для адміністратора (лікаря) для управління медсестрами, пацієнтами, ліками та процедурами.

Використовує компоненти React.js для форм і таблиць введення даних;

- аутентифікація реалізована з використанням JWT (JSON Web Tokens) для захисту маршрутів та управління сесіями користувачів. Express.js обробляє кінцеві точки входу/виходу;
- управління пацієнтами включає створення, оновлення, видалення та перегляд записів пацієнтів. Фронтенд надсилає запити до бекенду, який потім взаємодіє з MongoDB для виконання цих операцій;
- інформаційна панель медсестри, де медсестри можуть увійти в систему, щоб переглянути призначених їм пацієнтів, ліки та процедури. React.js відповідає за відображення та логіку взаємодії, а Express.js та Node.js керують отриманням та оновленням даних.

Реалізація панелі адміністратора для управління медсестрами, пацієнтами, медикаментами та процедурами використовує стек MERN. Ключові рішення були прийняті для забезпечення масштабованості, безпеки та простоти використання.

Гнучкий дизайн схеми MongoDB підтримує різноманітні вимоги до даних, тоді як React.js пропонує чуйний та динамічний користувацький інтерфейс. Компоненти React.js були використані для створення модульних, багаторазових частин інтерфейсу, що полегшує обслуговування та майбутню масштабованість. bcrypt.js використовувався для хешування паролів, щоб забезпечити безпечне зберігання облікових даних користувачів. RESTful API був розроблений для обробки CRUD-операцій, що дозволяє чітко розділити логіку фронтенду та бекенду. Панель адміністратора була розроблена адаптивною, що забезпечує зручність використання на різних пристроях і розмірах екранів. Для забезпечення цілісності даних та надання негайного зворотного зв’язку користувачам було реалізовано як фронтенд-, так і бекенд-валідацію.

Код нижче забезпечує основну функціональність бекенду для управління медсестрами в системі будинку престарілих за допомогою додатку Express.js. Система використовує стек MERN, і цей код обробляє різні CRUD-операції, пов’язані з медсестрами. Код починається з імпорту необхідних модулів: express для серверного фреймворку, bcryptjs для хешування паролів та моделі Nurse для

взаємодії з базою даних. Створюється новий об’єкт Express router для визначення маршрутів, пов’язаних з управлінням медсестрами:

```
const express = require('express');
const bcrypt = require('bcryptjs');
const Nurse = require('../models/Nurse');
const router = express.Router();
```

Перший визначений маршрут – це POST-маршрут за адресою /register для обробки реєстрації нових медсестер. Коли надходить запит на реєстрацію, код витягує дані про медсестру з тіла запиту, включаючи ім’я, ім’я користувача, пароль, електронну пошту, мобільний телефон, години роботи і адресу. Потім він перевіряє, чи існує в базі даних медсестра з таким іменем користувача, щоб уникнути дублікатів. Якщо медсестра не існує, створюється новий екземпляр медсестри з наданими даними. Пароль хешується за допомогою bcrypt для забезпечення безпеки, і нова медсестра зберігається в базі даних. Якщо процес є успішним, у відповідь надсилається повідомлення про успіх. Будь-які помилки під час цього процесу фіксуються і записуються в журнал, а також повертається повідомлення про помилку сервера:

```
router.post('/register', async (req, res) => {
  const { name, username, password, email, mobile, duty_hour, address } = req.body;
  try {
    let nurse = await Nurse.findOne({ username });
    if (nurse) {
      return res.status(400).json({ msg: 'Nurse already exists' });  }
    nurse = new Nurse({
      name,
```

```
        username,
        password,
        email,
        mobile,
        duty_hour,
        address });
      const salt = await bcrypt.genSalt(10);
      nurse.password = await bcrypt.hash(password, salt);
      await nurse.save();
      res.status(201).json({ msg: 'Nurse registered
successfully' });
    } catch (err) {
      console.error(err.message);
      res.status(500).send('Server error'); }});
```

Другий маршрут – це маршрут GET в корені (/) для отримання всіх медсестер з бази даних. При зверненні до цього маршруту код запитує базу даних, щоб знайти всі записи про медсестер, і повертає цей список у форматі JSON. Якщо під час цієї операції виникає помилка, вона реєструється, і надсилається повідомлення про помилку сервера:

```
router.get('/', async (req, res) => {
  try {
    const nurses = await Nurse.find();
    res.json(nurses);
  } catch (err) {
    console.error(err.message);
    res.status(500).send('Server error');  }});
```

Третій маршрут – це маршрут PUT за адресою /:id для оновлення даних про медсестер. Коли робиться запит на оновлення медсестри, код витягує оновлені дані з тіла запиту. Потім він знаходить медсестру за наданим ідентифікатором у базі даних. Якщо медсестра існує, код оновлює дані медсестри новими даними, якщо вони були надані; в іншому випадку він зберігає існуючі дані. Оновлені дані про медсестру зберігаються в базі даних, а у відповіді повертається повідомлення про успіх. Помилки обробляються аналогічно, шляхом їх реєстрації та повернення повідомлення про помилку сервера:

```
router.put('/:id', async (req, res) => {
  const { name, email, mobile, duty_hour, address } = req.body;
  try {
    let nurse = await Nurse.findById(req.params.id);
    if (!nurse) return res.status(404).json({ msg: 'Nurse not found' });
    nurse.name = name || nurse.name;
    nurse.email = email || nurse.email;
    nurse.mobile = mobile || nurse.mobile;
    nurse.duty_hour = duty_hour || nurse.duty_hour;
    nurse.address = address || nurse.address;
    await nurse.save();
    res.json({ msg: 'Nurse updated successfully' });
  } catch (err) {
    console.error(err.message);
    res.status(500).send('Server error'); }});
```

Останнім маршрутом є маршрут DELETE за адресою /:id для видалення медсестри. Цей маршрут знаходить медсестру за наданим ідентифікатором і видаляє запис з бази даних. Після успішного видалення у відповідь буде надіслано

повідомлення про успіх. Будь-які помилки, що виникають під час цього процесу, реєструються, і повертається повідомлення про помилку сервера:

```
router.delete('/:id', async (req, res) => {
  try {
    await Nurse.findByIdAndRemove(req.params.id);
    res.json({ msg: 'Nurse deleted successfully' });
  } catch (err) {
    console.error(err.message);
    res.status(500).send('Server error'); }});
module.exports = router;
```

Отже, цей код забезпечує повний набір внутрішніх функцій для управління медсестрами в системі, включаючи реєстрацію, пошук, оновлення та видалення записів про медсестер. Використання bcrypt гарантує безпечне зберігання паролів, а також належну обробку помилок для управління та реєстрації будь-яких проблем, що виникають під час роботи з базою даних. Використання Mongoose для взаємодії з базами даних забезпечує ефективне та надійне управління даними.

При розробці автентифікації було прийнято кілька ключових рішень для забезпечення безпеки та ефективного управління сесіями користувачів. Основним методом, обраним для автентифікації, є використання JSON веб-токенів (JWT). JWT забезпечують безпечний та незалежний спосіб обробки автентифікації, що робить їх ідеальними для веб-додатків, де масштабованість та безпека мають першорядне значення. Express.js використовується для управління логікою на стороні сервера, включаючи обробку кінцевих точок входу і виходу.

Використання JWT дозволяє використовувати механізм автентифікації без стану, де токени генеруються після успішного входу в систему та необхідні для доступу до захищених маршрутів. Це усуває необхідність зберігання сесій на стороні сервера та спрощує масштабування програми. Паролі користувачів хешуються за допомогою bcrypt перед тим, як зберігатися в базі даних. Це гарантує,

що навіть якщо база даних буде скомпрометована, паролі залишаться в безпеці. Маршрути, які вимагають автентифікації, захищені проміжним програмним забезпеченням, яке перевіряє JWT, наданий в заголовках запитів. Це гарантує, що тільки автентифіковані користувачі можуть отримати доступ до конфіденційної інформації та функцій.

Наведений код нижче налаштовує функціональність JWT (JSON Web Token) та проміжне програмне забезпечення для автентифікації та захисту маршрутів у веб-орієнтованій програмній системі для медсестер будинку людей похилого віку.

Код імпортує бібліотеку jsonwebtoken для роботи з функціоналом JWT та config для доступу до налаштувань конфігурації. Ключ jwtSecret з конфігураційного файлу використовується для підписання та перевірки токенів:

```
const jwt = require('jsonwebtoken');
const config = require('config');
```

Функція auth проміжного програмного забезпечення призначена для перехоплення вхідних HTTP-запитів і автентифікації користувачів на основі JWT, наданого в заголовку запиту. В рамках функції auth токен витягується із заголовка запиту за допомогою методу req.header(). Очікується, що токен буде включено в заголовок x-auth-token:

```
function auth(req, res, next) {
  const token = req.header('x-auth-token');
```

Витягнутий токен перевіряється за допомогою jwt.verify(). Якщо токен дійсний і його термін дії не закінчився, він декодується для отримання інформації про користувача, що міститься в ньому. Якщо токен дійсний, декодована інформація про користувача додається до об’єкта запиту (req.user) для подальшої обробки наступним проміжним програмним забезпеченням або обробниками маршрутів:

```
      if (!token) {
        return res.status(401).json({ msg: 'No token, authorization denied' }); }
      try {
        const decoded = jwt.verify(token, config.get('jwtSecret'));
        req.user = decoded.user;
        next();
```

Якщо токен відсутній або недійсний, клієнту надсилається відповідна відповідь про помилку (код стану HTTP 401), що вказує на несанкціонований доступ:

```
      } catch (err) {
        res.status(401).json({ msg: 'Token is not valid' }); // Unauthorized access  }}
```

Потім функція проміжного програмного забезпечення авторизації експортується для використання в інших частинах програми, що дозволяє захистити маршрути та автентифікувати їх за допомогою JWT:

```
module.exports = auth;
```

Код нижче реалізує кінцеву точку входу для медсестер у програмній системі будинку престарілих.

Маршрут входу налаштовується за допомогою router.post() для обробки HTTP POST запитів до '/login'. Для перевірки тіла запиту використовується проміжне програмне забезпечення експрес-валідатора. Він перевіряє, чи вказано ім’я користувача та чи існує пароль:

```
const express = require('express');
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const config = require('config');
const { check, validationResult } = require('express-
validator');
const Nurse = require('../models/Nurse');
const router = express.Router();
router.post(
  '/login',
  [
    check('username', 'Please include a valid
username').not().isEmpty(),
    check('password', 'Password is
required').exists(),
  ],
```

У обробнику маршруту ім’я користувача та пароль витягуються з тіла запиту. Потім запис медсестри витягується з бази даних на основі наданого імені користувача:

```
  async (req, res) => {
    const { username, password } = req.body;
```

Якщо медсестра з вказаним іменем користувача існує, використовується bcrypt.js для порівняння хешованого пароля, що зберігається в базі даних, з паролем, вказаним у запиті:

```
    try {
      let nurse = await Nurse.findOne({ username });
```

```
          if (!nurse) {
            return res.status(400).json({ msg: 'Invalid
credentials' });}
          const isMatch = await bcrypt.compare(password,
nurse.password);
          if (!isMatch) {
            return res.status(400).json({ msg: 'Invalid
credentials' });}
          const payload = {
            user: {
              id: nurse.id,   },};
```

Якщо паролі збігаються, створюється JWT-файл, що містить ідентифікатор медсестри. Це корисне навантаження підписується за допомогою секретного ключа JWT і генерується токен. Токен налаштований на закінчення терміну дії через 1 годину (3600 секунд). Згенерований токен надсилається назад клієнту у вигляді JSON-відповіді. Повідомлення про помилки повертаються у випадку невірних облікових даних або помилок сервера. Якщо під час процесу виникає помилка, разом із загальним повідомленням про помилку надсилається код стану 500:

```
          jwt.sign(
            payload,
            config.get('jwtSecret'),
            { expiresIn: 3600 },
            (err, token) => {
              if (err) throw err;
              res.json({ token });});
        } catch (err) {
          console.error(err.message);
          res.status(500).send('Server error'); }});
```

```
module.exports = router;
```

За допомогою проміжного програмного забезпечення авторизації можна захистити маршрути, які потребують автентифікації. Це гарантує, що тільки користувачі з дійсним токеном можуть отримати доступ до цих маршрутів. У цьому випадку він реалізує маршрут для отримання інформації про медсестру, яка увійшла в систему.

Маршрут налаштовується за допомогою router.get() для обробки HTTP GET-запитів до '/me'. Проміжне програмне забезпечення auth включається як другий аргумент в обробник маршруту. Це проміжне програмне забезпечення гарантує, що тільки автентифіковані користувачі з дійсним токеном можуть отримати доступ до маршруту:

```
const express = require('express');
const router = express.Router();
const auth = require('../middleware/auth');
const Nurse = require('../models/Nurse');
router.get('/me', auth, async (req, res) => {
  try {
```

Усередині функції обробника маршруту визначено асинхронну функцію для обробки запиту. Вона намагається знайти медсестру в базі даних на основі ідентифікатора медсестри, витягнутого з корисного навантаження JWT (req.user.id). Метод Nurse.findById() використовується для пошуку запису медсестри в базі даних. До методу select() передається аргумент '-password', щоб виключити поле пароля зі знайденого об'єкта медсестри. Якщо медсестра знайдена, її дані (за винятком пароля) повертаються у вигляді JSON-відповіді. Повідомлення про помилки записуються в консоль, а в разі виникнення помилок під час виконання процесу повертається загальне повідомлення про помилку сервера з кодом стану 500:

```
        const nurse = await
Nurse.findById(req.user.id).select('-password');
        res.json(nurse);
      } catch (err) {
        console.error(err.message);
        res.status(500).send('Server error'); }});
    module.exports = router;
```

Загалом, функція проміжного програмного забезпечення auth отримує токен із заголовків запиту та перевіряє його за допомогою бібліотеки jsonwebtoken. Якщо токен дійсний, вона приєднує розшифровану інформацію про користувача до об’єкта запиту, дозволяючи доступ до захищеного маршруту. Маршрут /login перевіряє надані ім’я користувача та пароль у базі даних. Якщо облікові дані дійсні, генерується JWT, який надсилається назад клієнту. Цей токен потім використовується для подальших запитів до захищених маршрутів. Маршрути типу /me захищені за допомогою проміжного програмного забезпечення авторизації. Цей маршрут отримує дані медсестри, яка увійшла в систему, з бази даних, за винятком пароля, і повертає їх користувачу.

Отже, застосування JWT для автентифікації в цій системі забезпечує безпечну та ефективну обробку користувацьких сесій. Express.js забезпечує надійний фреймворк для управління кінцевими точками, а bcrypt гарантує безпечне зберігання паролів. Поєднання цих технологій пропонує масштабоване та безпечне рішення для управління медсестрами, пацієнтами, ліками та процедурами в системі будинку престарілих.

У модулі управління пацієнтами реалізовано функціонал для створення, оновлення, видалення та перегляду записів пацієнтів. Це передбачає взаємодію між фронтенд- і бекенд-компонентами. Коли користувач взаємодіє з фронтенд-інтерфейсом, наприклад, надсилає форму для додавання нового пацієнта або оновлює дані про існуючого пацієнта, фронтенд надсилає HTTP-запити до бекенд-

сервера. Внутрішній сервер, побудований на Express.js, обробляє ці запити, виконуючи відповідні операції над базою даних MongoDB, в якій зберігаються записи про пацієнтів.

Нижче наведено код з бекенду, що відповідає за обробку запитів, пов'язаних з управлінням пацієнтами. Код ініціалізує експрес-маршрутизатор за допомогою express.Router(). Маршрут обробляє POST-запити для створення нового запису про пацієнта. Він отримує дані пацієнта з тіла запиту, створює новий екземпляр моделі Patient за допомогою схеми Mongoose, зберігає новий запис про пацієнта в базі даних і відповідає JSON-представленням новостворених даних про пацієнта. Якщо під час виконання запиту виникає помилка, він надсилає відповідь про помилку сервера:

```
const express = require('express');
const router = express.Router();
const Patient = require('../models/Patient');
router.post('/', async (req, res) => {
  try {
    const newPatient = new Patient(req.body);
    await newPatient.save();
    res.status(201).json(newPatient);
  } catch (error) {
    console.error(error.message);
    res.status(500).send('Server Error');   }});
```

Далі маршрут обробляє PUT-запити на оновлення існуючого запису про пацієнта. Він знаходить запис пацієнта за ідентифікатором, оновлює його даними з тіла запиту і повертає відповідь у вигляді JSON-представлення оновлених даних пацієнта. Якщо запис про пацієнта не знайдено, він надсилає відповідь 404. Якщо під час процесу виникає помилка, він надсилає відповідь про помилку сервера:

```
router.put('/:id', async (req, res) => {
  try {
    const patient = await Patient.findByIdAndUpdate(req.params.id, req.body, { new: true });
    if (!patient) {
      return res.status(404).json({ msg: 'Patient not found' });    }
    res.json(patient);
  } catch (error) {
    console.error(error.message);
    res.status(500).send('Server Error');   }});
```

Потім маршрут обробляє запити DELETE на видалення існуючих записів про пацієнтів. Він знаходить запис про пацієнта за ідентифікатором, видаляє його з бази даних і повертає повідомлення про успішне виконання. Якщо запис про пацієнта не знайдено, він надсилає відповідь 404. Якщо під час процесу виникає помилка, він надсилає відповідь про помилку сервера:

```
router.delete('/:id', async (req, res) => {
  try {
    const patient = await Patient.findByIdAndDelete(req.params.id);
    if (!patient) {
```

Продовження коду:

```
      return res.status(404).json({ msg: 'Patient not found' });    }
    res.json({ msg: 'Patient deleted successfully' });
  } catch (error) {
```

```
    console.error(error.message);
    res.status(500).send('Server Error');   }});
```

Останній маршрут обробляє GET-запити для отримання всіх записів про пацієнтів з бази даних. Він отримує всі записи пацієнтів за допомогою методу Patient.find() і повертає JSON-представлення списку записів пацієнтів. Якщо під час процесу виникає помилка, він надсилає відповідь про помилку сервера. Об’єкт маршрутизатора експортується, щоб зробити його доступним для використання в інших частинах програми:

```
router.get('/', async (req, res) => {
  try {
    const patients = await Patient.find();
    res.json(patients);
  } catch (error) {
    console.error(error.message);
    res.status(500).send('Server Error'); }});
module.exports = router;
```

Отже, завдяки добре структурованим маршрутам, надійній обробці помилок та безперебійній інтеграції з базами даних, код полегшує ефективне управління записами пацієнтів, сприяючи підвищенню загальної ефективності та зручності використання програмної системи.

Рішення про впровадження інформаційної панелі медсестри слугує для надання медсестрам централізованої платформи для доступу до відповідної інформації про закріплених за ними пацієнтів, ліки та процедури. Ця панель підвищує ефективність та покращує догляд за пацієнтами, дозволяючи медсестрам швидко отримувати доступ до важливих даних та керувати ними. Для досягнення цієї функціональності було обрано поєднання React.js для фронтенд-розробки та Express.js/Node.js для бекенд-розробки. React.js добре підходить для створення

інтерактивних користувацьких інтерфейсів, що робить його ідеальним для проектування відображення та логіки взаємодії інформаційної панелі медсестри. З іншого боку, Express.js та Node.js використовуються для обробки бекенд-операцій, таких як отримання та оновлення даних з сервера.

Нижче наведено код, який відповідає за рендеринг інтерфейсу інформаційної панелі медсестри. Код починається з імпорту необхідних модулів з React та Axios, популярної клієнтської бібліотеки HTTP. Хуки useState та useEffect також імпортуються з React. Потім оголошується функціональний компонент NurseDashboard. Усередині компонента NurseDashboard три змінні стану ініціалізуються за допомогою хука useState. assignedPatients зберігає масив даних про призначених пацієнтів. medications зберігає масив даних про ліки. procedures зберігає масив даних процедур. Ці змінні стану будуть оновлюватися динамічно по мірі отримання даних з сервера;

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
const NurseDashboard = () => {
  const [assignedPatients, setAssignedPatients] = useState([]);
  const [medications, setMedications] = useState([]);
  const [procedures, setProcedures] = useState([]);
  useEffect(() => {
```

Хук useEffect використовується для виконання побічних ефектів у функціональних компонентах. У цьому випадку він використовується для отримання даних з сервера, коли компонент монтується. Всередині useEffect визначена функція fetchData, яка асинхронно отримує дані з трьох різних кінцевих точок API (/api/nurse/patients, /api/nurse/medications, /api/nurse/procedures) за допомогою Axios. Після успішного отримання відповідні змінні стану оновлюються отриманими даними:

```
        const fetchData = async () => {
          try {
            const patientsResponse = await
axios.get('/api/nurse/patients');
            const medicationsResponse = await
axios.get('/api/nurse/medications');
            const proceduresResponse = await
axios.get('/api/nurse/procedures');
```

Продовження коду:

```
        setAssignedPatients(patientsResponse.data);
            setMedications(medicationsResponse.data);
        setProcedures(proceduresResponse.data);
          } catch (error) {
            console.error('Error fetching data:', error);
}    };
         fetchData();
       }, []);
```

JSX, що повертається компонентом NurseDashboard, відображає інтерфейс інформаційної панелі медсестри. Він складається з трьох розділів. Assigned Patients відображає список призначених пацієнтів, отриманий зі змінної стану assignedPatients. Medications відображає список медикаментів, отриманий зі змінної стану medications. Procedures відображає список процедур, отриманих зі змінної стану procedures. Кожен список генерується динамічно за допомогою функції map для ітерації над відповідними масивами даних. У кожному елементі списку (<li>) ключовому пропу (унікальний ідентифікатор, який React використовує для ефективного управління та оновлення елементів у списку) надається унікальний ідентифікатор (patient.id, medication.id, procedure.id), щоб допомогти React

ефективно ідентифікувати елементи для оновлення. Це має вирішальне значення для оптимізації продуктивності рендерингу, особливо при роботі з динамічними списками. Компонент NurseDashboard експортується за замовчуванням, що робить його доступним для використання в інших частинах програми:

```
      return (
        <div className="nurse-dashboard">
          <h2>Assigned Patients</h2> <ul>
            {assignedPatients.map((patient) => (
              <li key={patient.id}>{patient.name}</li>))}
          </ul>
          <h2>Medications</h2> <ul>
            {medications.map((medication) => (
              <li
key={medication.id}>{medication.name}</li>))}
          </ul>
          <h2>Procedures</h2>
          <ul>
            {procedures.map((procedure) => (
              <li key={procedure.id}>{procedure.name}</li>
))}
          </ul></div>  );};
    export default NurseDashboard;
```

Отже, цей код демонструє реалізацію інформаційної панелі медсестри на React, де дані отримуються асинхронно з сервера і динамічно відображаються у користувацькому інтерфейсі. Він слідує шаблону функціональних компонентів React і використовує хуки для управління станом і виконання побічних ефектів.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Модульне тестування передбачає перевірку правильності функціонування кожного компонента системи окремо [22]. Для веб-орієнтованої програмної системи для медсестер будинку літніх людей були протестовані такі ключові модулі, як автентифікація користувачів, управління медсестрами, управління пацієнтами, управління медикаментами та управління процедурами. Тестування проводилося у Visual Studio Code з використанням комплексного фреймворку для тестування.

Основне завдання модуля автентифікації користувачів полягає в тому, щоб забезпечити безпечну реєстрацію, вхід і керування сесіями користувачів за допомогою веб-токенів JSON (JWT). Це передбачає перевірку того, що нові користувачі можуть реєструватися без конфліктів, що механізми входу в систему функціонують правильно з дійсними та недійсними обліковими даними, і що генерація та перевірка токенів обробляються належним чином для забезпечення безпечних маршрутів.

Тестова функція test_user_registration_success() перевіряє успішність реєстрації нового користувача. Цей тест перевіряє весь процес реєстрації, щоб переконатися, що до системи додано нового користувача з дійсними обліковими даними. Тест імітує реєстрацію користувача шляхом надсилання POST-запиту з даними користувача на кінцеву точку реєстрації. Потім він перевірить, що відповідь свідчить про успішну реєстрацію та що дані користувача правильно збережені в базі даних. Тестова функція test_user_registration_duplicate() перевіряє правильність обробки дублікатів реєстрацій користувачів. Цей тест гарантує, що система запобігає повторним реєстраціям користувачів. Він імітує спробу реєстрації з використанням вже зареєстрованого імені користувача або адреси електронної пошти та перевіряє, чи повертається відповідне повідомлення про помилку, і чи не створюється дублікат запису в базі даних. Тестова функція test_user_login_success() перевіряє, що зареєстрований користувач може увійти з правильними обліковими даними. Цей тест перевіряє, чи може користувач успішно увійти в систему, використовуючи правильні облікові дані. Він надсилає POST-

запит на кінцеву точку входу з правильним ім’ям користувача та паролем, а потім перевіряє, чи повернувся у відповіді токен JWT, що свідчить про успішну автентифікацію. Тестова функція test_user_login_invalid_credentials() перевіряє, що вхід не вдається з неправильними обліковими даними. Цей тест перевіряє, чи система коректно відмовляє в доступі при введенні невірних облікових даних. Він імітує спробу входу з неправильним паролем і перевіряє, чи повертається відповідне повідомлення про помилку без видачі токена JWT. Тестова функція test_token_verification() перевіряє, що токени JWT згенеровано та верифіковано правильно. Цей тест гарантує, що токени JWT, видані під час входу в систему, є дійсними та можуть бути використані для доступу до захищених маршрутів. Він включає в себе вхід в систему для отримання токена, потім використання цього токена для доступу до захищеної кінцевої точки та перевірку того, що доступ надано.

Виконання цих тестових функцій зображено на рисунку 5.1.

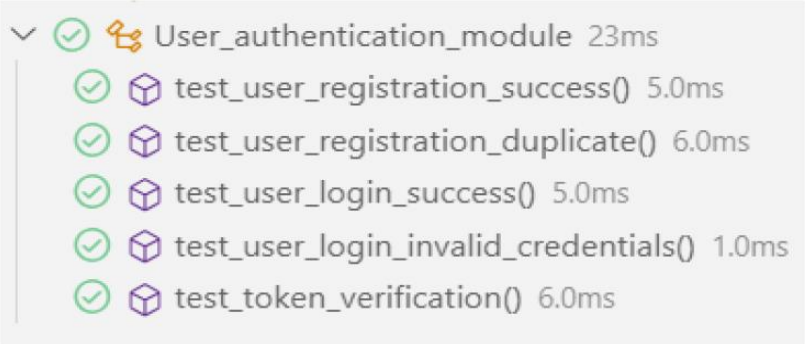


Рисунок 5.1 – Тестування модуля автентифікації користувачів

Тестування модуля автентифікації користувачів для веб-орієнтованої програмної системи, призначеної для медсестер будинку людей похилого віку, було успішним. Всі критичні функціональні можливості, включаючи реєстрацію користувачів, вхід та перевірку токенів, були ретельно протестовані та підтверджені. Тести підтвердили, що нові користувачі можуть реєструватися без проблем, дублікати реєстрацій обробляються належним чином, і користувачі можуть входити в систему з дійсними обліковими даними, а недійсні облікові дані

коректно відкидаються. Крім того, токени JWT, що генеруються під час входу, були перевірені на точність та здатність ефективно захищати захищені маршрути. В цілому, успішне тестування модуля аутентифікації користувачів демонструє ефективність прийнятих програмних рішень, забезпечуючи безпечну та ефективну систему управління медсестрами, пацієнтами, медикаментами та процедурами в умовах будинку для людей похилого віку.

Основною метою тестування модуля управління медсестрами є перевірка того, що всі CRUD-операції для записів медсестер функціонують коректно та як очікується. Це включає в себе забезпечення можливості безперешкодного додавання, пошуку, оновлення та видалення медсестер, а також доступ до всіх записів медсестер у разі потреби.

Тестова функція test_create_nurse() перевіряє, що медсестра може бути створена з правильними даними. Цей тест перевіряє створення нового запису про медсестру. Він надсилає POST-запит до кінцевої точки, відповідальної за створення медсестри, з валідними даними медсестри. Очікуваним результатом є успішна відповідь, яка вказує на те, що медсестра була створена, а також підтвердження того, що дані нової медсестри коректно зберігаються в базі даних. Тестова функція test_read_nurse() перевіряє коректності зчитування інформації про медсестру. Ця функція тестує отримання даних конкретної медсестри за її унікальним ідентифікатором. Надсилаючи GET-запит до відповідної кінцевої точки з ідентифікатором медсестри, функція перевіряє, чи відповідають повернуті дані очікуваній інформації про медсестру. Це гарантує, що система може точно отримати індивідуальні записи про медсестер. Тестова функція test_update_nurse() перевіряє, що дані про медсестру можна оновити. Цей тест оцінює функціональність оновлення, надсилаючи PUT-запит до кінцевої точки, відповідальної за оновлення записів медсестер. Він включає нові дані для існуючої медсестри та перевіряє, чи зміни правильно відображені в базі даних. Функція також гарантує, що відповідь вказує на успішне оновлення. Тестова функція test_delete_nurse() перевіряє, що запис про медсестру можна видалити. Метою цього тесту є перевірка функціональності видалення. Функція надсилає запит

DELETE з ідентифікатором конкретного медпрацівника до відповідної кінцевої точки. Потім вона підтверджує, що запис про медсестру видалено з бази даних і що відповідь підтверджує успішне видалення. Тестова функція test_read_all_nurses() перевіряє, чи можна отримати всі записи про медсестер. Цей тест перевіряє можливість отримати всі записи про медсестер, що зберігаються в базі даних. Відправляючи GET-запит до кінцевої точки, яка отримує всіх медсестер, функція перевіряє, чи відповідь містить повний список медсестер. Вона також перевіряє відповідність отриманих даних записам у базі даних.

Виконання цих тестових функцій зображено на рисунку 5.2.

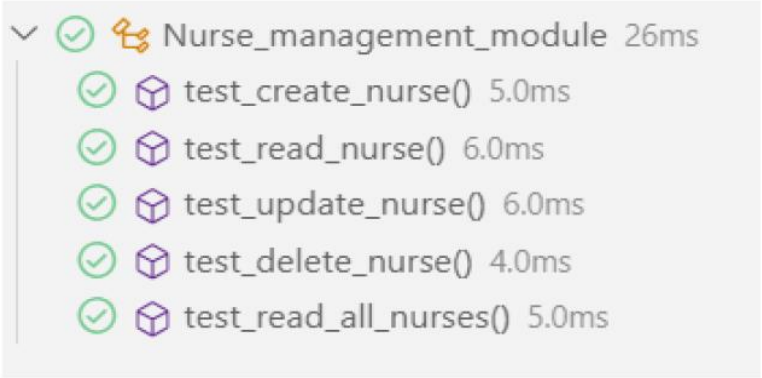


Рисунок 5.2 – Тестування модуля управління медсестрами

Тестування модуля управління медсестрами пройшло успішно. Тести підтвердили, що система може точно створювати, отримувати, оновлювати та видаляти записи про медсестер, гарантуючи, що всі функції працюють належним чином. Також було перевірено можливість отримати всі записи про медсестер з бази даних, що підтвердило надійність та стійкість модуля. Ці успішні тести дають впевненість у тому, що модуль управління медсестрами повністю функціональний та здатний обробляти операції з даними, необхідні в умовах будинку престарілих.

Основною метою тестування модуля управління пацієнтами є забезпечення точного та надійного управління записами пацієнтів. Це включає перевірку того, що система може ефективно виконувати операції CRUD з даними пацієнтів. Крім того, тестування має на меті підтвердити, що система може точно отримувати

інформацію про пацієнта та правильно відображати її для користувачів. Зрештою, мета полягає в тому, щоб гарантувати, що модуль управління пацієнтами функціонує належним чином, надаючи медсестрам необхідні інструменти для ефективного та точного управління записами пацієнтів.

Тестова функція test_create_patient() перевіряє, чи може система успішно створити новий запис про пацієнта за умови надання правильних даних. Вона перевіряє функціональність додавання пацієнта до системи, гарантуючи, що всі необхідні поля зберігаються належним чином. Мета тестової функції test_read_patient() – підтвердити, що дані про пацієнта можуть бути точно отримані з системи. Вона перевіряє, чи може система правильно отримати та відобразити інформацію про пацієнта на основі наданого ідентифікатора пацієнта або інших ідентифікаторів. Тестова функція test_update_patient() перевіряє, що система може належним чином оновити інформацію про пацієнта, коли це необхідно. Він перевіряє функціональність модифікації існуючих записів пацієнта, таких як оновлення демографічної інформації, історії хвороби або планів лікування. Мета тестової функції test_delete_patient() – перевірити, чи може система успішно видаляти записи про пацієнтів за запитом. Вона оцінює, чи може система належним чином обробляти процес видалення, гарантуючи, що дані пацієнта будуть видалені безпечно і безповоротно. Тестова функція test_read_all_patients() має на меті підтвердити, що система може отримати список всіх записів про пацієнтів, що зберігаються в базі даних. Вона перевіряє функціональність отримання та відображення декількох записів про пацієнтів, гарантуючи, що система може ефективно обробляти та представляти великі набори даних.

Виконання цих тестових функцій зображено на рисунку 5.3.

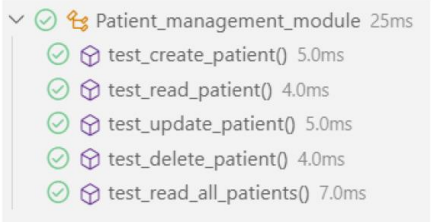


Рисунок 5.3 – Тестування модуля управління пацієнтами

Після ретельного тестування модуль "Управління пацієнтами" було визнано таким, що успішно виконує поставлені перед ним завдання. Модуль продемонстрував надійну функціональність у виконанні CRUD-операцій для записів пацієнтів, включаючи створення, зчитування, оновлення та видалення даних пацієнтів. Система точно витягувала дані про пацієнтів і коректно відображала їх користувачам. Успішне тестування модуля управління пацієнтами гарантує, що медсестри зможуть ефективно управляти записами пацієнтів у веб-орієнтованій програмній системі. Ця можливість має вирішальне значення для збереження точної та актуальної інформації про пацієнтів, що в кінцевому підсумку сприяє кращому догляду та лікуванню. Також перевірена функціональність модуля управління пацієнтами дають впевненість у загальній надійності та ефективності програмної системи, сприяючи її потенційному розгортанню та використанню в реальних умовах.

Основна мета тестування модуля управління лікарськими засобами – перевірити, чи може модуль ефективно виконувати операції CRUD над записами про лікарські засоби. Це включає в себе забезпечення коректного створення, пошуку, оновлення та видалення ліків у системі. Тестування має на меті підтвердити здатність модуля точно та ефективно управляти даними, пов’язаними з лікарськими засобами, гарантуючи, що медсестри та інші авторизовані користувачі можуть безперешкодно та безпомилково взаємодіяти з медичними картками.

Тестова функція test_create_medication() гарантує, що лікарський засіб може бути успішно створений у системі. Вона перевіряє функціональність додавання нового препарату з правильними даними, включаючи його назву, опис та іншу релевантну інформацію. Тестова функція test_read_medication() перевіряє, чи можна точно отримати дані про лікарський засіб із системи. Вона перевіряє здатність системи отримувати інформацію про ліки на основі певних критеріїв або ідентифікаторів і гарантує, що отримані дані відповідають очікуваним значенням. Тестова функція test_update_medication() підтверджує, що дані про ліки можуть бути оновлені за потреби. Вона тестує здатність системи змінювати існуючі записи

про лікарські засоби, включаючи зміну їх назви, опису або будь-яких інших атрибутів, і перевіряє, що оновлення коректно відображаються в системі. Тестова функція test_delete_medication() гарантує, що запис про лікарський засіб може бути успішно видалений з системи. Вона тестує функціонал для повного видалення запису про лікарський засіб, гарантуючи, що процес видалення працює належним чином, не спричиняючи жодних невідповідностей чи помилок у даних. Тестова функція test_read_all_medications() перевіряє, чи всі записи про ліки можна отримати з системи. Вона тестує здатність системи отримати список всіх ліків, що зберігаються в базі даних, гарантуючи, що жоден препарат не буде пропущений, а отриманий список буде повним і точним.

Виконання цих тестових функцій зображено на рисунку 5.4.

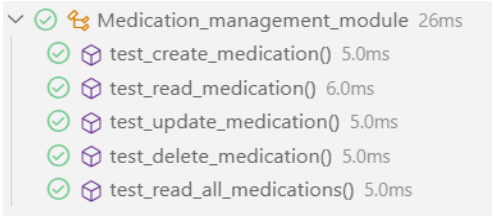


Рисунок 5.4 – Тестування модуля управління лікарськими засобами

Тестування модуля управління медикаментами пройшло успішно. Всі CRUD-операції з медикаментозними записами були ретельно протестовані, і виявилося, що вони працюють так, як очікувалося. Модуль дозволяє створювати, знаходити, оновлювати та видаляти медикаментозні записи без жодних проблем. Це гарантує, що медсестри та інші уповноважені користувачі можуть ефективно управляти даними, пов’язаними з медикаментами, в системі. Загалом, модуль управління медикаментами відповідає вимогам та функціонує належним чином, сприяючи підвищенню загальної функціональності та ефективності веб-орієнтованої програмної системи для медсестер будинку людей похилого віку.

Основна мета тестування модуля управління процедурами полягає в тому, щоб переконатися, що всі функції, пов’язані з управлінням записами процедур у веб-орієнтованій програмній системі для медсестер у будинку престарілих,

працюють належним чином. Це включає тестування можливості створення, читання, оновлення та видалення записів про процедури, а також перевірку точного пошуку деталей процедур. Мета – підтвердити, що система ефективно управляє даними про процедури, дозволяючи медсестрам отримувати доступ, оновлювати та зберігати інформацію про медичні процедури, проведені пацієнтам.

Тестова функція test_create_procedure() перевіряє функціональність створення нової процедури в системі. Вона гарантує, що при наданні правильних даних новий запис процедури буде успішно створено та збережено в базі даних. Тестова функція test_read_procedure() перевіряє можливість точного отримання деталей процедури з бази даних. Вона гарантує, що за запитом система коректно отримує та повертає деталі конкретної процедури. Тестова функція test_update_procedure() підтверджує, що система може оновлювати існуючі записи процедур новою інформацією. Вона гарантує, що при внесенні змін до реквізитів процедури, ці зміни будуть точно відображені в базі даних. Тестова функція test_delete_procedure() гарантує, що система може видаляти записи процедур, коли їй це доручено. Вона перевіряє, що при виконанні операції видалення вказаний запис процедури видаляється з бази даних без помилок. Тестова функція test_read_all_procedures() перевіряє здатність системи отримувати всі записи процедур, що зберігаються в базі даних.

Виконання цих тестових функцій зображено на рисунку 5.5.

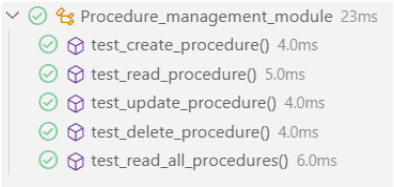


Рисунок 5.5 – Тестування модуля управління процедурами

Тестування модуля управління процедурами пройшло успішно, підтвердивши, що CRUD-операції для записів процедур функціонують належним чином. Кожен тестовий кейс, включаючи створення, читання, оновлення та

видалення процедур, був виконаний без помилок. Модуль продемонстрував здатність ефективно управляти записами процедур, дозволяючи медсестрам виконувати необхідні дії з даними процедур у програмній системі. Загалом, модуль "Управління процедурами" пройшов валідацію на відповідність вимогам та специфікаціям, встановленим для цього модуля.

Для модуля інформаційної панелі медсестри основною метою тестування є перевірка правильності відображення призначених пацієнтів, ліків і процедур на інформаційній панелі. Це передбачає перевірку того, що дані, отримані з серверної частини, точно відображаються на інтерфейсі панелі.

Тестова функція test_dashboard_fetch_assigned_patients() має на меті перевірити, що інформаційна панель медсестри отримує та відображає призначених пацієнтів коректно. Вона перевіряє, чи дані про пацієнта, отримані з внутрішнього API, точно відображаються в інтерфейсі дашборду, включаючи імена, ідентифікатори та інші важливі дані пацієнта. Мета тестової функції test_dashboard_fetch_medications() – підтвердити, що інформаційна панель медсестри отримує та відображає ліки правильно. Вона гарантує, що дані про ліки, отримані з внутрішнього API, правильно відображаються на дашборді, включаючи назви ліків, дозування та будь-яку іншу відповідну інформацію. Тестова функція test_dashboard_fetch_procedures() має на меті переконатися, що інформаційна панель медсестри отримує та відображає процедури точно. Вона перевіряє, що дані процедур, отримані з бекенду, правильно відображаються на інтерфейсі інформаційної панелі, включно з назвами процедур, описами та іншими важливими деталями.

Виконання цих тестових функцій зображено на рисунку 5.6.

Рисунок 5.6 – Тестування модуля інформаційної панелі медсестри

Тестування модуля "Інформаційна панель медсестри" пройшло успішно, підтвердивши, що модуль функціонує належним чином. Модуль надає медсестрам точний та надійний огляд призначених їм пацієнтів, ліків і процедур. Під час тестування було перевірено, що інформаційна панель правильно отримує та відображає призначених пацієнтів, ліки та процедури. Дані, що відображаються на інформаційній панелі, відповідали даним, що зберігаються в серверній частині, а інтерфейс панелі був зручним та інтуїтивно зрозумілим. Загалом, модуль "Інформаційна панель медичної сестри" пройшов усі тестові випробування, продемонструвавши свою ефективність у допомозі медичним сестрам ефективно виконувати свої щоденні завдання.

Отже, модульне тестування різних модулів веб-орієнтованої програмної системи для медсестер будинку престарілих було всебічним та успішним. Кожен модуль пройшов ретельне тестування, щоб гарантувати, що всі функціональні можливості працюють як очікувалося. Завдяки серії ретельно розроблених тестових функцій можемо переконатися, що кожен модуль виконує свої функції точно та ефективно. Успішне завершення етапу модульного тестування свідчить про стійкість та надійність програмної системи.

ВИСНОВКИ

У результаті кваліфікаційної роботи бакалавра була розроблена веб-орієнтована програмна система для медсестер будинку літніх людей, використовуючи стек MERN.

Був проведений ретельний аналіз предметної галузі, фокусуючись на наявних аналогах та визначаючи ключові проблеми, які необхідно вирішити. Цей аналіз створив основу для розуміння контексту, в якому буде працювати програмна система. На основі виявлених проблем була сформована постановка задачі, що закладає основу для процесу розробки.

Були сформовані вимоги до програмної системи, що гарантує ефективне врахування потреб та очікувань усіх зацікавлених сторін. Цей крок мав вирішальне значення для спрямування процесу розробки та забезпечення відповідності кінцевого продукту потребам користувачів.

Були розроблені архітектура та дизайн програмного забезпечення, використовуючи UML для візуалізації та специфікації структури та поведінки програмної системи. Проектування архітектури програмного забезпечення передбачало прийняття важливих рішень щодо компонентів, взаємодії та загальної організації системи для забезпечення масштабованості, гнучкості та ефективності.

Крім того, розробка надійної структури зберігання даних мала важливе значення для ефективного управління та доступу до даних у системі.

Було наведено приклади цікавих алгоритмів і методів, які демонструють, як можна ефективно вирішувати складні проблеми в програмній системі. Ці алгоритми та методи відіграють важливу роль у реалізації різних функціональних можливостей та особливостей системи.

Створення користувацького інтерфейсу та дизайну користувацького досвіду мало важливе значення для забезпечення того, щоб програмна система була інтуїтивно зрозумілою, зручною для користувача та відповідала стандартам юзабіліті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Miles R., Hamilton K. Learning UML 2.0: A Pragmatic Introduction to UML, 1st Edition. – Sebastopol: O'Reilly Media, 2006. – 283 p.

2. Cheesman J., Daniels J. Uml Components: A Simple Process for Specifying Component-Based Software, 1st Edition. – Boston: Addison-Wesley Professional, 2000. – 176 p.

3. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition. – Print2print, 2016. – 208 p.

4. Seidl M., Scholz M., Huemer C., Kappel G. UML @ Classroom: An Introduction to Object-Oriented Modeling (Undergraduate Topics in Computer Science). – Berlin: Springer, 2015. – 218 p.

5. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd Edition. – London: Pearson, 2004. – 736 p.

6. Rumbaug J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual, 2nd Edition. – Boston: Addison-Wesley Professional, 2004. – 721 p.

7. Bush E. JavaScript Three-Tier Architectures in AWS with React, Node and MongoDB: Design, code, test, deploy, and manage in Amazon AWS. – Connecticut: Blue Sky Productions Inc., 2022. – 566 p.

8. Duckett J. HTML and CSS: Design and Build Websites. – Hoboken: John Wiley & Sons, 2011. – 490 p.

9 Rappin N. Modern Front-End Development for Rails: Hotwire, Stimulus, Turbo, and React, 2nd Edition. – Pragmatic Bookshelf, 2022. – 410 p.

10. Hunter II T. Distributed Systems with Node.js: Building Enterprise-Ready Backend Services, 1st Edition. – Sebastopol: O'Reilly Media, 2020. – 377 p.

11. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack, 2nd Edition. – Sebastopol: O'Reilly Media, 2019. – 343 p.

12. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage, 3rd Edition. – Sebastopol: O'Reilly Media, 2019. – 511 p.

13. Aleksendric M., Borucki A., Domingues L. Mastering MongoDB 7.0 - Fourth Edition: Achieve data excellence by unlocking the full potential of MongoDB, 4th ed. Edition. – Birmingham: Packt Publishing, 2024. – 434 p.

14. Saxena H., Agarwal P. Introduction to CPU Scheduling Algorithms: Design and Performance Evaluation of Optimum Service Time Concept for Round Robin Algorithm (OSTRR). – Saarbrucken: LAP LAMBERT Academic Publishing, 2012. – 52 p.

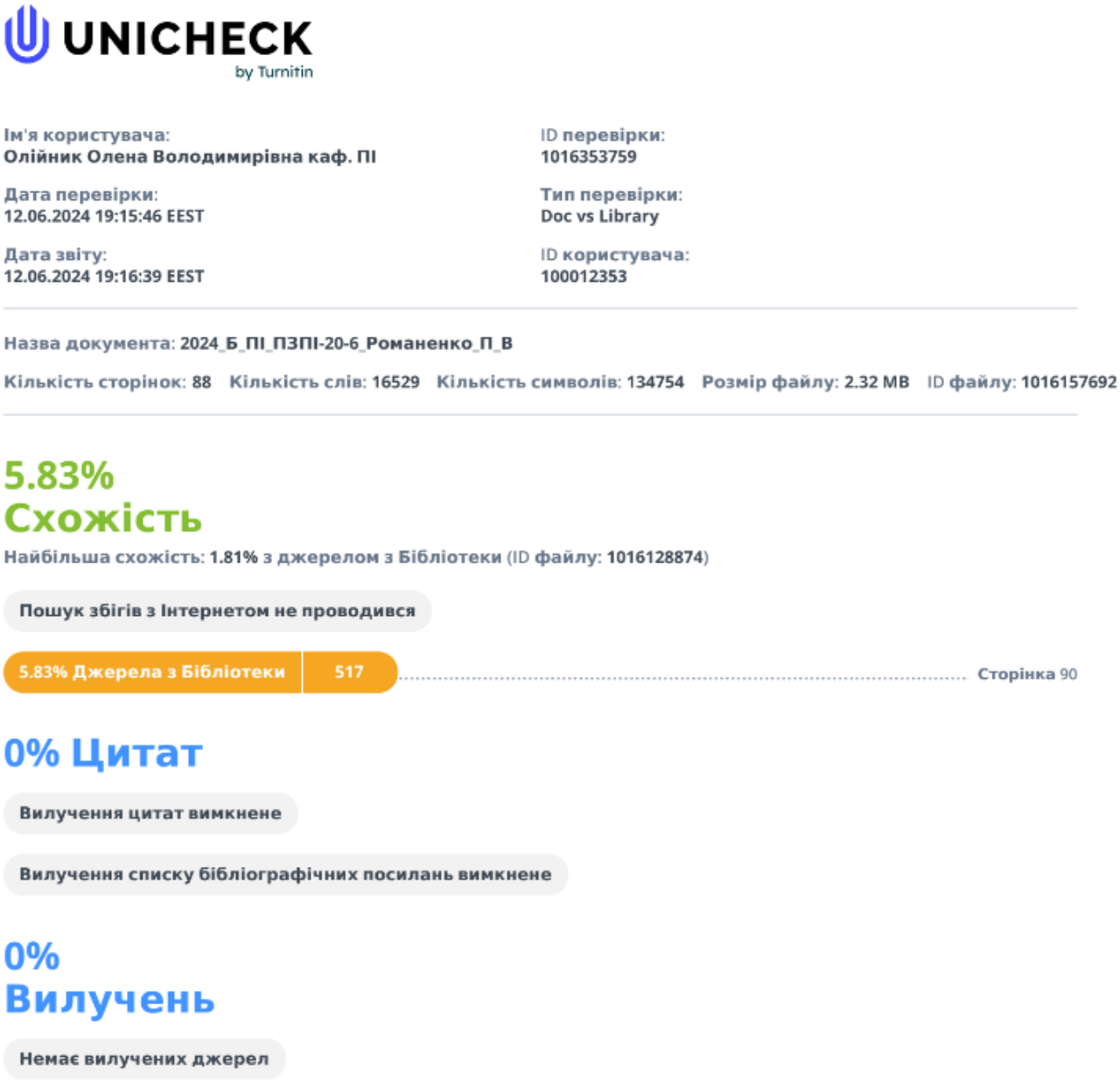
15. P. Bertsekas D. Dynamic Programming and Optimal Control, 4th Edition. – Nashua: Athena Scientific, 2012. – 1270 p.

16. Mitchell M. An Introduction to Genetic Algorithms (Complex Adaptive Systems). – Cambridge: MIT Press, 1998. – 221 p.

17. Park U. Introduction to Design Thinking for UX Beginners: 5 Steps to Creating a Digital Experience That Engages Users with UX Design, UI Design, and User Research. Start Building Your UX Career. – Independently published, 2023. – 166 p.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016353759

Дата перевірки:
12.06.2024 19:15:46 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
12.06.2024 19:16:39 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ-20-6_Романенко_П_В

Кількість сторінок: 88 Кількість слів: 16529 Кількість символів: 134754 Розмір файлу: 2.32 MB ID файлу: 1016157692

5.83%
Схожість

Найбільша схожість: 1.81% з джерелом з Бібліотеки (ID файлу: 1016128874)

Пошук збігів з Інтернетом не проводився

5.83% Джерела з Бібліотеки 517 Сторінка 90

0% Цитат

Вилучення цитат вимкнене

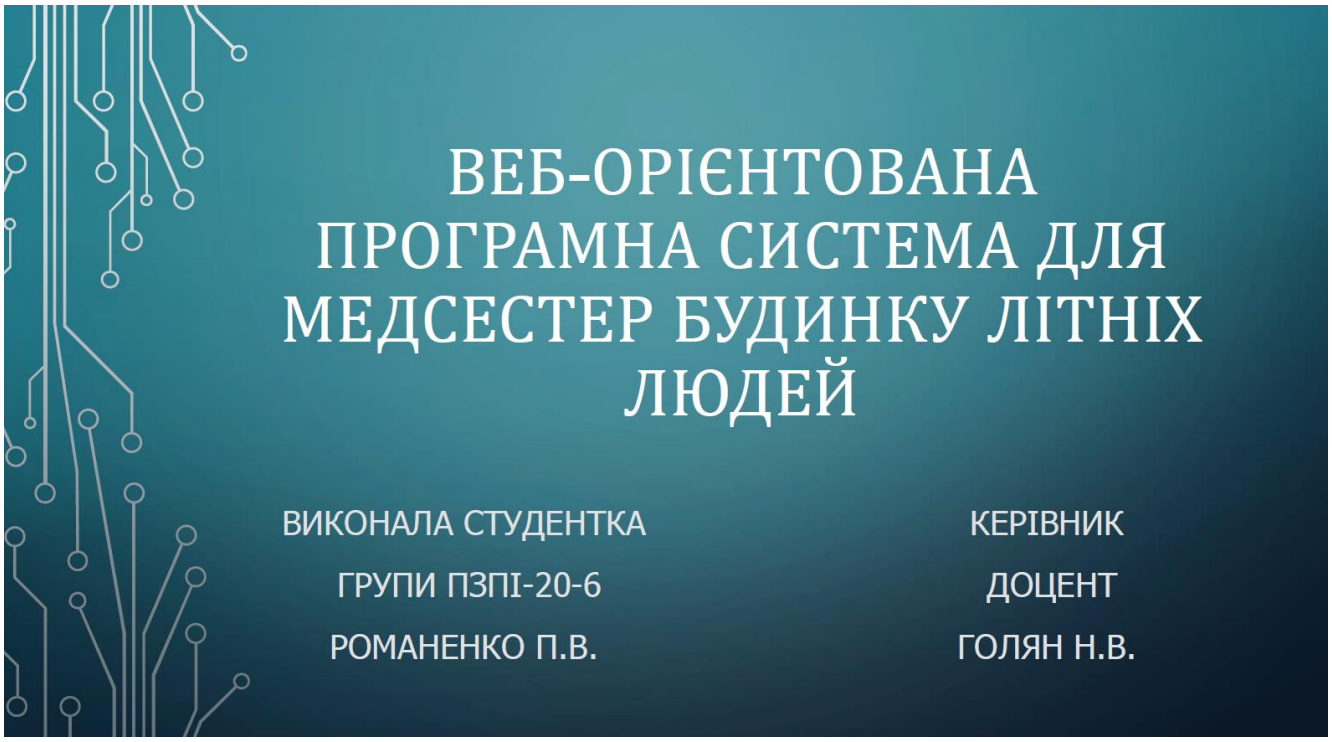
Вилучення списку бібліографічних посилань вимкнене

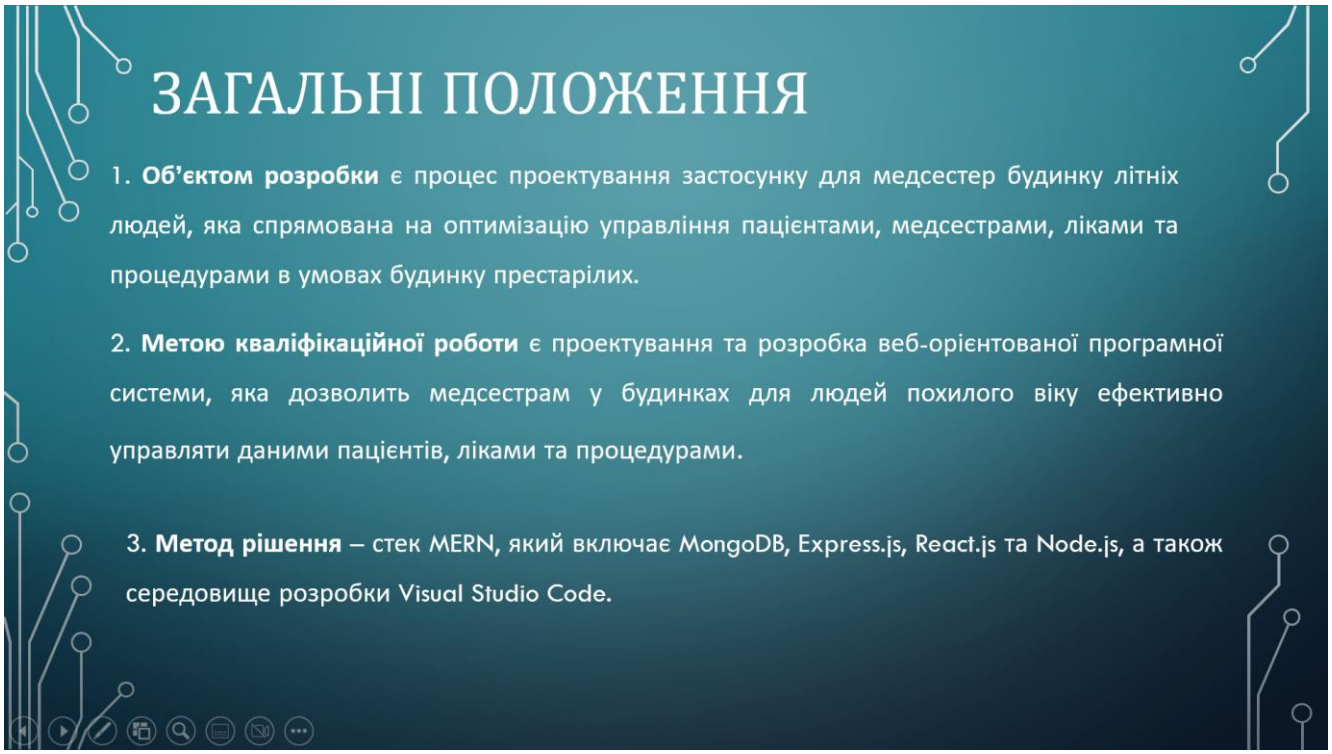
0%
Вилучень

Немає вилучених джерел

ДОДАТОК Б

Слайди презентації





ПОСТАНОВКА ЗАДАЧІ

- аналіз сфери застосування;
- оцінка існуючих аналогів;
- проектування структури веб-додатку;
- визначення мови програмування та технології, які будуть використані для розробки додатку;
- розробка зручного та привабливого інтерфейсу для користувачів;
- створення бази даних і її інтеграція в проект;
- процес тестування додатку перед впровадженням.

Це загальна постановка задачі для створення застосунку для медсестер будинку літніх людей. Конкретні деталі і вимоги можуть змінюватися в залежності від бізнес–потреб та бюджету проекту. Після постановки задачі важливо розробити технічне завдання та вибрати технології, які найкраще підходять для реалізації цих вимог.

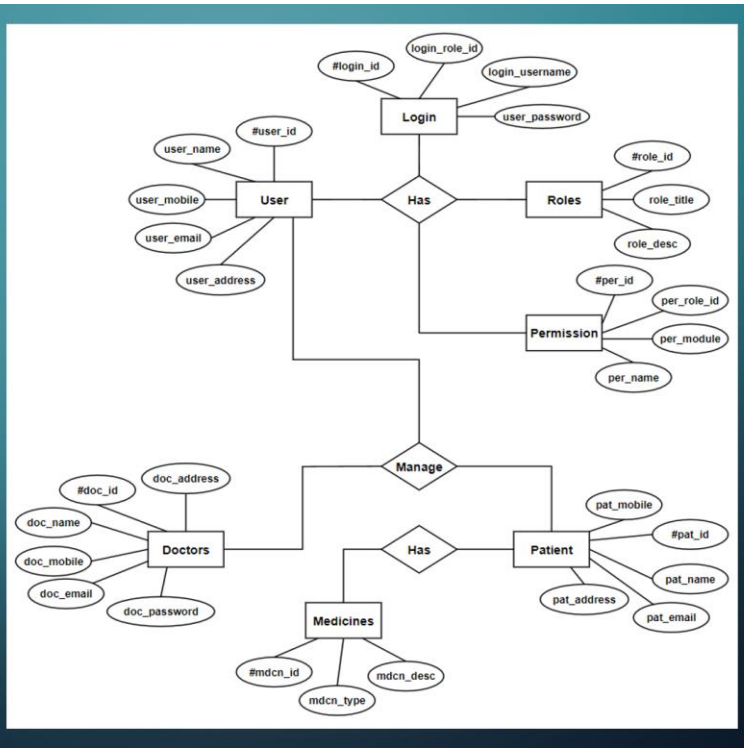
ФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ

1. Управління користувачами
2. Управління пацієнтами
3. Управління медикаментами та процедурами
4. Комунікація та співпраця
5. Адміністративні функції

НЕФУНКЦІОНАЛЬНІ ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ

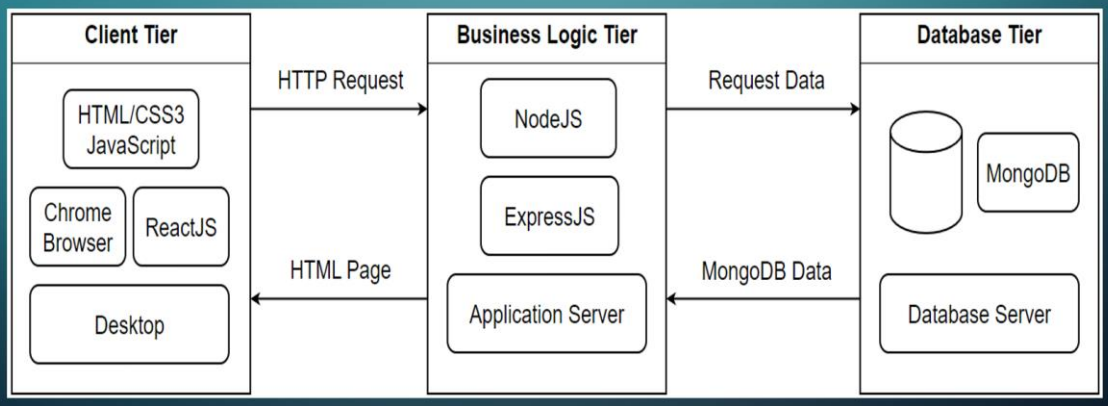
1. Безпека
2. Зручність використання
3. Надійність
4. Продуктивність
5. Маштабованість

ER - ДІАГРАМА



АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

При розробці цієї системи використовувався стек MERN, який складається з MongoDB, Express.js, React.js та Node.js



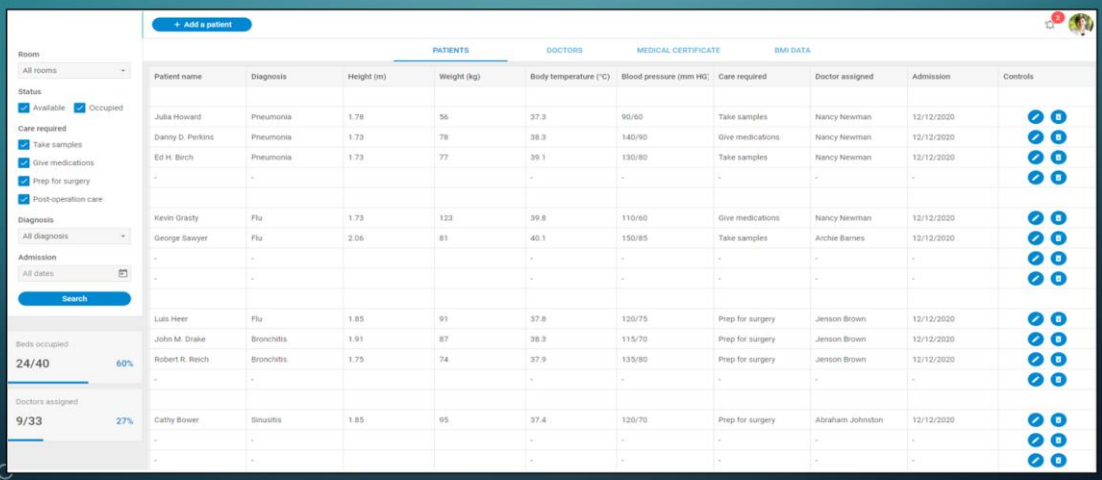
АЛГОРИТМ РОЗПОДІЛУ ПАЦІЄНТІВ

Алгоритм планування Round Robin (RR) є поширеним методом розподілу завдань, коли завдання призначаються ресурсам по колу, гарантуючи, що кожен ресурс отримує рівну частку завдань протягом певного часу.

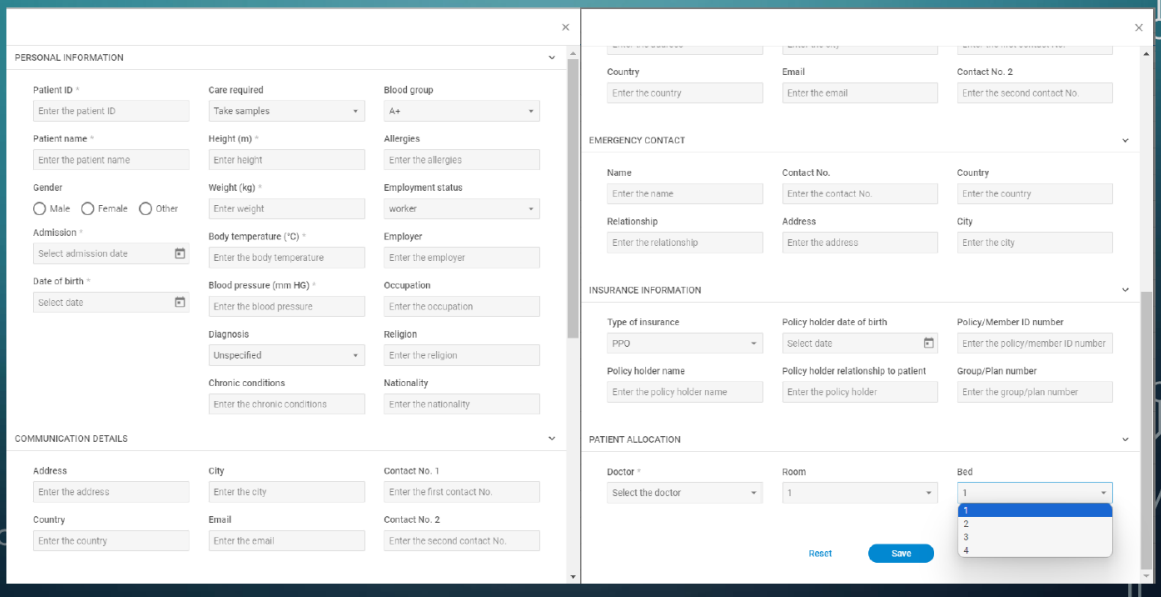
```
function allocateNurseToPatient(nurses, patients) {
    let nurseIndex = 0;

    patients.forEach((patient) => {
        if (nurseIndex >= nurses.length) {
            nurseIndex = 0; // Повернутися до першої медсестри, коли всі медсестри пройдені
        }

        const nurse = nurses[nurseIndex];
        nurse.assignPatient(patient); // Призначити пацієнта до медсестри
        nurseIndex++; // Перейти до наступної медсестри для наступного пацієнта
    });
}
```

UI/UX ДИЗАЙН СИСТЕМИ

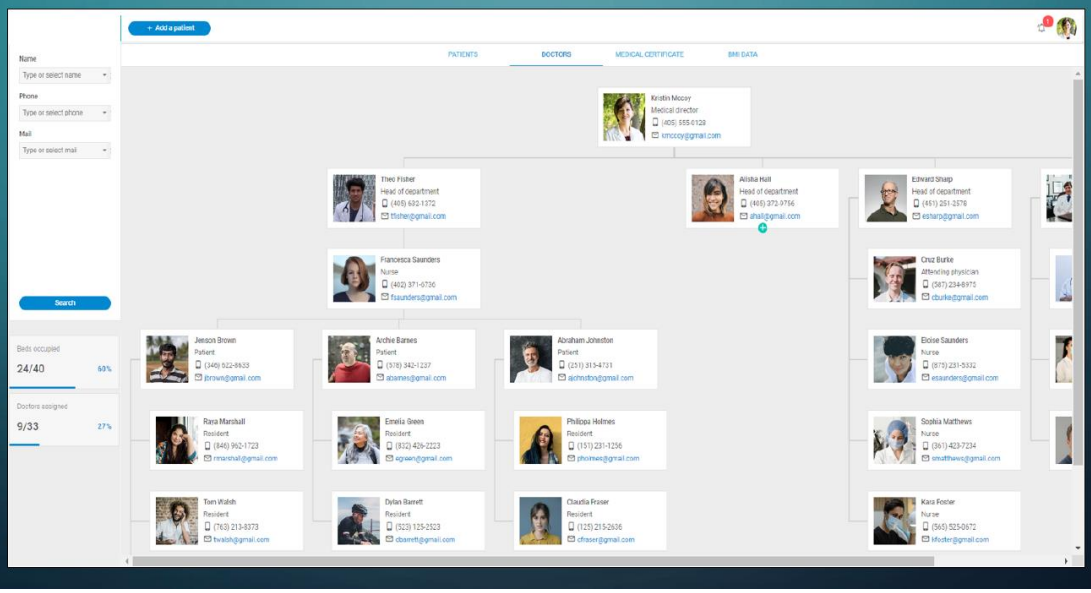
СТОРІНКА «ПАЦІЄНТИ»



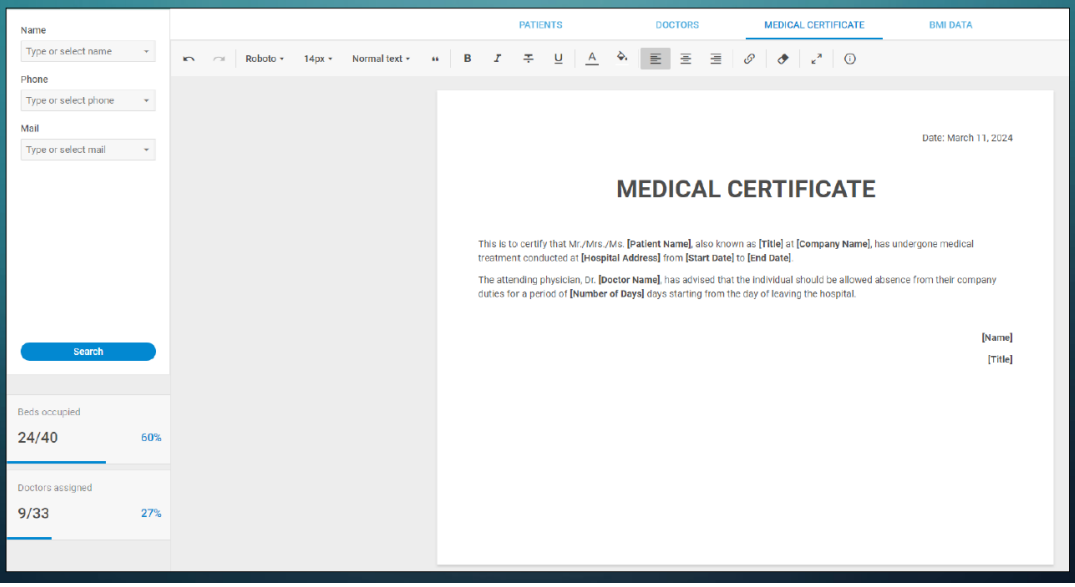
ДОДАВАННЯ КОРИСТУВАЧА



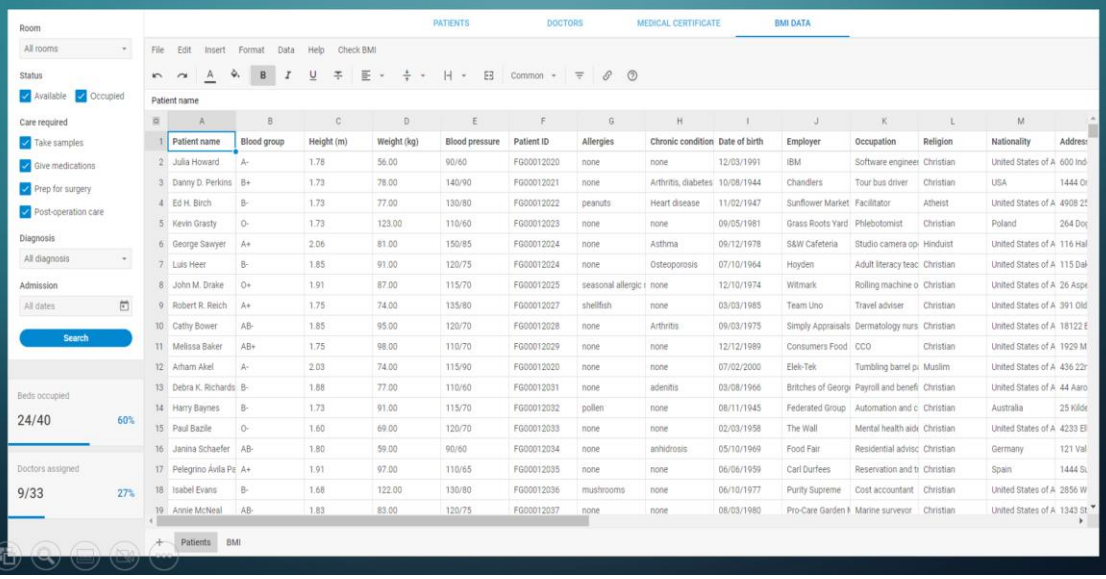
СТОРІНКА «ЛІКАРІ»



СТОРІНКА «МЕДИЧНА ДОВІДКА»



СТОРІНКА ДАНИХ ІМТ



ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є розробка веб-орієнтованої програмної системи для медсестер будинку літніх людей, використовуючи стек MERN.

Був проведений ретельний аналіз предметної галузі, фокусуючись на наявних аналогах та визначаючи ключові проблеми, які необхідно вирішити.

Були сформовані вимоги до програмної системи, що гарантує ефективне врахування потреб та очікувань усіх зацікавлених сторін.

Були розроблені архітектура та дизайн програмного забезпечення, використовуючи UML для візуалізації та специфікації структури та поведінки програмної системи.


ДЯКУЮ ЗА УВАГУ!