Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмна розробка автоматизованої системи обліку у ветеринарній клініці»

Студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки» Сендеркін Микита Володимирович

_____
підпис студента

Науковий керівник
Кандидат технічних наук, доцент Козлов Валерій Володимирович

_____
підпис керівника

Гарант освітньої програми
кандидат технічних наук, доцент Демідов Павло Георгійович

_____
підпис керівника

Київ 2023

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О. І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студенту

Сендеркіну Микиті Володимировичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи
«Програмна розробка автоматизованої системи обліку у ветеринарній клініці»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка додатку для автоматизації обліку у ветеринарній клініці.
Об'єкт дослідження: процеси обліку у ветеринарній клініці.
Предмет дослідження: інформаційні технології обліку у ветеринарній клініці.
4. Перелік графічного матеріалу *рисунки*

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Козлов В. В.	15.12.2022 р.	15.12.2022 р.
2	Козлов В. В.	15.12.2022 р.	15.12.2022 р.
3	Козлов В. В.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП
РОЗДІЛ 1. АНАЛІТИЧНЕ ДОСЛІДЖЕННЯ СПЕЦИФІКИ ДІЯЛЬНОСТІ ВЕТЕРИНАРНИХ КЛІНІК В УКРАЇНІ
1.1. Організація діяльності ветеринарних клінік в Україні
1.2. Огляд існуючих інформаційних систем автоматизації діяльності ветеринарної клініки
1.3. Формування вимог до автоматизованої системи обліку у ветеринарній клініці
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОБЛІКУ У ВЕТЕРИНАРНІЙ КЛІНІЦІ
2.1. Концепція розробки системи обліку
2.2. Проектування додатку для обліку у ветеринарній клініці
2.3. Обґрунтування засобів розробки інформаційної системи з обліку у ветеринарній клініці
РОЗДІЛ 3. РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ У ВЕТЕРИНАРНІЙ КЛІНІЦІ
3.1. Розробка інформаційного забезпечення системи обліку у ветеринарній клініці
3.2. Розробка алгоритму роботи інформаційної системи
3.3. Тестування роботи системи
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*01.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	
4	*Розділ 1. Аналітичне дослідження специфіки діяльності ветеринарних клінік в Україні*	*28.02.2023*	*28.02.2023*
5	*Розділ 2. Проектування системи обліку у ветеринарній клініці*	*06.04.2023*	*06.04.2023*
6	*Розділ 3. Розробка автоматизованої системи обліку у ветеринарній клініці*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 – 01.06.2023*	*31.05.2023 – 01.06.2023*
10	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	*За розкладом роботи ЕК*

8. Дата видачі завдання «15» грудня 2022 р.

9. Керівник випускної кваліфікаційної роботи
Козлов В. В.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П. Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Сендеркін М. В.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Керівник випускної кваліфікаційної роботи
________________ 30.05.2023 р.
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу (проект)

Випускна кваліфікаційна робота студента ___________________________
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми___________________________ Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _________________________________ Пурський О.І.
(підпис, прізвище, ініціали)
«_____»__________________2023 р.

ЗМІСТ

ВСТУП 9

РОЗДІЛ 1. АНАЛІТИЧНЕ ДОСЛІДЖЕННЯ СПЕЦИФІКИ ДІЯЛЬНОСТІ ВЕТЕРИНАРНИХ КЛІНІК В УКРАЇНІ 10

1.1. Організація діяльності ветеринарної клініки в Україні 10

1.2. Огляд існуючих інформаційних систем автоматизації діяльності ветеринарної клініки 11

1.3. Формування вимог до автоматизованої системи обліку у ветеринарній клініці 12

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОБЛІКУ У ВЕТЕРИНАРНІЙ КЛІНІЦІ 14

2.1. Концепція розробки системи обліку 14

2.2. Проектування додатку для обліку у ветеринарній клініці 16

2.3. Обґрунтування засобів розробки інформаційної системи з обліку у ветеринарній клініці 19

РОЗДІЛ 3. РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ У ВЕТЕРИНАРНІЙ КЛІНІЦІ 24

3.1. Розробка інформаційного забезпечення системи обліку у ветеринарній клініці 24

3.2. Розробка алгоритму роботи інформаційної системи 26

3.3. Тестування роботи системи 36

ВИСНОВКИ 39

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 40

ДОДАТОК А

Анотація

Випускна кваліфікаційна робота присвячена розробці автоматизованої системи обліку у ветеринарній клініці з метою покращення ефективності та якості надання послуг. В першому розділі проведено аналітичне дослідження особливостей діяльності ветеринарних клінік в Україні, проведено огляд існуючих інформаційних систем автоматизації діяльності ветеринарних клінік та формування вимог до розроблюваної системи. У другому розділі запропоновано концепцію розробки системи, обґрунтовано засоби розробки інформаційної системи обліку у ветеринарній клініці. У третьому розділі запропоновано алгоритм роботи інформаційної системи, та реалізовано інформаційне забезпечення запропонованої системи обліку. Результати цієї роботи можуть бути використані для покращення роботи ветеринарних клінік та забезпечення якісних послуг для тварин та їх власників.

Ключові слова: автоматизована система обліку, ветеринарна клініка, інформаційне забезпечення.

Abstract

The graduation thesis is dedicated to the development of an automated accounting system for a veterinary clinic with the aim of improving efficiency and service quality. The first chapter presents an analytical study of the peculiarities of veterinary clinics in Ukraine, provides an overview of existing information systems for automating veterinary clinic activities, and formulates requirements for the developed system. The second chapter proposes the concept of system development, justifies the tools for developing an information accounting system in a veterinary clinic. The third chapter presents the algorithm of the information system's operation and implements the information support for the proposed accounting system. The results of this work can be used to improve the functioning of veterinary clinics and ensure quality services for animals and their owners.

Keywords: automated accounting system, veterinary clinic, information support.

ВСТУП

У сучасному світі інформаційні технології відіграють важливу роль у різних сферах людського життя, включаючи сферу ветеринарної медицини. Ветеринарні клініки потребують ефективного та надійного інформаційного забезпечення для поліпшення організації роботи, оптимізації процесів обліку та покращення якості надання ветеринарних послуг.

Актуальність дослідження полягає в тому, що ветеринарні клініки потребують ефективного інформаційного забезпечення для поліпшення організації роботи та якості надання послуг. Розробка автоматизованої системи обліку ветеринарної клініки сприятиме покращенню ефективності роботи, точності обліку та задоволеності клієнтів.

Метою випускної кваліфікаційної роботи є розробка додатку для автоматизації обліку у ветеринарній клініці.

Об'єктом дослідження є процеси обліку у ветеринарній клініці.

Предметом дослідження є інформаційні технології обліку у ветеринарній клініці.

Для досягнення поставленої мети необхідно виконати наступний перелік завдань:

1. Аналізувати специфіку діяльності ветеринарних клінік в Україні та виявити основні проблеми, пов'язані з обліком та організацією роботи.
2. Провести огляд існуючих інформаційних систем автоматизації діяльності ветеринарних клінік і провести аналіз їхніх переваг та недоліків.
3. Сформулювати вимоги до автоматизованої системи обліку у ветеринарній клініці.
4. Розробити концепцію розробки системи обліку та проектувати додаток для обліку у ветеринарній клініці.
5. Обґрунтувати вибір засобів розробки інформаційної системи з обліку у ветеринарній клініці.

6. Розробити інформаційне забезпечення системи обліку у ветеринарній клініці, включаючи базу даних, функціональні можливості та інтерфейс користувача.

7. Розробити алгоритм роботи інформаційної системи, забезпечуючи правильну послідовність дій та функціональну взаємодію.

8. Провести тестування роботи системи для перевірки її функціональності, надійності та відповідності вимогам.

Результатами проведеного дослідження є розробка та реалізація автоматизованої системи обліку у ветеринарній клініці, що покращить організацію роботи, забезпечить точний облік та підвищить якість надання ветеринарних послуг.

РОЗДІЛ 1. АНАЛІТИЧНЕ ДОСЛІДЖЕННЯ СПЕЦИФІКИ ДІЯЛЬНОСТІ ВЕТЕРИНАРНИХ КЛІНІК В УКРАЇНІ

1.1. Організація діяльності ветеринарної клініки в Україні

У цьому підрозділі буде проведено детальний аналіз організації діяльності ветеринарних клінік в Україні. Для досягнення мети дослідження - розробки автоматизованої системи обліку у ветеринарній клініці, важливо розуміти специфіку функціонування ветеринарних клінік.

Законодавча база та регулювання

Першочергово, буде розглянута законодавча база та регулювання, яке стосується діяльності ветеринарних клінік в Україні. Зокрема, будуть розглянуті відповідні закони, нормативно-правові акти, положення та рекомендації, що визначають правила та вимоги до функціонування ветеринарних закладів.

Організаційна структура та персонал

Далі, розглянемо організаційну структуру та персонал ветеринарної клініки. Цей аспект включає аналіз структури керівництва, функціональних підрозділів та ролі кожного з них у процесі надання ветеринарних послуг. Також будуть розглянуті вимоги до кваліфікації та компетенцій персоналу, його розподілу за посадами та обов'язками.

Обладнання та інфраструктура

Важливим аспектом діяльності ветеринарних клінік є обладнання та інфраструктура, необхідні для надання якісних ветеринарних послуг. В рамках цього розділу будуть розглянуті основні види обладнання, що використовуються в клініках, а також вимоги до інфраструктури, зокрема приміщень, лікарських кабінетів, лабораторій та інших робочих зон.

Фінансування та фінансовий облік

Аналізуючи специфіку діяльності ветеринарних клінік, важливо розглянути питання фінансування та фінансового обліку. Будуть досліджені джерела фінансування, витрати, ціноутворення, система оплати послуг та основні

принципи фінансового обліку, які є важливими елементами успішної діяльності ветеринарної клініки.

Маркетинг та взаємодія з клієнтами

Останнім, але не менш важливим аспектом є маркетингова стратегія та взаємодія з клієнтами ветеринарної клініки. Будуть розглянуті методи просування послуг, залучення та утримання клієнтів, а також важливі аспекти комунікації та клієнтського сервісу.

1.2. Огляд існуючих інформаційних систем автоматизації діяльності ветеринарної клініки

У цьому підрозділі наведено огляд існуючих інформаційних систем, які використовуються для автоматизації діяльності ветеринарних клінік. Аналізуючи ці системи, можна сформувати висновки про рівень розвитку технологій в даній галузі, їх можливості та обмеження [1-4].

Система управління медичними записами (Electronic Medical Record System)

Системи управління медичними записами (EMR) є одними з ключових компонентів інформаційних систем ветеринарних клінік. Вони призначені для електронного зберігання, обробки та аналізу медичної інформації про пацієнтів. EMR дозволяє ветеринарним фахівцям швидко отримувати доступ до медичних записів пацієнтів, включаючи медичну історію, результати лабораторних досліджень, діагностику, рецепти тощо.

Система управління фінансами та обліком (Financial and Accounting Management System)

Для ефективного функціонування ветеринарних клінік необхідна система управління фінансами та обліком, яка допомагає вести облік фінансових операцій, витрат, оплати послуг та інших фінансових процесів. Ця система дозволяє відстежувати фінансову діяльність клініки, формувати звіти, контролювати витрати та прибуток.

Система планування та управління ресурсами (Resource Planning and Management System)

Системи планування та управління ресурсами ветеринарних клінік допомагають оптимізувати розподіл ресурсів, таких як лікарський персонал, обладнання, ліки та інші матеріали. Вони забезпечують ефективне планування робочого графіку лікарів, координацію замовлення та доставки необхідних ресурсів, а також допомагають уникнути перевантажень або недостатньої кількості ресурсів.

Система управління клієнтськими відносинами (Customer Relationship Management System)

Системи управління клієнтськими відносинами (CRM) дозволяють ветеринарним клінікам покращити комунікацію та взаємодію з клієнтами. Вони забезпечують збір та аналіз даних про клієнтів, допомагають управляти розсилками, реєстрацією та плануванням візитів, а також ведуть статистику та аналізують задоволеність клієнтів.

1.3. Формування вимог до автоматизованої системи обліку у ветеринарній клініці

Враховуючи специфіку діяльності клініки та аналізуючи потреби фахівців, пацієнтів та адміністративного персоналу, було сформовано набір вимог до автоматизованої системи обліку [5].

Функціональні вимоги

Функціональні вимоги визначають, які функції та можливості має мати автоматизована система обліку. Нижче наведено основні функціональні вимоги, які можуть бути важливими для ветеринарної клініки:

1. Реєстрація пацієнтів: система повинна дозволяти реєструвати пацієнтів, зберігати їх особисті дані, медичну історію та інші важливі деталі.
2. Термінологія та класифікація: система повинна містити стандартну термінологію та класифікацію для ветеринарної медицини, що сприятиме однозначному розумінню та обміну інформацією.
3. Управління прийомами: система повинна дозволяти планувати та керувати прийомами пацієнтів, включаючи запис, перенесення та скасування.

4. Лікарські препарати та рецепти: система повинна забезпечувати облік лікарських препаратів, виписування та контроль рецептів.

5. Фінансовий облік: система повинна здійснювати облік фінансових операцій, включаючи оплату послуг, стягнення платежів та формування звітів.

6. Зв'язок з лабораторіями: система повинна дозволяти передавати та отримувати результати лабораторних досліджень, забезпечуючи швидкий та безпечний обмін інформацією.

Нефункціональні вимоги

Нефункціональні вимоги визначають якість, надійність та інші характеристики, які має мати автоматизована система обліку. Деякі з нефункціональних вимог, які можуть бути важливими для ветеринарної клініки, включають:

1. Безпека даних: система повинна забезпечувати високий рівень захисту конфіденційної інформації пацієнтів та інших даних.

2. Швидкодія: система повинна працювати швидко та ефективно, забезпечуючи швидкий доступ до інформації та обробку даних.

3. Легкість використання: система повинна бути інтуїтивно зрозумілою та простою у використанні для різних категорій користувачів.

4. Масштабованість: система повинна мати можливість розширюватись та адаптуватись до зростаючих потреб клініки.

5. Сумісність: система повинна бути сумісною з існуючими інформаційними системами та технологічними стандартами.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ОБЛІКУ У ВЕТЕРИНАРНІЙ КЛІНІЦІ

2.1. Концепція розробки системи обліку

Концепція є першим кроком у проектуванні системи обліку і включає в себе визначення загальних принципів, цілей та функціональних вимог системи [3,4]. Основною метою є розробка концепції, яка відповідатиме потребам ветеринарної клініки та забезпечуватиме ефективну автоматизацію процесів обліку.

✓ ***Визначення цілей системи обліку***

Перш за все необхідно визначити цілі системи обліку у ветеринарній клініці. Основною метою є поліпшення ефективності та точності обліку, спрощення робочих процесів та забезпечення зручності для персоналу клініки. Нижче наведені деякі основні цілі системи обліку:

- автоматизація процесів: система повинна автоматизувати процеси обліку, що дозволить зменшити ручну роботу, уникнути помилок та зберегти час;
- збереження інформації: система повинна забезпечити надійне збереження всієї необхідної інформації про пацієнтів, медичні записи, фінансові дані та іншу важливу інформацію;
- покращення комунікації: система повинна сприяти покращенню комунікації між лікарями, медичним персоналом та адміністративними працівниками клініки;
- забезпечення аналітичних звітів: система повинна здати формувати аналітичні звіти, які допоможуть в управлінні клінікою, прийнятті рішень та плануванні діяльності;
- забезпечення безпеки даних: система повинна мати високий рівень захисту даних пацієнтів, забезпечуючи конфіденційність та відсутність несанкціонованого доступу.

✓ ***Функціональні вимоги системи обліку***

Нижче наведені основні функціональні вимоги системи обліку у ветеринарній клініці:

● реєстрація пацієнтів: система повинна дозволяти реєструвати нових пацієнтів та зберігати їхні особисті дані, медичні картки, історію хвороби тощо;

● запис медичних відвідувань: система повинна дозволяти вести записи про візити пацієнтів, включаючи симптоми, діагноз, призначення лікування, проведені процедури та рецепти;

● управління лікарськими препаратами: система повинна забезпечувати інформацію про наявність лікарських препаратів, їхнє замовлення, прийом, розподіл, та контроль за терміном придатності;

● фінансовий облік: система повинна забезпечувати облік фінансових операцій, включаючи виписку рахунків, оплату послуг, стягнення платежів та формування звітів;

● лабораторні дослідження: система повинна мати можливість передавати та отримувати результати лабораторних досліджень, забезпечуючи швидкий та безпечний обмін інформацією.

✓ ***Архітектура системи обліку***

Одним із ключових етапів в проектуванні системи обліку є проектування архітектури системи [4]. Архітектура визначає структуру, компоненти та взаємозв'язки системи. Для системи обліку у ветеринарній клініці може бути запропонована трирівнева архітектура, що включає:

● клієнтський рівень: включає інтерфейс користувача, який дозволяє лікарям, медичному персоналу та адміністративним працівникам взаємодіяти з системою;

● серверний рівень: включає базу даних, де зберігається інформація про пацієнтів, медичні записи, фінансові дані та інша важлива інформація. Також включає логіку системи та обробку запитів;

- мережевий рівень: забезпечує зв'язок між клієнтським та серверним рівнями через мережу, що дозволяє передавати дані та забезпечує безпеку мережевого обміну;

✓ ***Прийняття технічних рішень***

На цьому етапі важливо визначити технічні рішення, які допоможуть втілити концепцію системи обліку. До вирішення належать питання вибору програмного забезпечення, бази даних, мов програмування та апаратного забезпечення. Вибір технологій повинен бути обґрунтованим і враховувати потреби ветеринарної клініки, можливості розробників та вимоги до безпеки та стабільності системи.

2.2. Проектування додатку для обліку у ветеринарній клініці

Архітектура визначає структуру та взаємозв'язки компонентів додатку [3,4]. На вибір може бути запропонована трирівнева архітектура, що включатиме наступні компоненти:

- клієнтський інтерфейс: це частина додатку, яка взаємодіє з користувачами. Клієнтський інтерфейс повинен бути зручним у використанні та забезпечувати доступ до основних функцій системи, таких як реєстрація пацієнтів, запис візитів, управління лікарськими препаратами тощо. Використання модерних технологій розробки інтерфейсу, таких як веб-додатки або мобільні додатки, може зробити його більш зручним для користувачів;
- бізнес-логіка: це компонент додатку, що містить основну логіку системи обліку. Він відповідає за обробку запитів користувачів, виконання розрахунків, зберігання та оновлення даних в базі даних. Бізнес-логіка повинна бути гнучкою і розширюваною, щоб легко впроваджувати нові функції та зміни у системі;
- база даних: це компонент, що забезпечує зберігання та доступ до даних системи. База даних має містити необхідну інформацію про пацієнтів, медичні записи, фінансові дані тощо. Вибір відповідної бази даних, такої

як реляційна база даних або NoSQL база даних, повинен враховувати потреби системи, обсяг даних та вимоги до швидкодії та безпеки.

Функціональні можливості системи. Це можуть бути такі функції, як:

1. Реєстрація пацієнтів та ведення їхньої медичної історії.
2. Запис візитів до ветеринарного лікаря та планування графіку роботи.
3. Управління медичними препаратами та інвентарем.
4. Фінансовий облік та ведення рахунків.
5. Звітність та аналітика даних про роботу клініки.

Кожна функціональна можливість повинна бути описана детально, включаючи вхідні та вихідні дані, процеси обробки інформації та очікувані результати.

Інтерфейс користувача. Важливим аспектом проектування додатку є розробка зручного та ефективного інтерфейсу користувача. Інтерфейс повинен бути інтуїтивно зрозумілим, забезпечувати зручну навігацію та ефективну взаємодію користувача з системою.

Забезпечення безпеки. Питання безпеки є важливим аспектом при проектуванні додатку для обліку у ветеринарній клініці, в тому числі для використання механізмів аутентифікації та авторизації користувачів, шифрування даних, забезпечення контролю доступу до інформації та інших заходів безпеки для захисту конфіденційності та цілісності даних.

Технології та інструменти розробки

- програмування: вибір мови програмування та фреймворки, які можуть бути використані для розробки додатку, з урахуванням обмеження кожного варіанту;
- база даних: визначення типу бази даних, який буде використовуватись для зберігання даних (реляційні та NoSQL бази даних, їх переваги та недоліки);
- інструменти розробки: вибір інтегрованого середовища розробки (IDE), системи керування версіями, засобів для автоматичного тестування та інших інструментів, які будуть використовуватись під час розробки додатку.

Вимоги до системи. Серед вимог до системи варто відзначити наступні:

- додаток повинен забезпечувати можливість реєстрації нових пацієнтів та ведення їхньої медичної історії;

- додаток повинен бути доступним для використання на різних платформах, таких як комп'ютери, планшети та мобільні пристрої;

- додаток повинен забезпечувати безпеку та конфіденційність даних пацієнтів шляхом застосування механізмів шифрування та контролю доступу.

Архітектура системи

Вона визначає структуру, компоненти та взаємодію системи:

- Компоненти системи: ідентифікація основних компонентів додатку та їх функціональність. Наприклад, це можуть бути компоненти для реєстрації пацієнтів, ведення медичної історії, управління медичними препаратами тощо.

- Взаємодія між компонентами: опис способів взаємодії між компонентами систем и. Це може бути через API (Application Programming Interface), базу даних, повідомлення та інші способи обміну даними та функціональністю.

- Архітектурні шаблони: можливість використання архітектурних шаблонів, таких як клієнт-сервер, MVC (Model-View-Controller) тощо.

- Масштабованість: варіанти горизонтального та вертикального масштабування, резервне копіювання даних та інші підходи до забезпечення масштабованості.

Тестування та якість. Опис плану тестування та забезпечення якості додатку для обліку у ветеринарній клініці (вказати види тестів, які будуть проведені (наприклад, модульні тести, інтеграційні тести, функціональні тести), інструменти, які будуть використані для автоматизованого тестування), а також критерії прийняття для успішного проходження кожного тесту. Також важливо враховувати процес забезпечення якості, який включатиме регулярну перевірку якості коду, виявлення та виправлення помилок, а також впровадження кращих практик розробки програмного забезпечення.

Оцінка ризиків. Одним із важливих етапів реалізації системи є проведення оцінки ризиків, які можуть виникнути під час розробки та впровадження

системи обліку у ветеринарній клініці. Для кожного ризику вкажіть його опис, ймовірність виникнення та вплив на проект.

Наприклад:

- *Ризик: Зміна вимог замовника під час розробки системи.*

Опис: Замовник може внести зміни до вимог, що призведе до затримок у розробці та зміни обсягу робіт.

Ймовірність: Середня.

Вплив: Високий.

Стратегія мінімізації ризику: Регулярні зустрічі з замовником для уточнення вимог та встановлення жорстких дедлайнів для змін вимог. Заходи для вирішення наслідків: Розглянути можливість використання гнучких методологій розробки, таких як Agile, для адаптації до змін вимог.

План впровадження. Такий план дозволить ефективно організувати процес переходу на нову систему обліку, забезпечити належне навчання персоналу та підтримку користувачів після впровадження.

2.3. Обґрунтування засобів розробки інформаційної системи з обліку у ветеринарній клініці

Важливою складовою реалізації автоматизованої системи обліку є обґрунтування вибіру засобів розробки інформаційної системи:

1. Мови програмування: обґрунтування вибору мов програмування, що найбільш відповідають потребам проекту (доступність програмістів, популярність мови, підтримка та екосистема)

2. Фреймворки та бібліотеки: обґрунтування вибір конкретних фреймворків та бібліотек, які сприятимуть розробці та забезпечать швидке впровадження системи (їх функціональні можливості, документацію, підтримку спільноти розробників та інші фактори).

3. Система управління базами даних: обґрунтування вибору системи управління базами даних (СУБД) для збереження інформації про пацієнтів,

медичну історію, ліки тощо (функціональність СУБД, швидкодію, масштабованість та інші критерії).

4. Інструменти розробки: обґрунтування вибору інструментів розробки, які допоможуть у створенні, налагодженні та тестуванні системи (інтегровані середовища розробки (IDE), засоби автоматизованого тестування, системи контролю версій та інші інструменти).

5. Хмарні технології: доцільність використання хмарних технологій для розміщення системи та забезпечення її доступності та масштабованості (вибір конкретних хмарних платформ або провайдерів).

6. Безпека та захист даних: вибір заходів щодо забезпечення безпеки та захисту даних в інформаційній системі (можливості шифрування, аутентифікації, авторизації та інших механізмів для забезпечення конфіденційності та цілісності даних)

Мови програмування

Необхідно обґрунтувати вибір конкретних мов програмування, що найбільш відповідають потребам проекту:

1. Популярність: Вибір мови програмування, яка має широку спільноту розробників та активну екосистему, може забезпечити доступ до різноманітних ресурсів, бібліотек, фреймворків та інших інструментів підтримки.

2. Досвід розробників: Важливо врахувати наявність програмістів з досвідом роботи в певних мовах програмування, оскільки це забезпечить більш ефективну розробку і підтримку проекту.

3. Відповідність потребам проекту: Розгляньте особливості проекту і вимоги до його функціональності, продуктивності та масштабованості. Вибір мови програмування, яка найкраще відповідає цим потребам, може сприяти швидкій та ефективній розробці.

Наприклад, для розробки додатку для обліку у ветеринарній клініці можуть бути використані такі мови програмування, як Python, Java або JavaScript. Python відомий своєю простотою, легкістю в освоєнні та великою кількістю

наявних бібліотек для обробки даних і розробки веб-додатків. Java має широке застосування в багатьох індустріях, міцну типізацію та високу продуктивність. JavaScript є незамінним для розробки веб-інтерфейсу та взаємодії з користувачем.

Фреймворки та бібліотеки

При обранні фреймворків доцільно розглядати наступні аспекти:

1. Функціональність: фреймворки повинні відповідати вимогам проекту і надавати необхідні інструменти для розробки різних модулів системи обліку.
2. Документація та підтримка: обрані фреймворки та бібліотеки повинні мати задокументовані API та наявну підтримку спільноти розробників. Це дозволить ефективніше використовувати їх можливості та отримувати допомогу у вирішенні проблем.
3. Швидкість розробки: важливо обрати такі фреймворки та бібліотеки, які дозволяють прискорити розробку за рахунок використання готових компонентів та рішень. Це забезпечить ефективнішу розробку і знизить затрати часу та зусиль.
4. Масштабованість: фреймворки та бібліотеки повинні дозволяти ефективно розширювати функціональність і підтримувати зростання обсягів даних та навантаженості.

Обґрунтування вибору конкретних фреймворків та бібліотек допоможе підтвердити їхню відповідність до потреб проекту та забезпечить ефективну розробку додатку для обліку у ветеринарній клініці.

База даних

Для зберігання даних інформаційної системи обліку ветеринарної клініки необхідно розглянути наступні аспекти:

1. реляційність: реляційні бази даних є широко використовуваними та мають добре вивірені структури даних;
2. продуктивність: ветеринарна клініка може мати значну кількість пацієнтів та медичних записів, які потрібно ефективно обробляти;

3. масштабованість: є необхідним критерієм у разі зростання обсягів даних, забезпечує роботу з великим обсягом даних, підтримує механізми реплікації та кластеризації;

4. зручність використання: можливість використовувати інтуїтивно зрозумілі SQL-запити, наявність графічного інтерфейсу для управління базою даних тощо.

Враховуючи ці аспекти, ви можете обґрунтувати вибір конкретної СУБД, яка найкраще відповідає потребам вашої інформаційної системи обліку ветеринарної клініки.

Розробка інтерфейсу користувача

1. Аналіз вимог користувача: визначення вимог, які стосуються інтерфейсу користувача, такі як зручність використання, ергономіка, доступність для різних категорій користувачів, визначення основних функцій та завдань, які користувач буде виконувати через інтерфейс;

2. Проектування інтерфейсу: розробка дизайну інтерфейсу, враховуючи принципи зручності використання та естетичного оформлення (кольорова палітра, шрифти, розташування елементів та інші аспекти, що допоможуть створити зручний інтерфейс);

3. Реалізація інтерфейсу: забезпечення інтерактивності, валідації даних та інші функціональні можливості, які необхідні для зручного використання системи;

4. Тестування інтерфейсу: виявлення помилок та недоліків для забезпечення якісного та надійного інтерфейсу користувача;

5. Візуалізація даних: можливості візуалізації даних в інтерфейсі користувача (використання діаграм, графіків, таблиць та інші візуальні елементи для зручного та інформативного представлення даних користувачу);

6. Доступність: реалізація рекомендацій та стандартів щодо доступності веб-сайтів та додатків;

7. Мобільність: можливість розробки мобільної версії інтерфейсу для зручного використання на мобільних пристроях (адаптивний дизайн, який забезпечить оптимальне відображення інтерфейсу на різних розмірах екранів);

8. Узгодженість з брендом: дотримання корпоративного стилю та елементів бренду ветеринарної клініки (візуальна узгодженість з іншими матеріалами та рекламними засобами клініки);

9. Забезпечення безпеки: врахування принципів безпеки та захисту даних (шифрування, автентифікація користувачів, контроль доступу тощо).

РОЗДІЛ 3. РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ ОБЛІКУ У ВЕТЕРИНАРНІЙ КЛІНІЦІ

3.1. Розробка інформаційного забезпечення системи обліку у ветеринарній клініці

При розробці інформаційного забезпечення системи обліку у ветеринарній клініці важливим етапом є вибір мов програмування та інструментів з урахуванням вимог до системи та потреб користувачів. Одним із ключових аспектів розробки інформаційного забезпечення є створення бази даних для зберігання медичних записів, клієнтської інформації, фінансових даних та інших даних, що використовуються ветеринарною клінікою. В процесі реалізації системи було враховано потреби у структуруванні та ефективному доступі до даних, а також забезпеченні їх безпеки та конфіденційності. Також розроблено модулі для реєстрації пацієнтів, запису медичних діагнозів та лікування, формування звітів та статистики. Була забезпечена можливість інтеграції з іншими системами, такими як лабораторні системи або системи управління запасами.

Загальною метою розробки інформаційного забезпечення було створення функціональної, ефективної та легко управляємої системи обліку у ветеринарній клініці. Для реалізації інтерфейсу були використані сучасні веб-технології та фреймворки, що дозволило забезпечити швидку та ефективну роботу системи. При розробці інтерфейсу також здійснювалося тестування та виправлення помилок для забезпечення його стабільності та надійності.

Розробка бази даних є ключовим етапом у створенні автоматизованої системи обліку у ветеринарній клініці. Метою цього етапу є створення структурованої та ефективної бази даних, яка забезпечує зберігання та обробку інформації про клініку, пацієнтів, власників тварин, медичних процедур та інших даних, необхідних для обліку ветеринарних послуг. При розробці бази даних були враховані потреби та вимоги ветеринарної клініки. При розробці бази даних використовувалися нормалізація даних, забезпечення

цілісності даних та оптимізація запитів. Було також враховано потребу у захисті даних та забезпеченні конфіденційності, використовуючи відповідні механізми аутентифікації та авторизації. Результатом розробки бази даних є структурована та ефективна база даних, яка забезпечує надійне зберігання та обробку інформації у ветеринарній клініці. Це дозволяє виконувати різні облікові операції швидко та ефективно, забезпечуючи точність та цілісність даних. База даних є важливою складовою системи обліку та допомагає підтримувати високу якість ветеринарних послуг у клініці.

Під час розробки функціоналу було враховано основні потреби та вимоги ветеринарної клініки, зокрема:

- ✓ реєстрація пацієнтів: система дозволяє реєструвати тварин, зберігати їх основні дані, такі як ім'я, порода, вік, стать тощо. Крім того, існує можливість додаткового введення інформації про медичну картку пацієнта, вакцинації та інші важливі дані;
- ✓ запис на прийом: клієнти можуть здійснювати запис на прийом через систему, обираючи зручну дату та час. Автоматичне попередження про наближення прийому дозволяє уникнути забуття та забезпечує своєчасний прихід клієнтів;
- ✓ медичні процедури: система дозволяє вносити та відстежувати проведення медичних процедур, таких як огляди, лікування, хірургічні втручання та інші. Для кожної процедури можна вносити відповідну інформацію, таку як діагноз, призначений лікувальний курс, препарати тощо;
- ✓ фінансовий облік: система забезпечує облік фінансових операцій, включаючи виписку рахунків, оплату послуг, виділення розрахункових даних та формування звітів про фінансовий стан клініки;
- ✓ звіти та статистика: система надає можливість формування різних звітів та статистики, що дозволяє аналізувати роботу ветеринарної клініки, відстежувати фінансові показники, кількість проведених процедур тощо.

У процесі розробки функціоналу реалізовано можливість масштабування системи та додавання нового функціоналу у майбутньому. В цілому, розробка функціоналу автоматизованої системи обліку у ветеринарній клініці відповідає поставленим вимогам та сприяє поліпшенню ефективності та якості обліку в клініці.

3.2. Розробка алгоритму роботи інформаційної системи

Розробка алгоритму роботи інформаційної системи є ключовим етапом у створенні автоматизованої системи обліку у ветеринарній клініці. Основні етапи розробки алгоритму роботи інформаційної системи включають:

1. Аналіз вхідних даних: під час цього етапу алгоритм роботи системи проводить аналіз та перевірку вхідних даних, таких як реєстраційна інформація пацієнтів, записи на прийом, результати медичних процедур тощо. Здійснюється перевірка достовірності та цілісності даних;
2. Обробка та зберігання даних: на цьому етапі алгоритм виконує операції з обробки та зберігання даних у відповідних базах даних. Це включає додавання, оновлення та видалення даних, створення резервних копій та забезпечення безпеки даних;
3. Виконання функціональних операцій: алгоритм роботи системи забезпечує виконання різноманітних функцій, таких як реєстрація пацієнтів, запис на прийом, проведення медичних процедур, облік фінансових операцій та інші. Кожна функція має свій власний алгоритм, який описує послідовність дій для його виконання;
4. Генерація звітів та статистики: алгоритм роботи системи також включає процес генерації звітів та статистики на основі накопичених даних. Це дозволяє користувачам отримувати різноманітну інформацію про роботу клініки, наприклад, звіти про прийоми, фінансову звітність, статистику за певний період тощо.

У процесі розробки алгоритму роботи інформаційної системи було враховано можливість масштабування системи та внесення змін у алгоритми

при необхідності. Відповідно до поставлених завдань та вимог, були розроблені алгоритми, які забезпечують ефективне функціонування системи. При розробці алгоритму були враховані основні процеси та функції, які відбуваються у ветеринарній клініці, такі як реєстрація пацієнтів, прийом, діагностика, лікування, складання медичних карток та інші. Алгоритми були розроблені з урахуванням послідовності дій, залежностей між ними та управління потоками даних у системі.

Результатом розробки алгоритму роботи інформаційної системи є функціональна, ефективна та надійна система, яка забезпечує автоматизований облік у ветеринарній клініці. Під час реалізації було забезпечено взаємодію між різними модулями системи, обробку та збереження даних, а також забезпечено безпеку і конфіденційність інформації. Були розроблені відповідні бази даних та налаштовані механізми забезпечення цілісності та доступу до даних. У процесі реалізації були проведені тестування та налагодження системи для виявлення та виправлення можливих помилок та недоліків. Результатом реалізації є готова до експлуатації інформаційна система обліку у ветеринарній клініці, яка відповідає вимогам, поставленим у початковому проекті. Система готова до впровадження та використання в реальних умовах роботи ветеринарної клініки та дозволить автоматизувати процеси обліку та полегшити роботу медичного персоналу. Таким чином, реалізація інформаційної системи обліку є важливим етапом в розробці даного проекту, який забезпечує функціональність, надійність та ефективність системи у ветеринарній клініці.

Для того, щоб розпочати роботу, треба мати встановлений Google Chrome та VisiualStudio. Вони завантажуються на встановлюються в стандартному порядку. Далі потрібно завантажити Node.JS (перейти на офіційний сайт https://nodejs.org і на головній сторінці завантажити останню стабільну версію). Після завантаження інсталятор слід запустити і встановити Node.js, як будь-яку іншу програму. Для роботи з Angular необхідно встановити сервер Node.js і пакетний менеджер npm, якщо вони відсутні на

робочій машині. Для установки можна використовувати програму установки node.js. Разом з сервером вона також встановить і npm. При цьому особливого якогось знання для роботи з NodeJS і npm не потрібно. Angular CLI спрощує створення додатка, його компіляцію. Angular CLI поширюється як пакет npm, тому для його використання його необхідно спочатку встановити [8-12]. Для запуску розробки проекту в командному рядку (терміналі) необхідно перейти до папки проекту за допомогою команди cd і потім виконати команду ng serve –open (Рис 3.1.).

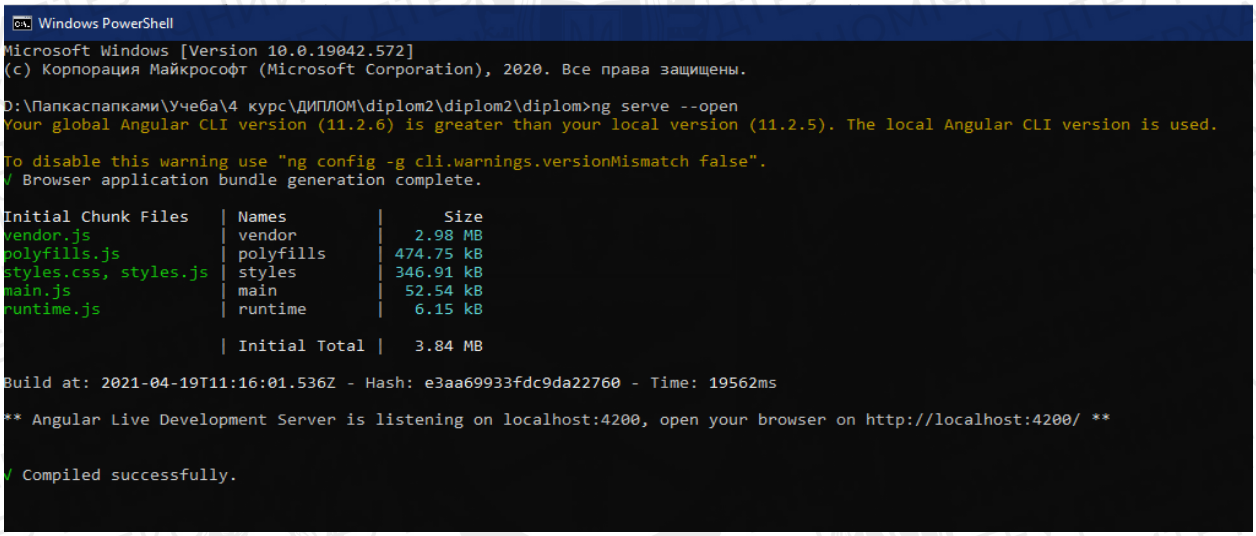

Рис. 3.1. Консоль, яка показує що проект відкритий у браузері.

Консоль проінформує про створені системою файли та їх розміри. Також вона надає адресу, яку використовую працюючий тестовий сервер, за замовченням вона буде - "http: // localhost: 4200 /".

Для початку необхідно створити сторінку через яку користувач буде мати змогу перейти до власного кабінету. Код буде виглядати наступним чином: (Рис.3.2).

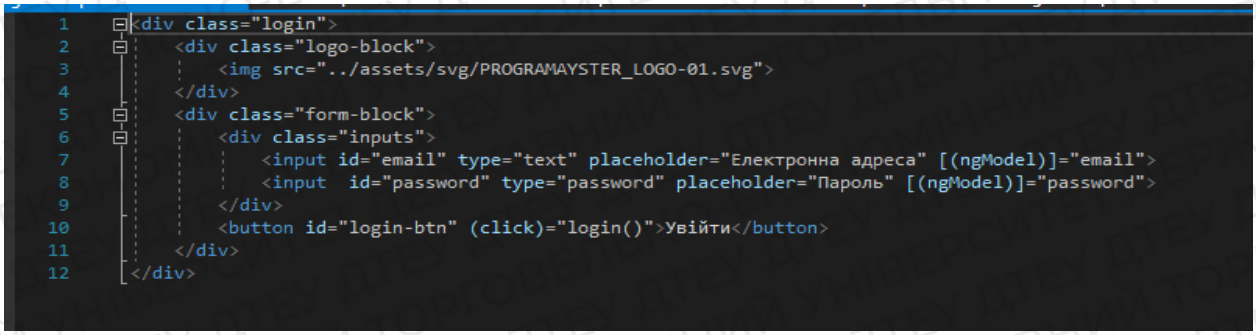

Рис.3.2. Html першої веб-сторінки.

Елемент це блоковий елемент, який призначений для виділення фрагмента документа з метою зміни виду вмісту. Він буде впливати на контент тільки після того як буде стилізований за допомогою CSS. Тег використовують для надання стилів та угруповання блокових елементів. Використовуючи атрибути class або id можна угруповувати контент та стилізувати його, також позначати розділ документа, написаний на різних мовах [11]. Елементи SVG можна змінювати за допомогою атрибутів, які задають докладні відомості про те, яким чином елемент повинен оброблятися. Assets означає набори файлів, які використовуються на HTML-сторінці - це файли CSS, JavaScript, різні шрифти і зображення. Нерідко терміном assets називають тільки набори css- і js-файлів. Атрибут «id» - це унікальне ім'я елемента, яке використовується для зміни його стилю і звернення до нього через скрипти [12]. Атрибут <input> призначений для створення різних кнопок, текстових полів, прапорців, перемикачів [13].

Placeholder виводить текст в текстовій скринці який зникає при отриманні фокусу. Тобто це текст який зникає коли користувач натискає на відповідне поле, та починає вводити дані [14].

Type - Повідомляє браузеру, до якого типу належить елемент форми [15].

[(ngModel)] Ця директива вказує Angular, що необхідно двостороннє зв'язування даних. Що завжди застосовується в директивах input, select, textarea [16].

Button – це кнопка на яку користувач може натиснути.

Click – процес який відбувається після натискання кнопки користувачем.

Результатом всіх цих дій буде веб-сторінка з наступним виглядом(Рис 3.3).

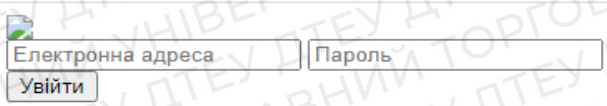

Рис.3.3. Вигляд першої веб-сторінки без CSS.

Тепер необхідно підключити CSS до проекту, щоб зовні вона виглядала більш знайомою для користувача (Рис.3.4).

login.component.scss

```scss
:host {
    background: linear-gradient(90deg, #32327A 25%, #4D4D9B 50%, #32327A 75%);
    display: flex;
    justify-content: center;
    height: 100%;
    width: 100%;
}

.login {
    height: 80%;
    width: fit-content;
    display: flex;
    flex-direction: column;
    justify-content: center;
    font-family: Arial, sans-serif;
}

.login .logo-block {
    height: 200px;
    margin-bottom: 20px;
    display: flex;
    justify-content: center;
}

.login .logo-block img {
    width: 80%;
}

.login .form-block {
    width: 300px;
    height: 200px;
    padding: 30px;
    background-color: white;
    border-radius: 1%;
    display: flex;
    flex-direction: column;
    justify-content: space-around;
}

.login .form-block .inputs {
    width: 100%;
    margin-bottom: 15px;
}

.login .form-block .inputs input {
    width: 100%;
    height: 40px;
    margin-bottom: 5px;
    border-radius: 5px;
    padding: 5px;
    border: 1px solid lightgray;
    text-decoration: none;
    outline: none;

}

.login .form-block .inputs input:focus {
    border-color: #4D4D9B;
    text-decoration: none;
    outline: none;
}

.login .form-block .inputs input:focus::placeholder {
    color: #4D4D9B;
}

.login .form-block button {
    margin-bottom: 5px;
    border-radius: 5px;
    padding: 5px;
    text-decoration: none;
    outline: none;
    border: 2px solid transparent;
    background-color: #838BC5;
    color: white;
    font-size: 18px;
}

.login .form-block button:focus {
    border: 2px solid #4D4D9B;
}

.login .form-block button:active {
    background-color: #5966cc;
}

.login .form-block button:hover {
    cursor: pointer;
}
```

Рис. 3.4. Код CSS першої веб-сторінки.

Функція-псевдоклас: host вибираєносія shadow DOM-a, який містить CSS, який використовується всередині - але тільки якщо селектор, переданий як параметр функції, відповідає shadow хосту. Найбільш поширений спосіб використання його - поставити певний клас тільки на певні екземпляри призначених для користувача елементів, а потім передати відповідний класовий селектор як аргумент функції. Тіньовий DOM («Shadow DOM») використовується для інкапсуляції. Background – фон даної сторінки. Далі був доданий градієнт використовуючи «linear-gradient», та добавленні такі значення як розмір та колір.

Наступним кроком була прописана команда «display: flex».Так у дисплею з'явиться властивість flex-контекст, яка в подальшому дозволить ним управляти набагато простіше, ніж з використанням стандартного CSS. Далі іде властивість justify-content, вона визначає, як браузер розподіляє простір між та навколо елементів контенту уздовж головної осі flex контейнера, або уздовж малої осі grid контейнера. Оскілки за замовченням браузер розташовує контент з лівого боку, необхідно було додати функцію «center». Таким чином контент буде разташован по середині.

CSS атрибут height і width дозволять позначати висоту та ширину елемента. За замовчуванням, властивість визначає висоту і ширину внутрішньої області. Після внесення даних всі створенні блоки будуть розташовані вздовж всього вікна браузеру. Властивість flex-direction задає напрямок основних осей в контейнері і тим самим визначає положення флексів в контейнері. Використовуючи властивість justify-content: center, даємо браузеру зрозуміти що цей блок також розташовується по центру вікна.

Text-decoration створює оформлення тексту у вигляді його підкреслення, лінії над текстом та миготіння. Одночасно можна застосувати більш одного стилю, перераховуючи значення через пробіл. Outline – це властивість, яка встановлює товщину, стиль і колір зовнішнього кордону на всіх чотирьох сторонах елементу. На відміну від лінії, що задається через border, властивість outline не впливає на положення блоку ширину блоку.

Завдяки роботі з CSS сторінка набула вигляд схожий до типових веб-сторінок в інтернеті. Після підключення HTML, CSS і TS до Angular цей файл зв'язано з можливістю обирати групи. (Рис.3.6)

```
import { Component, OnInit } from '@angular/core';
import { UserService } from 'src/app/services/user.service';

@Component({
    selector: 'app-main',
    templateUrl: './main.component.html',
    styleUrls: ['./main.component.scss']
})
export class MainComponent implements OnInit {
    selectedGroup: string;
```

Рис. 3.6. Код для підключення Angular.

Далі необхідно прописати в системі функцію, яка дозволить користувачу вхід до власного кабінету тільки якщо він вкаже дані, які є у системі (Рис.3.7) Тобто вірне поєднання паролю та пошти. Після введення даних користувачем система почне звіряти ці дані з тими, що є у базі, якщо всі дані вірні, то система надає доступ.

```
constructor(private router: Router, private userService: UserService) { }

ngOnInit(): void {
}


login(): void {
  const acc = this.accounts.find(acc => acc.email === this.email && acc.password === this.password);

  if (acc) {
    this.userService.user = acc;
    this.router.navigate(['main'])
  }
}
}
```

Рис. 3.7. Частина коду системи відповідності пошти та паролю.

На цьому можна перейти до створення другої веб-сторінки системи.

У перший блок додається зображення логотипу клініки.(Рис. 3.8.)

main.component.html | main.component.ts | login.component.ts* | login.component.scss

```
<div class="main">
    <header>
        <div class="header-logo">
            <img src="../../assets/svg/PROGRAMAYSTER_LOGO-02.svg" alt="header-logo">
        </div>
```

Рис. 3.8. Код котрий додає зображення логотипу.

Далі слід додати зображення для кнопок меню у верхній частині екрану (Рис.3.9)

```
<div class="header-buttons">
    <div id="btn-home">
        <img src="../../assets/svg/-_home.svg">
    </div>
    <div id="btn-groups">
        <img src="../../assets/svg/-_book.svg">
    </div>
    <div id="btn-mails">
        <img src="../../assets/svg/-_mail.svg">
    </div>
    <div id="btn-settings" (click)="clearStorage()">
        <img src="../../assets/svg/-_settings.svg">
    </div>
</div>
```

Рис.3.9. Код додавання зображення кнопок.

Також слід додати кнопку «Зберегти», для того щоб після натиску на кнопку дані які ввів користувач зберігались у системі. Тег <select> дозволяє створити елемент інтерфейсу у вигляді списку, а також список з одним або множинним вибором, а директива ngIf дозволяє видалити або, навпаки, додати елемент за певної умови. Далі йде директива ngFor що дозволяє перебрати в шаблоні елементи масиву. Властивість target інтерфейсу Event є посиланням на об'єкт, який був ініціатором події. Ці дії можна побачити на рисунку 3.10.

```
<div class="content">
    <div class="workspace">
        <div class="table-head">
            <div class="btn-wrapper">
                <select name="groups" id="groups" *ngIf="groupNames" (change)="changeGroup($event.target.value)">
                    <option>-</option>
                    <option *ngFor="let group of userService.user?.groups" [value]="group">{{groups[group].name}}
                    </option>
                </select>
            </div>
            <h2 id="group-name">{{groups[selectedGroup]?.name || ''}}</h2>
            <div class="btn-wrapper">
                <button class="save-btn" (click)="saveData()">Зберегти</button>
            </div>
        </div>
```

Рис. 3.10. Таблиця для вибору послуги

У результаті цих дій створено табличку, яку користувач буде використовувати для вибору необхідної для нього послуги. Далі необхідно перейти до другої сторінки, для якої потрібно прописати CSS, щоб вона отримала оформлений зовнішній вигляд (Рис.3.11)

```
.main {
    height: 100%;
    background-color: #E1E1E2;
}
```

Рис.3.11. Робота з основним фоном.

Наступним кроком необхідно додати властивості у блок який містить у собі логотип (Рис.3.12.).

```
.main header .header-logo {
    height: auto;
    width: 200px;
}
```

Рис.3.12. Властивості для блоку з логотипом

Далі буде робота над блоком, в якому розташовані декілька верхніх кнопок. Рівномірно розподіляємо всі елементи по ширині flex-блоку, перший елемент спочатку, останній в кінці (рис.3.13).

```
.main header .header-buttons {
    height: 100%;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
```

Рис.3.13 Розподіл елементів для блоку з кнопками.

Наступним кроком є робота з правою частиною екрану, де розташоване ПІБ клієнта, його пошта, та зображення. Додаємо властивості для тексту у вигляді зміни шрифту на Arial, Helvetica, та sans-serif (рис.3.14).

```
.main header .header-info {
    width: fit-content;
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: Arial, Helvetica, sans-serif;
}
```

Рис.3.14. Робота з блоком ПІБ клієнта.

Далі треба перейти до роботи з контуром таблиці(Рис.3.15). Колір вікна можна зробити світло-синього кольору, а текст у центрі блока змінити на білий.

```css
table {
  border: 1px solid black;
  border-spacing: 0 0;

    select {
      background-color: #838BC5;
      color: white;
      outline: none;
      border: none;
      text-decoration: none;
      margin-bottom: 5px;
      border-radius: 5px;
      font-size: 18px;
      padding: 7px;
    }


    .td-input {
      display: flex;
      justify-content: center;
      width: fit-content;

      input {
        text-decoration: none;
        outline: none;
        border: none;
        text-align: center;
        font-size: 18px;
        width: 50px;
      }
    }


}
```

Рис.3.15. Робота з зовнішнім виглядом таблиці журналу обліку клієнтів.

На Рис.3.16 представлено програмний код, який дозволить користувачеві обирати необхідну йому послугу.

```typescript
  get groupNames(): string[] {
    return Object.keys(this.groups);
  }

  constructor(public userService: UserService) { }

  ngOnInit(): void {
    this.loadData()
  }

  changeGroup(value) {
    this.selectedGroup = value;
  }

  saveData(): void {
    localStorage.setItem('groups', JSON.stringify(this.groups))
  }

  loadData(): void {
    const data = localStorage.getItem('groups');

    if (data) {
      this.groups = JSON.parse(data);
    } else {
      this.groups = this.reservedGroups;
    }
  }


  clearStorage(): void {
    localStorage.clear()
  }
}
```

Рис.3.16. Код який дозволяє обирати послугу

3.3. Тестування роботи системи.

Тестування є важливим етапом в процесі розробки, оскільки дозволяє перевірити функціональність, надійність та відповідність системи поставленим вимогам та очікуванням.

Під час тестування було виконано наступні кроки:

1. Функціональне тестування: було перевірено правильність роботи основних функцій системи, таких як реєстрація пацієнтів, запис на прийом, ведення медичної картки, формування звітів та інші. Кожна функція була перевірена на коректність введення даних, обробку та відображення результатів.
2. Навантажувальне тестування: було проведено тестування системи під навантаженням, щоб перевірити її працездатність та швидкодію при великій кількості одночасних запитів. Використовувалися спеціальні інструменти для створення великого навантаження на систему та вимірювання її продуктивності.
3. Тестування безпеки: було перевірено рівень безпеки системи, зокрема захищеність від несанкціонованого доступу до даних, збереження конфіденційності пацієнтів та інші аспекти безпеки. Проводилися атаки та спроби злому системи для виявлення можливих вразливостей та їх виправлення.
4. Тестування відповідності вимогам: у цьому етапі було перевірено, наскільки система відповідає вимогам, встановленим у попередніх розділах. Кожна вимога була перевірена окремо, забезпечуючи її повне виконання. Результати тестування свідчать про те, що система повністю задовольняє поставлені вимоги і відповідає очікуванням замовника.
5. Тестування в реальних умовах: з метою перевірки функціональності та продуктивності системи в реальних умовах роботи, було проведено тестування на практиці. Система була впроваджена у ветеринарну клініку, і протягом певного періоду спостерігалася його робота. Під час цього тестування було звернуто увагу на реакцію системи на реальні ситуації та її

взаємодію з користувачами. Результати тестування показали, що система успішно працює в реальних умовах та забезпечує зручне та ефективне ведення обліку у ветеринарній клініці.

6. Висновки з тестування роботи системи підтверджують, що розроблена інформаційна система обліку у ветеринарній клініці є функціональною, надійною, ефективною та відповідає встановленим вимогам. Вона демонструє стабільну роботу навіть при великому навантаженні та забезпечує високий рівень безпеки. Тестування в реальних умовах підтвердило готовність системи до практичного використання у ветеринарній клініці.

7. Враховуючи результати тестування, можна стверджувати, що розроблена система обліку у ветеринарній клініці є інструментом для покращення організації роботи клініки, спрощення процесів обліку та поліпшення якості надання ветеринарних послуг.

Результати тестування свідчать, що функції працюють коректно, система продемонструвала стабільну роботу навіть при великому навантаженні. Крім того, система повністю відповідає встановленим вимогам та сприяє полегшенню процесів обліку у ветеринарній клініці. Тестування роботи системи є важливим кроком перед її впровадженням та використанням в практичній роботі. Воно дозволяє виявити та виправити можливі помилки та недоліки, забезпечити стабільну та надійну роботу системи.

Розроблена інформаційна система обліку у ветеринарній клініці є ефективним інструментом для поліпшення роботи клініки, оптимізації процесів обліку та покращення якості надання ветеринарних послуг. Вона дозволяє зручно та ефективно вести облік пацієнтів, їх медичну історію, взаємодіяти з клієнтами, контролювати запаси та здійснювати фінансовий облік. Результати розробки та тестування показали, що система відповідає всім поставленим вимогам, є функціональною, надійною та ефективною. Вона забезпечує зручний та швидкий доступ до інформації, спрощує рутинні процеси та допомагає забезпечити точність та достовірність даних. Розроблена система має потенціал для подальшого розширення та вдосконалення. За

допомогою регулярних оновлень та впровадження нових функцій, вона може стати ще більш потужним інструментом для управління ветеринарною клінікою та покращення її ефективності. Враховуючи вище зазначене, можна зробити висновок про успішне виконання розробки автоматизованої системи обліку у ветеринарній клініці. Результати свідчать про важливість використання сучасних технологій та інформаційних систем у ветеринарній сфері для поліпшення організації роботи, підвищення ефективності та надання якісних послуг клієнтам.

ВИСНОВКИ

У рамках даної випускної кваліфікаційної роботи була проведена розробка автоматизованої системи обліку у ветеринарній клініці з метою поліпшення організації роботи та якості надання ветеринарних послуг. Робота складається з трьох розділів, кожен з яких відображає важливі аспекти розробки та впровадження інформаційної системи. У першому розділі проведено аналіз організації діяльності ветеринарних клінік в Україні, огляд існуючих інформаційних систем та сформульовано вимоги до автоматизованої системи обліку. У другому розділі була розроблена концепція розробки системи обліку та детально описано проектування додатку для обліку у ветеринарній клініці. Також було обґрунтовано вибір засобів розробки інформаційної системи. У третьому розділі було описано процес розробки інформаційного забезпечення системи обліку, розроблено алгоритм роботи системи та описано результати роботи системи.

На основі проведеного дослідження та розробки можна зробити висновок, що розроблена автоматизована система обліку у ветеринарній клініці буде інструментом для покращення організації роботи клініки, спрощення процесів обліку та поліпшення якості надання ветеринарних послуг. Запропонована система дає можливість ефективного ведення обліку пацієнтів, медичних карт, прийому та розкладу візитів, складання звітності та інші функціональних можливості, що сприяють підвищенню продуктивності. Система дозволяє зменшити час на виконання завдань, покращити точність обліку, забезпечити надійне зберігання даних та покращити взаємодію з клієнтами. Отже, розроблена автоматизована система обліку у ветеринарній клініці буде ефективним інструментом для покращення роботи клініки, забезпечення точного обліку та задоволення потреб клієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Денісова О. О., Автоматизоване проектування інформаційних систем : навчальний посібник / О.О. Денісова ; Міністерство освіти і науки, молоді та спорту України, Державний вищий навчальний заклад "Київський національний економічний університет імені Вадима Гетьмана". - Київ : КНЕУ, 2011. - 412 с.
2. Катренко А.В., Системний аналіз об'єктів та процесів комп'ютеризації : навчальний посібник /А.В. Катренко Системний аналіз об'єктів та процесів комп'ютеризації Львів:"Новий світ-2000".-2003.-424с.
3. Шаховська Н. Б. Проектування інформаційних систем : навчальний посібник / Н. Б. Шаховська, В. В. Литвин ; за наук. ред. В. В. Пасічника ; М-во освіти і науки України. - Л. : Магнолія 2006, 2011. - 380 с.
4. Гломозда Д. К., Проектування, системний аналіз і розробка корпоративних інформаційних систем : навчальний посібник / Гломозда Дмитро ; Нац. ун-т "Києво-Могилян. акад.". - Київ : [НаУКМА], 2015. - 95 с.
5. Трегуб В. Г., Проектування систем автоматизації : навчальний посібник для студентів вищих навчальних закладів / В. Г. Трегуб ; М-во освіти і науки України, Нац. ун-т харчових технологій. - К. : Ліра-К, 2014. - 341 с.
6. Марченко А.В. Проектування інформаційних систем [електронний ресурс] / А. В. Марченко. – К., 2016. – Режим доступу: http://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20151030212747/content-20151030212747.pdf
7. Стаття про переваги Gulp. [Електронний ресурс] – Режим доступу до ресурсу: https://frontender.info/no-need-to-grunt-take-a-gulp-of-fresh-air/
8. Офіційний сайт Node.js. [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/uk/.
9. Офіційний сайт MongoDB. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/2.
10. NPM. [Електронний ресурс] – Режим доступу до ресурсу: https://www.npmjs.com/.

11. Юрчак І. Ю. Конспект лекцій. Веб-проектування [Електронний ресурс] / І. Ю. Юрчак. – Режим доступу : http://victoria.lviv.ua/html/gim/document.html.
12. Білоус Л. Ф. Інформаційні мережі : навч. посібник / Л. Ф. Білоус Інформаційні мережі : навч. посібник – К. : Логос, 2005. – 140 с.
13. Павленко Л. А. Корпоративні інформаційні системи: Навчаль¬ний посібник./ Л. А. Павленко - Харків: ВД "ІНЖЕК", 2005. – 260
14. Катренко А.В., Системний аналіз об'єктів та процесів комп'ютеризації : Навчаль-ний посібник./А.В. Катренко - Львів: "Новий світ-2000".-2003.-424с.
15. Інфографіка: навчальний посібник / упорядник Гудіма О. В. – Чернівці, Чернівецький національний університет, 2017. – 107 с.
16. Кобилін А. М. Системи обробки економічної інформації : навчальний посібник / А. М. Кобилін . – Київ : Центр учбової літератури, 2019. – 234 с.
17. Павленко Л. А. Корпоративні інформаційні системи: Навчаль¬ний посібник. - Харків: ВД "ІНЖЕК", 2005. - 260 с.
18. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. / В. В. Пасічник., В. А. Резніченко Організація баз даних та знань: Навчаль¬ний посібник – К.: Видавнича група BHV, 2006. – 384 с.
19. Аналіз контенту веб-сайтів українських центрів науково-технічної інформації з точки зору використання інтернет-сервісів - Людмила Філіпова - ISSN 2076-9326. Вісник Книжкової палати. 2012. № 10.
20. Способи унікалізації контенту на сайті [Електронний ресурс]. – Режим доступу: http://andrey.lviv.ua/blog/sposobi-unikalizatsii-kontentu
21. Контент сайту та його складові [Електронний ресурс]. – Режим доступу:. http://webstudio2u.net/ua/studio-web/686-kontent-saita.html .\
22. Розробка WEB – додатків [Електронний ресурс]. – Режим доступу: https://tqm.com.ua/ua/rozrobka-veb-dodatkiv-servisiv-web-application-development/rozrobka-web-dodatkiv-dlia-bizniesu#klient-servernye%20https://metanit.com/sharp/aspnet5/3.1.php .
23. Відомості ASP.NET Core MVC, функції, структура та компоненти [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/ru-

ru/aspnet/core/mvc/overview?view=aspnetcore-5.0.

24. Переваги ASP.NET Core MVC [Електронний ресурс]. – Режим доступу: https://professorweb.ru/my/ASP_NET/mvc/level1/1_2.php
25. Основні види тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymy/.

ДОДАТОК А

```
div class="login">
  <div class="logo-block">
    <img src="../assets/svg/PROGRAMAYSTER_LOGO-01.svg">
  </div>
  <div class="form-block">
    <div class="inputs">
      <input id="email" type="text" placeholder="Електронна адреса"
[(ngModel)]="email">
      <input id="password" type="password" placeholder="Пароль"
[(ngModel)]="password">
    </div>
    <button id="login-btn" (click)="login()">Увійти</button>
  </div>
</div>

:host {
  background: linear-gradient(90deg, #32327A 25%, #4D4D9B 50%, #32327A
75%);
  display: flex;
  justify-content: center;
  height: 100%;
  width: 100%;
}
.login {
  height: 80%;
  width: fit-content;
  display: flex;
  flex-direction: column;
```

```
    justify-content: center;
    font-family: Arial, sans-serif;
}
.login .logo-block {
    height: 200px;
    margin-bottom: 20px;
    display: flex;
    justify-content: center;
}
.login .logo-block img {
    width: 80%;
}
.login .form-block {
    width: 300px;
    height: 200px;
    padding: 30px;
    background-color: white;
    border-radius: 1%;
    display: flex;
    flex-direction: column;
    justify-content: space-around;
}
.login .form-block .inputs {
    width: 100%;
    margin-bottom: 15px;
}
.login .form-block .inputs input {
    width: 100%;
    height: 40px;
    margin-bottom: 5px;
```

```
    border-radius: 5px;
    padding: 5px;
    border: 1px solid lightgray;
    text-decoration: none;
    outline: none;

}
.login .form-block .inputs input:focus {
    border-color: #4D4D9B;
    text-decoration: none;
    outline: none;
}
.login .form-block .inputs input:focus::placeholder {
    color: #4D4D9B;
}
.login .form-block button {
    margin-bottom: 5px;
    border-radius: 5px;
    padding: 5px;
    text-decoration: none;
    outline: none;
    border: 2px solid transparent;
    background-color: #838BC5;
    color: white;
    font-size: 18px;
}
.login .form-block button:focus {
    border: 2px solid #4D4D9B;
}
.login .form-block button:active {
```

```
  background-color: #5966cc;
}
.login .form-block button:hover {
  cursor: pointer;
}
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { UserService } from 'src/app/services/user.service';
@Component({
 selector: 'app-login',
 templateUrl: './login.component.html',
 styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
 email: string;
 password: string;accounts = [
  {
   name: "Дьяченко Дар’я Володимирівна",
   email: 'vikladach@gmail.com',
   password: '1234',
   groups: [
    'group1',
    'group3'
   ]
  },
  {
   name: "Кравчук Антон Валерійович",
   email: 'vikladach2@gmail.com',
   password: '4321',
   groups: [
```

```
      'group2',
      'group3'
    ]
  },
];
constructor(private router: Router, private userService: UserService) { }
ngOnInit(): void {
}
login(): void {
  const acc = this.accounts.find(acc => acc.email === this.email && acc.password
=== this.password);

  if (acc) {
    this.userService.user = acc;
    this.router.navigate(['main'])
  }
}
}
  </div>
      <div id="btn-mails">
        <img src="../../assets/svg/-_mail.svg">
      </div>
      <div id="btn-settings" (click)="clearStorage()">
        <img src="../../assets/svg/-_settings.svg">
      </div>
    </div>

    <div class="header-info">
      <div class="text">
        <div id="name">{{userService.user?.name || ''}}</div>
```

```
            <div id="email">{{userService.user?.email || "}}</div>
          </div>
          <img src="../../assets/svg/user-01.svg" alt="photo">
        </div>
      </header>
    <div class="content">
      <div class="workspace">
        <div class="table-head">
          <div class="btn-wrapper">
            <select name="groups" id="groups" *ngIf="groupNames"
(change)="changeGroup($event.target.value)">
              <option>-</option>
              <option *ngFor="let group of userService.user?.groups"
[value]="group">{{groups[group].name}}
              </option>
            </select>
          </div>
          <h2 id="group-name">{{groups[selectedGroup]?.name || "}}</h2>
          <div class="btn-wrapper">
            <button class="save-btn" (click)="saveData()">Зберегти</button>
          </div>
        </div>
        <div class="table-wrapper">
          <table *ngIf="selectedGroup">
            <tr>
              <th>#</th>
              <th>ПІБ</th>
              <th *ngFor="let d of days">{{d}}</th>
              <th>Екзамен</th>
            </tr>
```

```
            <tr *ngFor="let student of groups[selectedGroup]?.data; let i = index">
              <td>{{i}}</td>
              <td>{{student.name}}</td>
              <td *ngFor="let d of days">
                <div class="td-input">
                  <input type="text" [(ngModel)]="student.marks[d]">
                </div>
              </td>
              <td>
                <select name="exam" id="exam" [(ngModel)]="student.exam">
                  <option [value]="false">Не зданий</option>
                  <option [value]="true">Зданий</option>
                </select>
              </td>
            </tr>
          </table>
        </div>
      </div>
    </div>
  </div>
```

CSS друга сторінка

```
.main header .header-info #email {
  font-size: 0.8rem;
  text-align: right;
  color: #CDC8FF;
}
.main header .header-info img {
  width: 50px;
  margin-left: 5px;
}
```

```
.main .content {
    height: 90%;
    width: 100%;
    display: flex;
    justify-content: center;
}
.main .content .workspace {
    width: 70%;
    height: 100%;
    font-family: Arial, Helvetica, sans-serif;
}
.main .content .workspace .table-head {
    margin: 5px 0;
    display: flex;
    justify-content: space-between;
}
.main .content .workspace .table-wrapper {
    display: flex;
    justify-content: center;
}
#groups {
    background-color: #838BC5;
    color: white;
    outline: none;
    border: none;
    text-decoration: none;
    margin-bottom: 5px;
    border-radius: 5px;
    font-size: 18px;
    padding: 7px;
```

```
}
#groups:active {
    border: none;
    text-decoration: none;
    outline: none;
}
#group-name {
    font-size: 40px;
}
.main .content .workspace .table-wrapper table {
    margin-top: 30px;
    background-color: white;
}
th {
    background-color: #838BC5;
}
th, td {
    border: 1px solid black;
    padding: 10px;
}
table {
 border: 1px solid black;
 border-spacing: 0 0;
select {
        background-color: #838BC5;
        color: white;
        outline: none;
        border: none;
        text-decoration: none;
        margin-bottom: 5px;
```

```
        border-radius: 5px;
        font-size: 18px;
        padding: 7px;
    }
.td-input {
        display: flex;
        justify-content: center;
        width: fit-content;
input {
            text-decoration: none;
            outline: none;
            border: none;
            text-align: center;
            font-size: 18px;
            width: 50px;
        }
    }
}
.btn-wrapper {
    display: flex;
    align-items: center;
}
button.save-btn {
    margin-bottom: 5px;
    border-radius: 5px;
    padding: 5px;
    text-decoration: none;
    outline: none;
    border: 2px solid transparent;
    background-color: #838BC5;
```

```
  color: white;
  font-size: 18px;
}
button.save-btn:focus {
  border: 2px solid #4D4D9B;
}
button.save-btn:active {
  background-color: #5966cc;
}
import { Component, OnInit } from '@angular/core';
import { UserService } from 'src/app/services/user.service';

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.scss']
})
export class MainComponent implements OnInit {
  selectedGroup: string;

  days = ["05.09", "12.09", "19.09", "26.09", "05.10", "12.10"];
  groups: any;

  reservedGroups = {
    group1: {
      name: 'Трупа №1',
      data: [
        {
          name: "Крамаренко Михайло Сергійович",
            marks: {
```

```
        "05.09": "15",
        "12.09": "15",
        "19.09": "15",
        "26.09": "15",
        "05.10": "15",
        "12.10": "15"
      },
      exam: true
    },
  {
    name: "Середа Олександр Володимирович",
      marks: {
        "05.09": "15",
        "19.09": "15",
        "26.09": "15",
        "12.10": "15"
      },
      exam: true
    },
    {
      name: "Васильчук Анна Михайлівна",
  marks: {
      "05.09": "15",
      "12.09": "10",
      "19.09": "15",
      "26.09": "15",
      "05.10": "15",
      "12.10": "15"
    },
      exam: true
```

```
    },
    {
      name: "Коваленко Олена Вікторівна",
      marks: {
        "05.09": "15",
        "12.09": "15",
        "19.09": "15",
        "26.09": "15",
        "05.10": "15",
        "12.10": "15"
      },
      exam: true
    },
    {
      name: "Антоненко Денис Денисович",
      marks: {
        "05.09": "15",
        "12.09": "н",
        "19.09": "н",
        "26.09": "15",
        "05.10": "н",
        "12.10": "н"
      },
      exam: true
    },

  ]
},
group2: {
  name: 'Група №2',
```

```
data: [
  {
   name: "Мороз Марк Сергійович",
     marks: {
       "05.09": "15",
       "12.09": "15",
       "19.09": "15",
       "26.09": "15",
       "05.10": "15",
       "12.10": "15"
     },
     exam: false
   },
   {
    name: "Швець Дарина Олегівна",
     marks: {
      "05.09": "15",
      "12.09": "15",
      "19.09": "15",
      "26.09": "15",
      "05.10": "15",
      "12.10": "15"
     },
     exam: true
   },
    {
     name: "Карпенко Інна Тимофіївна",
   marks: {
    "05.09": "15",
    "12.09": "10",
```

```
    "19.09": "15",
    "26.09": "15",
    "05.10": "15",
    "12.10": "15"
   },
   exam: false
  },
  {
   name: "Гончар Іван Миколайович",
   marks: {
    "05.09": "15",
    "12.09": "10",
    "19.09": "15",
    "26.09": "15",
    "05.10": "15",
    "12.10": "15"
   },
   exam: false
  },
  {
   name: "Вовк Володимир Валерійович",
   marks: {
    "05.09": "15",
    "12.09": "10",
    "19.09": "15",
    "26.09": "15",
    "05.10": "15",
    "12.10": "15"
   },
   exam: false
```

```
    },
   ]
},
group3: {
  name: 'Група №3',
 data: [{
  name: "Панасюк Анастасія Віталіївна ",
  marks: {
   "05.09": "10",
   "12.09": "10",
   "19.09": "н",
   "26.09": "н",
   "05.10": "н",
   "12.10": "н"
  },
  exam: false
 },
  {
   name: "Журба Микита Андрійович",
   marks: {
    "05.09": "15",
    "12.09": "10",
    "19.09": "15",
    "26.09": "15",
    "05.10": "15",
    "12.10": "15"
   },
   exam: false
  },
  {
```

```
  name: "Шевченко Діана Олександрівна ",
  marks: {
   "05.09": "5",
   "12.09": "10",
   "19.09": "10",
   "26.09": "5",
   "05.10": "н",
   "12.10": "15"
  },
  exam: false
 },
 {
  name: "Тарасенко Ольга Петрівна",
  marks: {
   "05.09": "15",
   "12.09": "10",
   "19.09": "н",
   "26.09": "15",
   "05.10": "15",
   "12.10": "15"
  },
  exam: true
 },
 {
  name: "Бондаренко Олексій Дмитрович",
  marks: {
   "05.09": "н",
   "12.09": "н",
   "19.09": "н",
   "26.09": "н",
```

```
            "05.10": "10",
            "12.10": "15"
          },
          exam: false
        },
      ]
    },
  }get groupNames(): string[] {
    return Object.keys(this.groups);
  }
 constructor(public userService: UserService) { }
ngOnInit(): void {
    this.loadData()
  }
 changeGroup(value) {
    this.selectedGroup = value;
  }
 saveData(): void {
    localStorage.setItem('groups', JSON.stringify(this.groups))
  } loadData(): void {
    const data = localStorage.getItem('groups');
 if (data) {
      this.groups = JSON.parse(data);
    } else {
      this.groups = this.reservedGroups;
    }
  }
clearStorage(): void {
    localStorage.clear()
  }
```