Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Кутнича Томаша Томашовича*
(ПІБ)

академічної групи *121-19-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему:

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Реута О.В.*			
розділів:				
спеціальний	*доц. Реута О.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*проф Гнатушенко В.В.*			
Нормоконтролер	*ст. викл Мартиненко А.А.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-2* *Кутнича Т.Т.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-застосунку для автоматизації планування поточних справ на платформі ASP.NET*

Development of the web-application for automation of current activities planning based on ASP.NET

затверджена наказом ректора НТУ «ДП» від *16 травня* 2023 *р.* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Реута О. В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Кутнич Т.Т.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023р.*

РЕФЕРАТ

Пояснювальна записка: 129 с., 49 рис., 3 дод., 28 джерел.

Об'єкт розробки: Веб-застосунок для автоматизації планування поточних справ на платформі ASP.NET.

Мета кваліфікаційної роботи: Розробити веб-застосунок для автоматизації планування поточних справ на платформі ASP.NET.

У вступі виконується аналіз сучасного стану проблеми, уточнюється постановка завдання, мета кваліфікаційної роботи та галузь її застосування, обґрунтовується актуальність теми.

У першому розділі проводиться дослідження предметної галузі таіснуючих рішень, визначається актуальність завдання та призначення розробки, розроблюється постановка завдання.

У другому розділі обирається платформа для розробки, виконується проектування програми і її розробка, наводиться опис алгоритму і структури функціонування системи, визначаються вхідні і вихідні дані, наводяться характеристики складу параметрів технічних засобів, описується робота програми.

В економічному розділі визначається трудомісткість розробленого програмного продукту, проводиться підрахунок вартості роботи по створенню застосунку та розраховується час на його створення.

Практичне значення роботи полягає в тому, що створене програмне забезпечення може бути використане для полегшення планування та відстеження запланованих задач та прогресу їх виконання.

Актуальність веб застосунку визначається потребою людей в інструменті для зручної організації планування.

Список ключових слів: Веб застосунок, планування, список завдань, Asp.Net.

ABSTRACT

Explanatory note: 129 p., 49 figs., 3 apps., 28 sources.

Object of development: Web-application for automation of current activities planning based on ASP.NET.

Purpose of the qualification work: Web-application for automation of current activities planning based on ASP.NET.

In the introduction it is analyzed the current state of the problem, clarified the problem, the purpose of the qualification work and the scope of its application, substantiated the relevance of the topic.

In the first section the research of the subject area and existing decisions is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed.

In the second section the platform for development is chosen, the program design and its development are carried out, the description of algorithm and structure of functioning of system is given, input and output data are defined, characteristics of structure of parameters of technical means are given, work of the program is described.

In the economic section it is determined the complexity of the developed software product, calculated the cost of work to create an application and calculated the time to create it.

The practical value is to develop software that might be used to make planning and tracking of tasks easier.

The relevance of the web application is determined by people`s need for the tool for comfortable organization of planning.

Keywords: Web application, planning, ToDo List, Asp.Net.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

БД – база даних;

ООП – об’єктно-орієнтоване програмування;

ОС – операційна система;

ПК – персональний комп'ютер;

ПЗ – програмне забезпечення;

СУБД – система управління базою даних;

SaaS – software as a service, програмне забезпечення як послуга;

HTTP – hypertext transfer protocol, протокол передачі гіпертексту;

URL – uniform resource locator, уніфікований покажчик інформаційного ресурсу.

ЗМІСТ

ВСТУП

Наразі завдання стрімко ускладнюються, розпорошуються. Це сильно ускладнює процес планування та відстеження поточних завдань. До того ж багато людей стикається з нестачею організації, використовують неефективні методи для планування: завдання можуть бути розсіяними між різними записками, електронними документами або навіть у пам'яті. До того ж багато людей не мають чіткого систематичного підходу до планування та відстеження своїх завдань. Вони можуть не мати ефективної методології або інструментів для організації своєї роботи. У сучасному світі люди все більше орієнтовані на мобільність та доступність. Це ставить вимоги до зручних інструментів, які були б доступні з різних пристроїв та місць.

Саме тому було вирішено розробити продукт, який не потребує довгого та складного налаштування, встановлення додаткових програм, не займає пам’яті на пристрої користувача і має простий та інтуїтивний інтерфейс щоб швидко освоїтися і користуватися.

Враховуючи ці фактори, розробка веб-застосунку може бути актуальним рішенням для вирішення проблеми труднощів планування та відстеження поточних завдань. Ось деякі переваги, які такий веб-сайт може пропонувати:

- Централізоване планування: Веб-застосунок може надати централізовану платформу, де користувачі зможуть створювати, організовувати та відстежувати свої завдання. Всі дані будуть зберігатися на одному місці, що спрощує управління ними.
- Нагадування та сповіщення: Веб-застосунок може надати можливість встановлювати нагадування про терміни завершення завдань або важливих подій. Такі сповіщення можуть бути відправлені на електронну пошту або мобільний пристрій користувача, допомагаючи уникнути забуття або пропуску дедлайнів.
- Спільна робота та комунікація: Веб-застосунок може створити

можливість для спільної роботи над завданнями та комунікації між колегами. Це дозволить забезпечити чітку взаємодію, обмін інформацією та вирішення спільних завдань.

- Мобільний доступ: Розробка веб-застосунку, який пристосований для мобільних пристроїв, дозволить користувачам мати доступ до своїх завдань з будь-якого місця і в будь-який час. Це зручно для людей, що працюють з різних місць або перебувають у дорозі.
- Аналітика та звітність: Веб-застосунок може надати інструменти для аналізу прогресу, статистики та звітності щодо виконання завдань. Це допоможе користувачам оцінювати свою продуктивність та виявляти області для поліпшення.

Розробка веб-застосунку для вирішення проблеми труднощів планування та відстеження поточних завдань може мати значний позитивний вплив на організацію та продуктивність. Зважайте на потреби та вимоги користувачів, а також на доступні технологічні рішення для досягнення найкращого результату.

Мета кваліфікаційної роботи це розробка Web-застосунку для полегшення організації планування та відстеження поточних завдань. Інформаційна система буде надавати можливість авторизації, додавання, редагування та видалення завдань, їх сортування та фільтрації, перегляду деталей конкретного завдання. У завдань буде кілька параметрів, а саме:

- Назва
- Категорія
- Опис завдання
- Крайній строк виконання
- Статус виконання

Для досягнення мети були виділені наступні завдання:

- Проаналізувати актуальність та потребу
- Провести аналіз наявних аналогів
- Сформулювати вимоги до Інформаційної системи
- Описати інструменти для розробки

- Розробити алгоритм та структуру програми
- Розробити Web-застосунок

Практичне завдання роботи полягає у можливості використання розробленого програмного забезпечення для полегшення планування, зручної організації завдань та їх відстеження. Це може бути корисно для різних людей, оскільки у цього програмного забезпечення немає якогось професійного спрямування. До того ж Web-застосунок може працювати на абсолютно різних платформах.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Поняття та актуальність теми

Сучасний світ характеризується великою кількістю інформації, з якою люди повинні працювати. Завдання можуть надходити з різних джерел, включаючи електронну пошту, соціальні медіа, проектні менеджери тощо. Це може призводити до розсіяності та ускладнювати процес планування та відстеження завдань. Також багато людей не мають чіткого та систематичного підходу до планування та відстеження своїх завдань. Вони можуть використовувати різні методики, відмічати завдання на папері або в електронних документах, що може призводити до незручностей та недоліків у організації.

Саме це наштовхнуло мене на думку про створення програмного забезпечення для допомоги вирішення цієї проблеми. Вивчаючи варіанти для цього, я прийшов до висновку реалізації проекту у вигляді Web-застосунку.
Веб-додатки мають кілька переваг порівняно з іншими типами додатків. Ось деякі переваги веб-додатків, згідно з результатами мого вивчення:

- Доступність: Веб-додатки можна отримати з будь-якого місця з підключенням до Інтернету, що робить їх більш доступними, ніж традиційні настільні додатки.[1]
- Компатибельність з різними платформами: Веб-додатки можна отримати з будь-якого пристрою з веб-браузером, незалежно від операційної системи.[1]
- Масштабованість: Веб-додатки легко масштабуються вгору або вниз для врахування змін в потребах користувачів.[1]
- Ефективність витрат: Веб-додатки можуть бути ефективнішими з точки зору витрат, оскільки вони не вимагають встановлення або підтримки на окремих пристроях.[1]

- Простота оновлень: Веб-додатки можна оновлювати централізовано, що полегшує забезпечення того, що всі користувачі використовують останню версію додатку.[1]
- Зручність для користувачів: Веб-додатки можуть бути розроблені з урахуванням зручності користувача, з такими функціями, як адаптивний дизайн і прогресивні веб-додатки.[1]

Веб-додаток (веб-аплікація) - це програма, яка зберігається на віддаленому сервері і постачається через Інтернет за допомогою інтерфейсу браузера. Веб-сервіси, за визначенням, є веб-додатками, і багато веб-сайтів містять веб-додатки, хоча не всі. Розробники створюють веб-додатки для різноманітних цілей та категорій користувачів - від організацій до окремих осіб. До типових веб-додатків входять веб-пошта, онлайн-калькулятори або інтернет-магазини. Хоча до деяких веб-додатків можна отримати доступ лише за допомогою певного браузера, більшість з них доступні незалежно від використовуваного браузера. [3]

Веб-додаток працює наступним чином:[4]

Архітектура веб-додатків ґрунтується на клієнт-серверній моделі. Код таких додатків розділяється на дві складові: клієнтські скрипти та серверні скрипти.

Архітектура на стороні клієнта:

Клієнтські скрипти відповідають за функціонал інтерфейсу користувача, такий як кнопки та випадаючі списки. Коли кінцевий користувач натискає посилання на веб-додаток, веб-браузер завантажує клієнтські скрипти та відтворює графічні елементи та текст для взаємодії з користувачем. Наприклад, користувач може читати контент, дивитися відео або заповнювати форми з контактними даними. Дії, такі як натискання кнопки відправки, переходять на сервер у вигляді запиту від клієнта.

Архітектура на стороні сервера:

Серверні скрипти відповідають за обробку даних. Сервер веб-додатка

оброблює запити від клієнтів та надсилає відповіді. Запити зазвичай стосуються отримання додаткових даних, редагування або збереження нових даних. Наприклад, якщо користувач натискає кнопку "Читати більше", серверний скрипт відправляє вміст назад користувачеві. Якщо користувач натискає кнопку "Відправити", серверний скрипт зберігає дані користувача у базі даних. У деяких випадках сервер виконує запит на дані та надсилає повну HTML-сторінку клієнту. Це називається серверною рендеризацією.

1.1.2. Огляд наявних програмних аналогів

Перед початком розробки того чи іншого програмного продукту, потрібно розглянути вже існуючи рішення, дослідити, проаналізувати їх переваги та недоліки. Це допоможе зрозуміти, яких помилок слід уникати та які функції запровадити, розробляючи свою програму. Проаналізувавши тематичні веб додатки я виділив найпопулярніші: https://todoist.com, https://to-do.live.com, https://app.any.do.

1. https://todoist.com – веб сайт, що дає змогу полегшити своє планування (рис. 1.1).

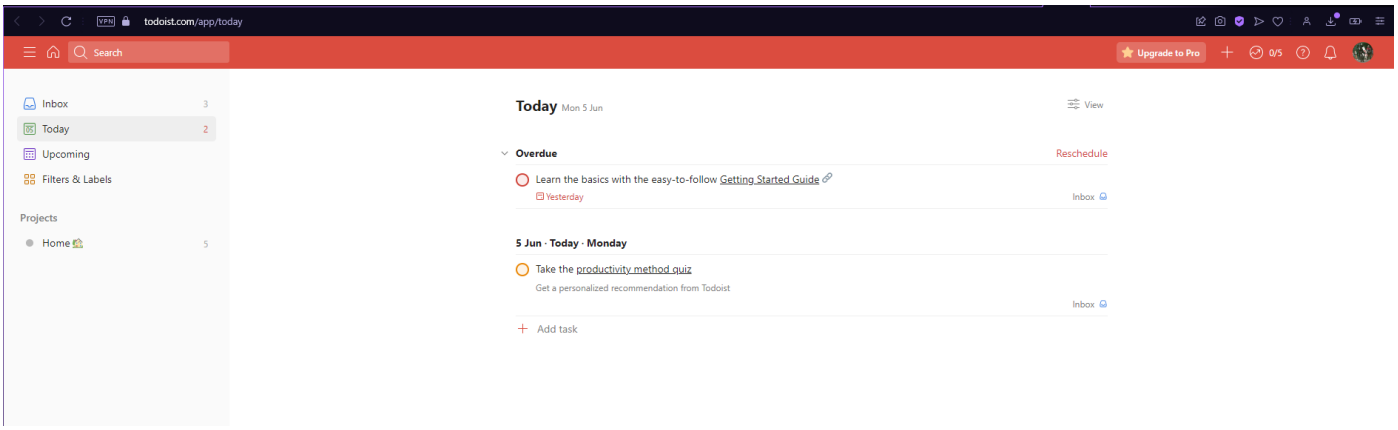

Рис. 1.1 Приклад роботи todolist.com

Основний функціонал Todoist.com включає наступне:

Створення завдань: Ви можете легко створювати нові завдання, встановлювати для них назву, опис, дату та час виконання, пріоритет, мітки та інші важливі атрибути. Організація завдань: Ви можете створювати проекти та підпроекти для групування ваших завдань по різних категоріях або проектах. Ви

також можете використовувати мітки для подальшої класифікації та сортування завдань. Керування термінами: Todoist надає можливість встановлення дедлайнів для завдань. Ви можете визначити конкретну дату та час, до якого повинно бути виконане завдання. Система також надсилає сповіщення про наближення термінів. Календарний перегляд: Ви можете переглядати свої завдання в календарному форматі, що дозволяє зручно планувати свій час і бачити, коли ви маєте виконати певне завдання. Спільна робота: Todoist дозволяє запрошувати інших користувачів до спільних проектів. Ви можете надавати їм доступ до завдань і спілкуватися з ними, розподіляти обов'язки та спільно працювати над проектами. Прикріплення файлів: Ви можете додавати файли до завдань для зручного зберігання документів, зображень або інших додаткових матеріалів, пов'язаних із завданням. Підтримка на різних платформах: Todoist працює на веб-платформі та надає додатки для різних операційних систем, таких як Windows, macOS, iOS, Android, інтеграцію з Gmail, Outlook та іншими сервісами.

Переваги:

- Гнучкі налаштування завдань
- Можливість фільтрації, для зручного пошуку
- Календар для зручного відображення
- Кросплатформеність
- Можливість вибрати мову

Недоліки:

- Занадто складний інтерфейс
- Обмежені можливості для безоплатної версії

2. https://app.any.do – додаток , який дає заміну записнику, допомагає будувати та нагадувати про плани. Веб додаток виглядає як на рис. 1.2

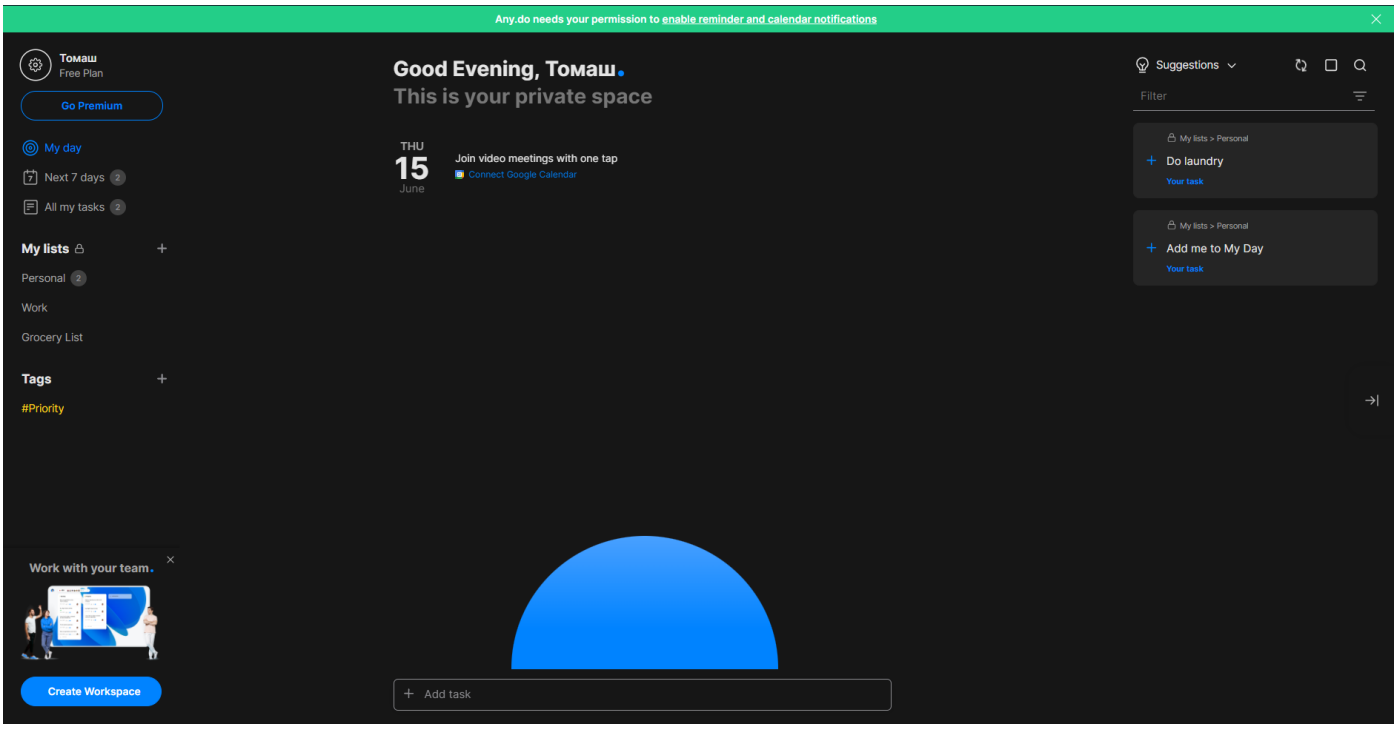

Рис. 1.2 Приклад роботи todolist.com

Після старту вас зустріне головне меню, де можна обрати такі розділи як:

- Мій день
- Наступні 7
- Усі задачі
- Календар
- Проект
- Особисті доручення
- Список покупок

Якщо обрали "мій день" з'явиться рядок для вводу та чекбокс при введенні завдання. Також можна поставити час в який треба нагадати про завдання та виставити його пріоритетність. При виконанні завдання

можна відмітити його натиснувши на чекбокс та видалити. Також є можливість розбити завдання на підпункти.

У розділі "наступні 7 днів" маєте можливість планування справ на весь тиждень. Для запису завдань алгоритм дій схожий на попередній.

Вибравши розділ " усі завдання " ви зможете розподілити завдання на:

- " сьогодні "
- " завтра "
- "майбутні "
- " без строку"

Розділ " список покупок " надає можливість спланувати покупки та відсортувати їх автоматично у певні категорії. У налаштуваннях можна видалити увесь список, певну категорію, відмітити усе або навпаки очистити.

Також є можливість створювати свої "дошки " які потім можна розбити на списки.

У розділі " Календар" є можливість синхронізувати свій календар з додатком де будуть відображатися заплановані події та нагадувати про них.

Переваги:

- синхронізування з календарем
- можливість виставити нагадування про завдання
- цікаві та корисні розділи

- можливість розбити завдання на підзавдання
- Зміна колірної теми

Недоліки:

- необхідність платної підписки для розкриття всіх можливостей
- видалення завдання одразу після відмічення як виконаного

1.2. Призначення розробки та галузь застосування

Кваліфікаційна роботи призначена для створення Веб сайт, що полегшить процес планування та відстеження завдань.

Мета даного дипломного проекту це реалізація Веб сайту для організації завдань, який дасть змогу зручно планувати справи та відстежувати їх.

TodoList – це асинхронний веб застосунок, який дозволяєзручно планувати свої завдання. Він дозволить користувачам створювати завдання та редагувати, відстежувати їх. Створення завдання супроводжується формою для надання потрібних даних. Особливістю програми є простий, інтуїтивний дизайн, можливість сортування та фільтрації завдань а також можливість зручного їх налаштування, та інше.

Розроблюваний сайт призначений для використання будь яким юзером. Програмний продукт не має якогось професійного спрямування, тож не буде викликати додаткових труднощей.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка веб-застосунку для автоматизації планування поточних справ на платформі ASP.NET» є наказ по Національному технічному університету «Дніпровська політехніка» від 16.05.2023 р. № 350-с.

1.4. Постановка завдання

Метою завдання є розробка програмного забезпечення для полегшення планування на прикладі веб застосунку. Завдяки платформі не потрібно встановлювати додаткове програмне забезпечення.

Основними характеристиками продукту буде:

- Можливість реєстрації та авторизації
- Можливість додавати, редагувати та видаляти завдання
- Можливість відстежувати завдання
- Можливість подивитися деталі завдання
- Можливість сортування та фільтрації завдань

Завдання буде реалізовано при виконанні наступних вимог:

- ознайомлення з C# ASP.NET MVC;
- вибір необхідних інструментів для реалізації;
- розробка структури проекту;
- написання коду застосунку;
- результатом буде вважатись працюючий локально веб застосунок.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Функціональні вимоги до бота:

- дружній та зрозумілий інтерфейс;
- можливість змінити користувача;
- збереження даних для підтвердження верифікації у БД;
- можливість додавати, редагувати, видаляти завдання з БД;

1.5.2. Вимоги до інформаційної безпеки

Щоб бот надійно працював є такі вимоги до інформаційної безпеки:

- доступність інформації для користувачів;
- здійснювати захист від несанкціонованого доступу;
- безперервний доступ до мережі.

1.5.3. Вимоги до складу та параметрів технічних засобів

Щоб користуватися ботом необхідно мати:

- будь-яка операційна система з додатком Telegram;
- доступ до мережі Інтернет.

1.5.4. Вимоги до інформаційної та програмної сумісності

Вимоги до програмного забезпечення:

- Встановлений браузер;

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Даний веб сайт призначений для полегшення планування та відстеження завдань, і має наступні функціональні характеристики:

- ведення в базу даних користувачів, їх даних;
- взаємодія з базою даних;
- можливість створення, редагування, видалення завдань;
- можливість сортування та фільтрації завдань.

Веб сайт буде мати експлуатаційні характеристики, такі як:

- простий, інтуїтивний інтерфейс;
- індивідуальні завдання для кожного користувача;

2.2. Опис застосованих математичних методів

Під час розробки даного програмного продукти не використовувались математичні методи.

2.3. Опис використаної архітектури та шаблонів проектування

При проектуванні застосунку потрібно приділити особливу увагу його архітектурі. Недостатньо працююча структура може створити проблеми під час розробки та випуску програмного продукту. Додаток з добре продуманою архітектурою буде більш простим у масштабуванні, зміні та взаємодії з ним.

Завдяки розвитку глобальної мережі та збільшенню обсягу даних, які потрібно зберігати, обробляти та передавати, відбулася еволюція традиційної клієнт-серверної моделі, і з'явилися хмарні обчислення. Хмарні обчислення не є

новою технологією, вони відрізняються лише способом представлення обчислювальних ресурсів [5]. Існує декілька моделей хмарних служб за рівнями:

- Infrastructure-as-a-service (IaaS), інфраструктура як послуга, коли споживач використовує обчислювальні ресурси постачальника (сервер, мережеву інфраструктуру, сховище даних);
- Platform-as-a-service (PaaS), платформа як послуга, коли постачальник надає споживачеві доступ до використання програмної платформи;
- Software-as-a-service (SaaS), програмне забезпечення як послуга, коли споживач може користуватися готовою програмою постачальника.

Я обрав модель Програмне забезпечення як послуга(далі SaaS), через певні переваги. Наприклад:

- SaaS може надавати економію витрат, використання ресурсів, контроль версій та багато іншого через багатокористувацьку модель.[6]
- SaaS може бути ефективною альтернативою локальному програмному забезпеченню з багатьма перевагами, такими як якість системи, якість обслуговування та довіра.[7]
- SaaS може використовуватися для забезпечення безпеки інформації, що є важливим для онлайн-бізнесу з урахуванням кіберзагроз.[8]

У синхронному коді встановлюється послідовна черга операцій, де кожна операція очікує завершення попередньої. Це може створювати проблеми, особливо коли веб-застосунок використовується багатьма користувачами. Щоб уникнути непотрібних затримок, було прийнято рішення використовувати асинхронну модель у серверній частині для обробки запитів. Крім того, асинхронний код може видалити блокуючу операцію з основного потоку, дозволяючи їй виконуватися паралельно, тим самим дозволяючи обробнику переходити до наступних операцій. Порівняння між асинхронністю і синхронністю можна побачити на рис 2.1

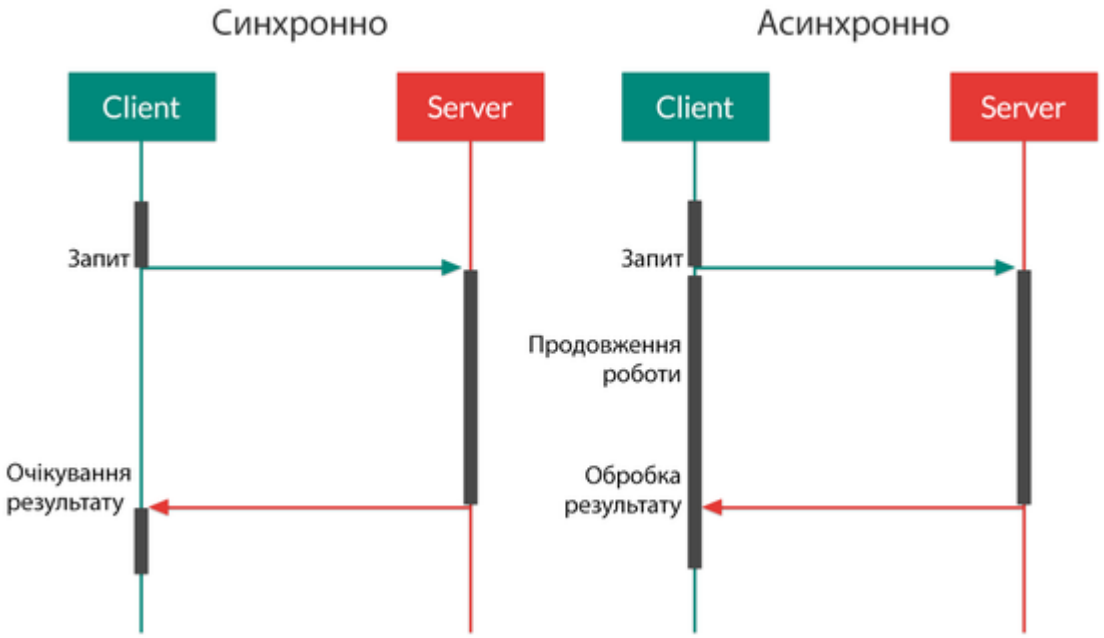

Рис. 2.1 Діаграма порівняння асинхронності і синхронності.

2.4. Опис використаних технологій та мов програмування

Даний веб додаток був розроблений за допомогою таких технологій:

- C# Asp.Net MVC;
- Entity Framework Core;
- MSSQL;
- Html;

ASP.NET Core є кросплатформовим, високопродуктивним та відкритим фреймворком для створення сучасних, хмарно-орієнтованих та підключених до Інтернету додатків. ASP.NET Core дає змогу:[2]

- Створювати веб-додатки та сервіси, додатки для Інтернету речей (IoT) та мобільні підсистеми.
- Використовувати свої улюблені інструменти розробки на Windows, macOS та Linux.
- Розгортати додатки в хмарі або на місцевому сервері.
- Виконувати на .NET Core.

ASP.NET Core MVC надає функціональність для створення веб-API та веб-додатків:[2]

- Шаблон Model-View-Controller (MVC) допомагає зробити ваші веб-API та веб-додатки тестовими.
- Razor Pages є моделлю програмування на основі сторінок, яка полегшує створення веб-інтерфейсу та підвищує продуктивність.
- Мовний синтаксис Razor надає продуктивну синтаксичну конструкцію для сторінок Razor та представлень MVC.
- Tag Helpers дозволяють серверному коду брати участь у створенні та відображенні HTML-елементів у файлах Razor.
- Вбудована підтримка для різних форматів даних та контенту дозволяє вашим веб-API досягати широкого спектру клієнтів, включаючи браузери та мобільні пристрої.
- Прив'язка моделі автоматично відображає дані з HTTP-запитів в параметри методів дій.
- Валідація моделі автоматично здійснює клієнтську та серверну валідацію.

Entity Framework Core є легким, розширюваним та кросплатформеним варіантом Entity Framework, популярного фреймворку для відображення об'єктів/реляційного моделювання (O/RM) для .NET. Ось деякі ключові моменти про Entity Framework Core:

- Entity Framework Core розроблений для роботи з різноманітними базами даних, включаючи SQL Server, MySQL, PostgreSQL та SQLite.[9]
- Entity Framework Core є платформою доступу до даних, орієнтованою на модель, з великою кількістю нових концепцій та патернів, які розробники можуть вивчити.[9]
- Entity Framework Core підтримує LINQ-запити, відстеження змін та

міграції бази даних.[9]

- Entity Framework Core може використовуватися разом з ASP.NET Core для створення сучасних веб-додатків.[8]
- Entity Framework Core є популярним вибором для доступу до даних у додатках .NET Core. [8]

Загалом, Entity Framework Core є потужним і гнучким інструментом для роботи з базами даних у додатках .NET Core.

MS SQL Server - це система управління реляційними базами даних, розроблена компанією Microsoft. Ось деякі ключові моменти про MS SQL Server:

- MS SQL Server є широко використовуваною системою управління базами даних для корпоративних застосунків, електронно-комерційних веб-сайтів та інформаційних платформ.[11]
- MS SQL Server є складною системою, яка потребує належного налаштування безпеки для уникнення проблем з безпекою.[12]
- MS SQL Server підтримує мову структурованих запитів (SQL) для отримання та аудиту інформації.[13]
- MS SQL Server може використовуватися для мультиреляційного видобутку даних та вивчення дерев прийняття рішень.[14]
- MS SQL Server пропонує економію витрат, масштабованість, високу доступність, відновлення після катастрофи, кращу продуктивність та легке управління при використанні в хмарі.[14]

Загалом, MS SQL Server є потужною та широко використовуваною системою управління базами даних, яка пропонує різноманітні функції та переваги для корпоративних застосунків.

HTML (Hypertext Markup Language) - це мова розмітки, яка використовується для структурування контенту в Інтернеті. Вона використовується для створення веб-сторінок та веб-додатків. Ось деякі

ключові моменти про HTML:

- HTML використовується для структурування контенту в Інтернеті, включаючи текст, зображення та мультимедіа.[15]
- HTML є мовою розмітки, яка використовує теги для визначення структури та вмісту веб-сторінки.[15]
- HTML є невід'ємною частиною веб-розробки і використовується спільно з іншими технологіями, такими як CSS та JavaScript.[16]
- HTML може бути використаний для створення адаптивних веб-сторінок, які адаптуються до різних розмірів екранів та пристроїв.[16]
- HTML може бути використаний для створення доступних веб-сторінок, які можуть використовувати люди з обмеженими можливостями.[16]
- HTML може бути використаний для створення веб-додатків за допомогою фреймворків, таких як ASP.NET MVC.[16]

Загалом, HTML є основною технологією для веб-розробки, використовується для структурування контенту в Інтернеті та створення веб-сторінок та веб-додатків.

2.5. Опис структури програми та алгоритмів її функціонування

На рисунку 2.2 візуалізовано схему маршрутизації клієнтської панелі.

Із головної сторінки можна перейти на сторінки логін, фільтрація, створити завдання, редагувати завдання, переглянути конкретне завдання.

Із сторінки логін на головну сторінку та сторінку реєстрації.

Із сторінки реєстрація на сторінку логін

Із сторінок фільтрація, переглянути конкретне завдання, редагувати завдання, створити завдання на головну сторінку.

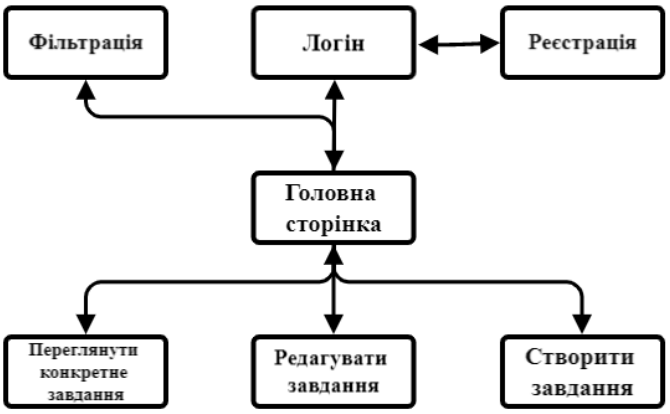

Рис. 2.2. Схема маршрутизації клієнтської панелі

Для роботи веб застосунку використовується база даних MSSQL. У базі даних зберігається вся інформація про користувачів та їх завдання. Схему бази даних можна побачити на рис. 2.3.

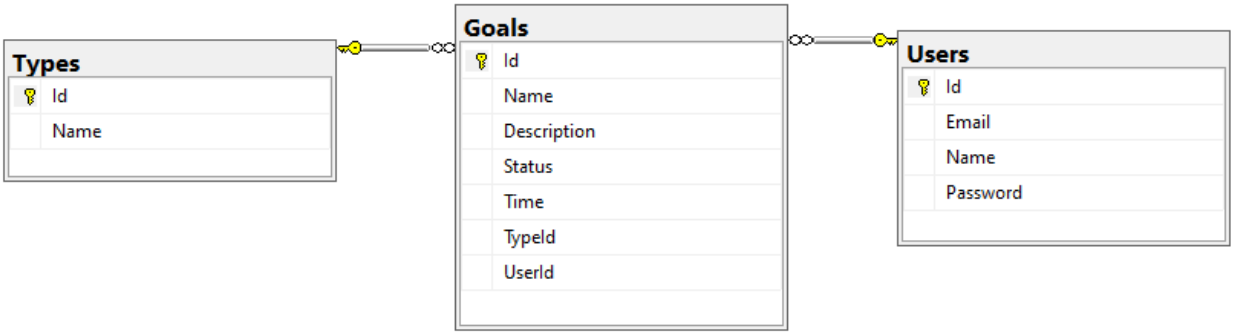

Рис. 2.3. Схема бази даних

Вона містить в собі 3 таблиці:

- Types – зберігає інформацію про типи, а саме: унікальний ідентифікатор та назву
- Goals – зберігає інформацію про завдання, а саме: унікальний ідентифікатор, назву, опис, статус виконання, термін виконання, ідентифікатор типу та ідентифікатор користувача
- Users – зберігає інформацію про користувачів, а саме: унікальний

ідентифікатор, ім’я користувача, електронна пошта, вказана користувачем та пароль

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані це текстові дані з інформацією про користувача, його завдання та їх категорії, що отримуються внаслідок заповнення форм. Дані мають різні типи, що описані в таблицях сутностей 2.1-2.3 Вихідні ж дані це звернення до полів об’єктів та вивід їх у вигляді тексту.

Таблиця 2.1

Таблиця «Users»

Назва поля	Тип даних
Id	Int
Email	string
Name	string
Password	string

Таблиця 2.2

Таблиця «Goals»

Назва поля	Тип даних
Id	Int
Name	string
Description	string
Status	boolean
Time	datetime
TypeId	int
UserId	int

Таблиця 2.3

Таблиця «Types»

Назва поля	Тип даних
Id	int
Name	string

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки веб застосунку використовувався комп'ютер з наступними параметрами:

- операційна система: Windows 10;
- чотириядерний процесор з частотою більше 2,4 ГГц;
- об'єм оперативної пам'яті 16 Гб;
- вільний об'єм пам'яті на твердотілому накопичувачі 70,6Гб;
- доступ до мережі Інтернет.

2.7.2. Використані програмні засоби

Під час розробки були використані такі програмні засоби:

- Microsoft Visual Studio
- SQL Server Management Studio
- GitHub

Інтегроване середовище розробки (IDE) Visual Studio є творчою стартовою площадкою, яку ви можете використовувати для редагування, налагодження та збирання коду, а потім публікувати програму. Окрім стандартного редактора і засобів налагодження, які надають більшість інших IDE, Visual Studio включає компілятори, інструменти автодоповнення коду, графічні дизайнери та багато інших функцій для поліпшення процесу розробки програмного забезпечення.[19]

SQL Server Management Studio (SSMS) - це інтегроване середовище для

керування будь-якою SQL-інфраструктурою. Використовуйте SSMS для доступу, конфігурації, управління, адміністрування та розробки всіх компонентів SQL Server, Azure SQL Database, Azure SQL Managed Instance, SQL Server на Azure VM та Azure Synapse Analytics. SSMS надає один зручний інструмент, який поєднує широкий набір графічних засобів з багатьма потужними редакторами сценаріїв для доступу до SQL Server для розробників та адміністраторів баз даних різного рівня кваліфікації.[20]

GitHub є платформою онлайн-хостингу репозиторіїв, яка надає повний набір можливостей розподіленого контролю версій та управління вихідним кодом, включаючи всі функції Git і набагато більше. Крім того, GitHub пропонує додаткові функції, такі як контроль доступу, систему відстеження проблем (багтрекінг), керування завданнями та вікі для кожного проекту. [21]

2.7.3. Виклик та завантаження програми

Для початку користування потрібно розпакувати архів ToDoList.rar до каталогу ваших проектів, відкрити у IDE, що підтримує ASP.NET(наприклад Microsoft Visual Studio) та запустити. після цього відкриється вкладка з розробленим сайтом.

2.7.4. Опис інтерфейсу користувача

Користувача зустрічає головна сторінка, яка при неавторизованому користувачу виглядає як на рис. 2.4

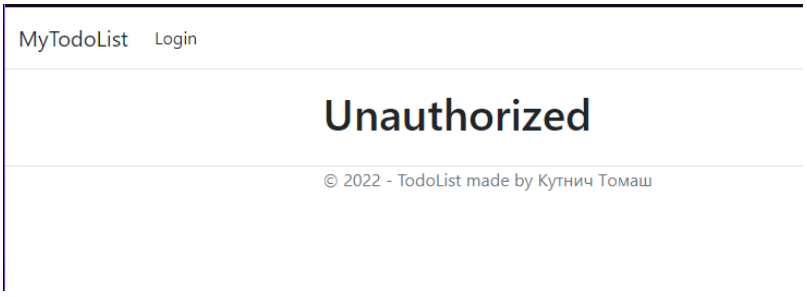

Рис. 2.4. Початкова сторінка для неавторизованих

Єдине посилання на сторінці веде на сторінку для авторизації рис. 2.5

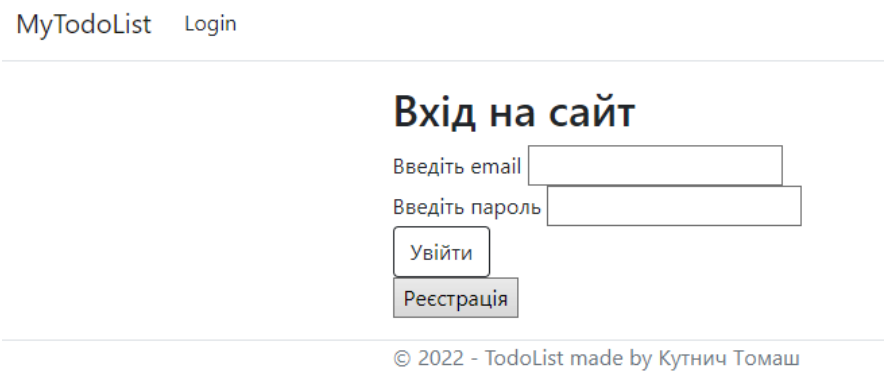

Рис. 2.5. Сторінка авторизації

У випадку якщо користувач не має аккаунту, він може перейти на сторінку реєстрації рис. 2.6 натиснувши на відповідну кнопку.

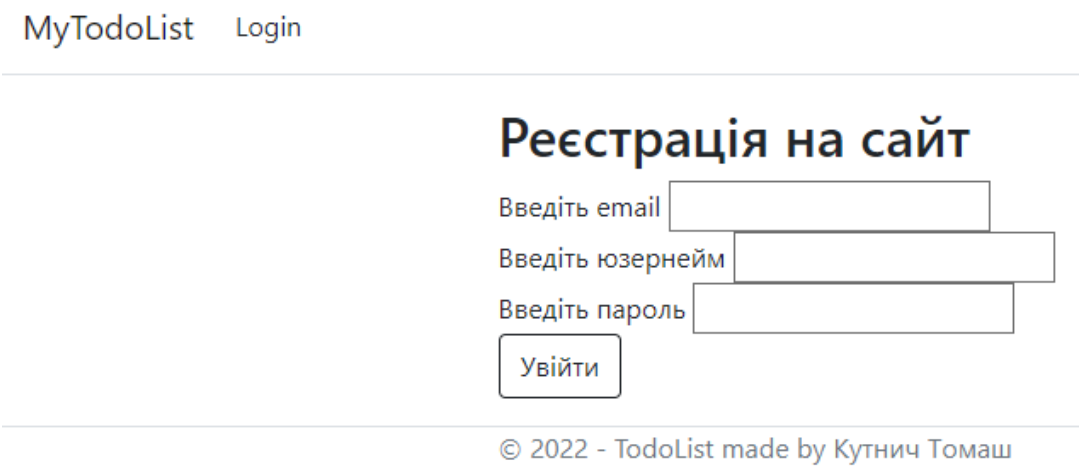

Рис. 2.6. Сторінка реєстрації

Після реєстрації користувача переадресовують на сторінку логін, після авторизації ж користувача переадресовують знову на головну сторінку, яка вже виглядає як на рис. 2.7

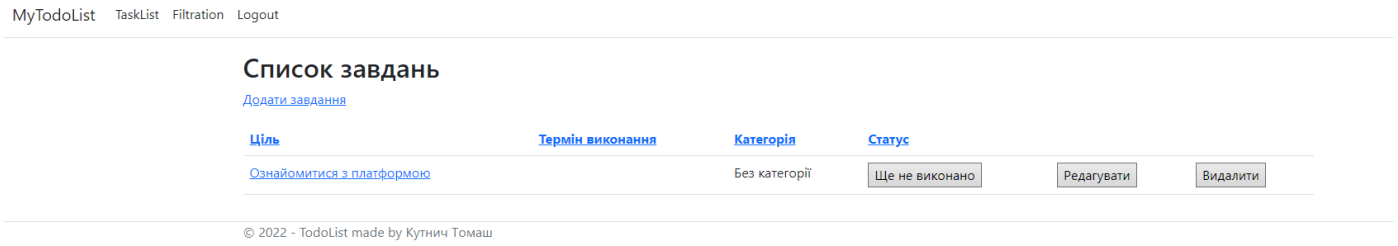

Рис. 2.7. Головна сторінка після авторизації

Згори є посилання на головну сторінку, сторінку фільтрації рис. 2.8 та кнопка для виходу із системи. Посилання Додати завдання веде на сторінку створення завдання рис. 2.9 Кнопки Ціль, Термін виконання, Категорія та Статус сортують список завдань по відповідним категоріям. При повторному натисненні можна змінити сортування за зростанням/за спаданням. При натисненні на назву завдання йде переадресація на сторінку конкретного завдання рис. 2.11 Кнопка редагувати веде на сторінку редагування завдання рис. 2.12 Кнопка видалити видаляє завдання. Кнопка в стовпці статус змінює своє значення в залежності від статусу завдання, та при натисненні змінює його на протилежний.

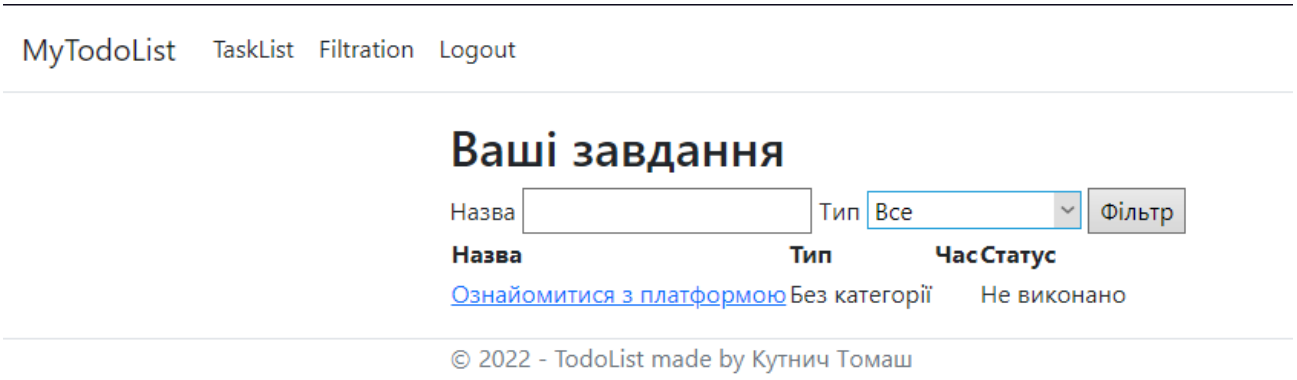

Рис. 2.8. Сторінка фільтрації

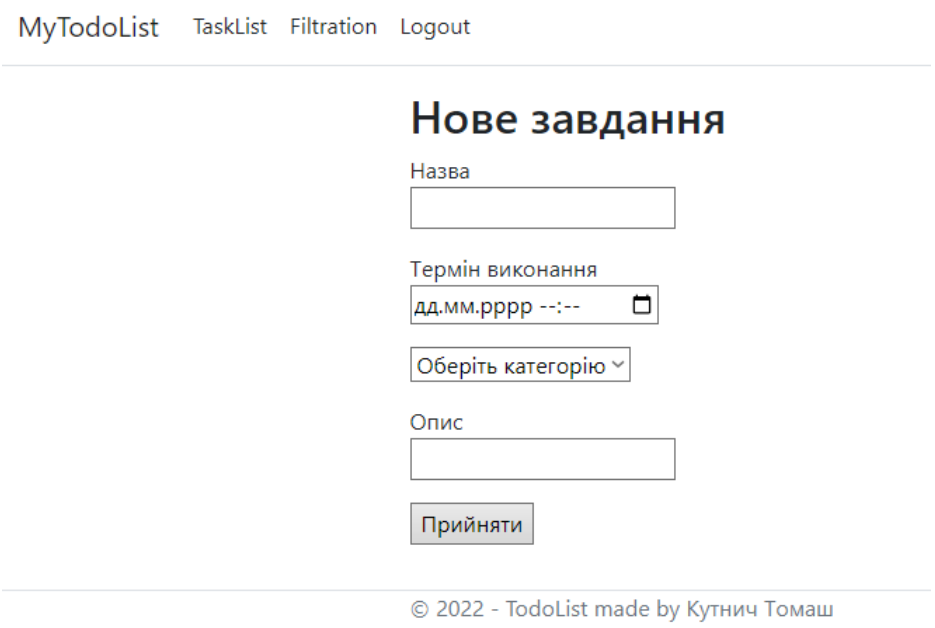

Рис. 2.9. Сторінка створення завдання

На сторінці створення завдання є форма, після заповнення потрібно натиснути кнопку прийняти. Після цього завдання буде збережено, а користувача переадресує на головну сторінку. Якщо не заповнити критичні поля у формі, то при натисненні прийняти сторінка оновиться із збереженими даними для дозаповнення рис.2.10

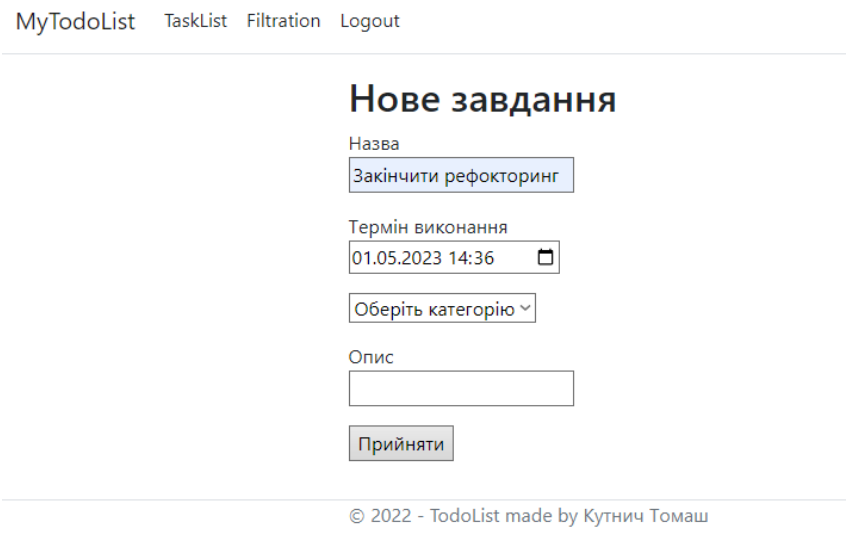

Рис. 2.10. Недостатньо даних для завдання

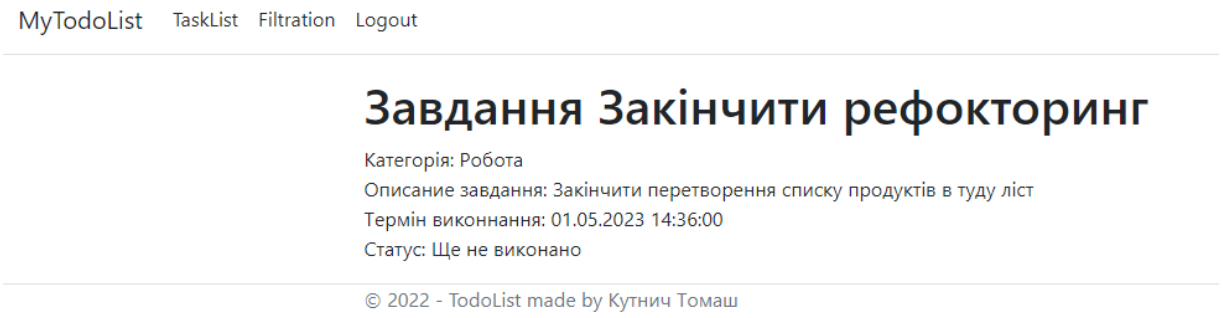

Рис. 2.11. Сторінка завдання

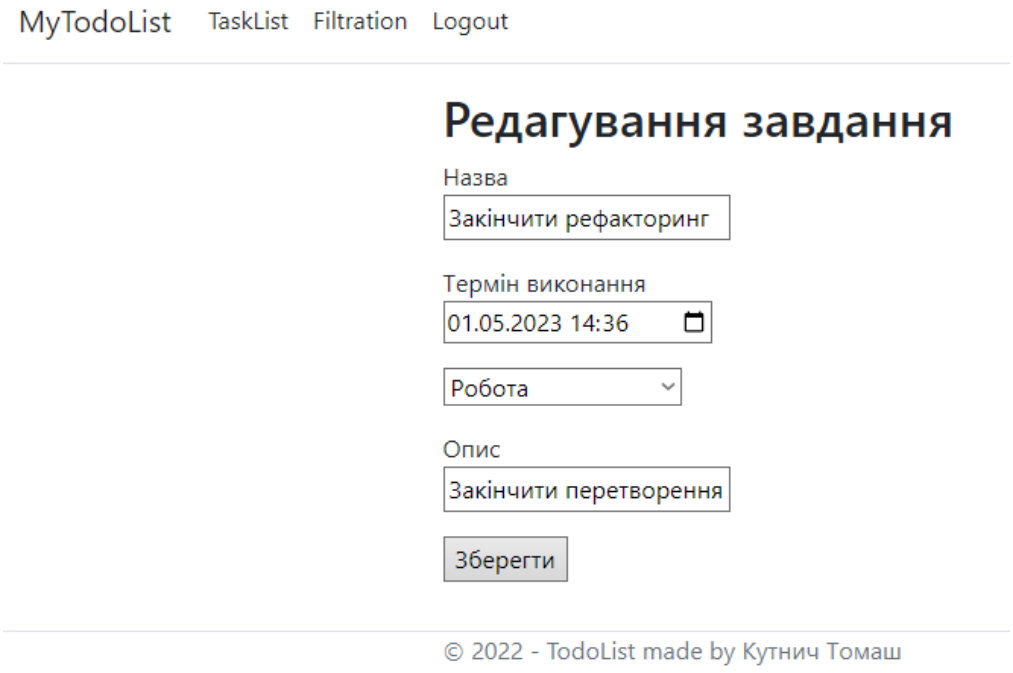

Рис. 2.12. Сторінка редагування завдання

На сторінці редагування завдання є форма, після заповнення потрібно натиснути кнопку прийняти. Після цього завдання буде збережено, а користувача переадресує на головну сторінку. Якщо не заповнити критичні поля у формі, то при натисненні прийняти сторінка оновиться із збереженими даними для дозаповнення

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вихідні дані:

1. передбачуване число операторів програми – 1186;
2. коефіцієнт складності програми – 1,25;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата розробника – 115 грн/год[10];
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,1;
7. вартість машино-години ЕОМ – 20 грн/год (1 грн – е/е, 10 грн – ПЗ, 9 грн - амортизація).

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки. Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де q – передбачуване число операторів (1186);

C – коефіцієнт складності програми (1,25);

p – коефіцієнт корекції програми в ході її розробки (0,2).

$$Q = 1186 \cdot 1{,}25 \cdot (1 + 0{,}2) = 1779;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot K}, \text{ людино-годин} \qquad (3.3)$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

К – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1,1);

$$t_u = \frac{1779 \cdot 1{,}2}{80 \cdot 1{,}1} = 24{,}2 \text{ людино-години.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}; \qquad (3.4)$$

$$t_a = \frac{1779}{20 \cdot 1{,}1} = 80{,}86 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot K}; \qquad (3.5)$$

$$t_{п} = \frac{1779}{20 \cdot 1,1} = 80,86 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

− за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5) \cdot K}; \qquad (3.6)$$

$$t_{отл} = \frac{1779}{5 \cdot 1,1} = 323,5 \text{ людино-години.}$$

− за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,2 \cdot t_{отл}; \qquad (3.7)$$

$$t_{отл}^{к} = 1,2 \cdot 323,5 = 388,2 \text{ людино-години.}$$

Витрати праці на підготовку документації:

$$t_{д} = t_{др} + t_{до}; \qquad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_{д} = \frac{Q}{(15...20) \cdot K}; \qquad (3.9)$$

$$t_{др} = \frac{1779}{20 \cdot 1,1} = 80,8 \text{ людино-години.}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0,75 \cdot t_{др}; \qquad (3.10)$$

$$t_{до} = 0,75 \cdot 80,8 = 60,64 \text{ людино-години.}$$

$$t_{\partial} = 80{,}8 + 60{,}64 = 141{,}44 \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 24{,}2 + 80{,}86 + 80{,}86 + 388{,}2 + 141{,}44 = 765{,}56 \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 765,56 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $Кпо$ ючають витрати на заробітну плату виконавця програми і витрат м $З_{ЗП}$ инного часу, необхідного на налагодження програми на ЕОМ.

$$Кпо = З_{ЗП} + З_{МВ}, \text{ грн,} \qquad (3.11)$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата розробника становить 115 грн/год, то отримаємо:

$$З_{ЗП} = 765{,}56 \cdot 115 = 88\,039{,}4 \text{ грн}$$

Вартість машинного часу $З_{МВ}$, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_М \text{, грн,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год;

$C_М$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 388{,}2 \cdot 20 = 7764 \text{ грн}$$

Звідси витрати на створення програмного продукту:

$$Кпо = 88\,039{,}4 + 7764 = 95\,803{,}4 \text{ грн}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{, мес.} \qquad (3.14)$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 30 годинному робочому тижні F_p = 132 годин).

Витрати на створення програмного продукту:

$$T = \frac{765{,}56}{1 \cdot 132} = 5{,}8 \text{ міс.}$$

Вартість розробки веб додатку становить 88 039,4 грн. Час розробки очікується приблизно 5,8 місяців при 30-годинному робочому тижні і 132-годинному робочому місяці. Цей термін включає час, необхідний для дослідження, розробки алгоритму, дизайну і документування. Загальна кількістю людино-годин, яку буде витрачено на розробку – 765,56.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи був створений веб додаток для полегшення планування та відстеження завдань.

Метою кваліфікаційної роботи є розробка веб додатку для полегшення планування та відстеження завдань. Інформаційна система надає можливість зареєструватися, додати, змінити, редагувати та видалити завдання, та відстежити вже додані.

Під час виконання кваліфікаційної роботи були виконані наступні задачі:

- проаналізовано актуальність та потребу;
- зроблено огляд наявних програмних аналогів;
- сформульовані вимоги до інформаційної системи;
- описані інструменти для розробки;
- розроблений алгоритм та структура програми;
- створений веб додаток

Для реалізації інформаційної системи були обрані мови програмування C#, HTML, середовище розробки Microsoft Visual Studio, СУБД MSSQL.

Практичне значення роботи полягає у створенні програмного забезпечення в рамках веб додатку, що може бути використане для полегшення планування та відстеження завдань. Воно є корисним не лише для користувачів певної галузі, оскільки не має професійного спрямування.

В Економічному розділі визначено трудомісткість розробки програмного забезпечення (765,56 людино-годин), підраховані витрати на створення програмного забезпечення (88 039,4 грн) і очікувана тривалість розробки приблизно 5,8 місяців для одного розробника при 30-годинному робочому тижні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Exploring the role of progressive web applications in modern web development: https://www.semanticscholar.org/paper/6c1ec8426805b9099b3053e4e649bbe656d41f75Magomadov/6c1ec8426805b9099b3053e4e649bbe656d41f75

2. Overview of ASP.NET Core – URL:https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

3. DEFINITION web application (web app) : https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

4. What Is A Web Application? https://aws.amazon.com/what-is/web-application/

5. What is Cloud Computing – URL: https://www.ibm.com/cloud/learn/cloud-computing

6. Multitenant SaaS model of cloud computing: Issues and solutions – URL: https://www.semanticscholar.org/paper/Multitenant-SaaS-model-of-cloud-computing%3A-Issues-Saraswathi-Bhuvaneswari/935655e1c1b870cdeafbe424ce6391bee94317ee

7. An empirical examination of customer ’ s continuance intention of SaaS based on expectation confirmation model – URL: https://www.semanticscholar.org/paper/An-empirical-examination-of-customer-’-s-intention/459ba1fb678bd2c4511ebd48b6f6f76e3b42501c

8. Programming ASP.NET MVC 4 - Dveloping Real-world Web Applications with ASP.NET MVC – URL: https://www.semanticscholar.org/paper/Programming-ASP.NET-MVC-4-Dveloping-Real-world-Web-Chadwick-Snyder/ae7a57238c22a4e7a616ed29d382b7e20251f1b7

9. Entity Framework 4.0 Recipes: A Problem-Solution Approach – URL:

https://www.semanticscholar.org/paper/Entity-Framework-4.0-Recipes%3A-A-Problem-Solution-Tenny-Hirani/4661bc2baddb9099591dd926e0251234b0e3f90e

10. Work UA середня заробітна плата в IT – URL: https://www.work.ua/salary-dnipro-it/

11. The Security Question and Security Disposed of MSSQL Server's Database Server – URL: https://www.semanticscholar.org/paper/The-Security-Question-and-Security-Disposed-of-Wen-tao/f5c7c82a976eadad7ddce6f7103c0846f79de311

12. SQL (structured query language) information acquiring and auditing system based on MSSQL (Microsoft SQL server) database – URL: https://www.semanticscholar.org/paper/SQL-(structured-query-language)-information-and-on-吕兵-李曙□ /cf1d975329b22320f0fe27feeddf0981fca93be5

13. Multi-relational data mining in Microsoft SQL Server 2005 – URL: https://www.semanticscholar.org/paper/Multi-relational-data-mining-in-Microsoft-SQL-2005-Curotto-Ebecken/26caabd18bebaf4186afbf99f3386b7258f3af99

14. AWS Prescriptive Guidance Migrating Microsoft SQL Server databases to the AWS Cloud – URL: https://www.semanticscholar.org/paper/AWS-Prescriptive-Guidance-Migrating-Microsoft-SQL/683760d4ba2fcdbda24eacb0392a17a8fc3f8beb

15. Understanding HTML with Large Language Models – URL: https://arxiv.org/abs/2210.03945

16. Programming ASP.NET MVC 4 - Dveloping Real-world Web Applications with ASP.NET MVC – URL: https://www.semanticscholar.org/paper/Programming-ASP.NET-MVC-4-Dveloping-Real-world-Web-Chadwick-Snyder/ae7a57238c22a4e7a616ed29d382b7e20251f1b7

17. Фаулер М. UML. Основи: короткий посібник зі стандартної мови об'єктного моделювання, 2018 – 51 с.

18. Біттнер К., Спенс І., Use Case моделювання, 2003 – 30 с.

19. Офіційний сайт Microsoft Visual Studio –URL: https://visualstudio.microsoft.com

20. What is SQL Server Management Studio (SSMS)? – URL: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16

21. GitHub documentation – URL: https://docs.github.com/en

ДОДАТОК А

КОД ПРОГРАМИ

TodoList.sln

```
<Project Sdk="Microsoft.NET.Sdk.Web">

 <PropertyGroup>
  <TargetFramework>net6.0</TargetFramework>
  <Nullable>enable</Nullable>
  <ImplicitUsings>enable</ImplicitUsings>
 </PropertyGroup>

 <ItemGroup>
  <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="6.0.15" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="7.0.5" />
  <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="6.0.13" />
 </ItemGroup>

 <ItemGroup>
  <ProjectReference Include="..\TodoList.DAL\TodoList.DAL.csproj" />
 </ItemGroup>

</Project>
```

Account Controller.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using TodoList.DAL.Entities;
using TodoList.DAL.Repositories;

namespace TodoList.Controllers
{
	[Authorize]
	public class AccountController : Controller
	{
		IConfiguration _configuration;
		private readonly GoalRepository _goalRepository;

		public AccountController(IConfiguration configuration, GoalRepository goalRepository)
		{
			_configuration = configuration;
```

```
            _goalRepository = goalRepository;
        }

        #region Login
        [AllowAnonymous]
        [HttpGet]
        public IActionResult Login()
        {
            return View();
        }
        [AllowAnonymous]
        [HttpPost]
        public async Task<IActionResult> Login(User user)
        {
            try
            {
                if (string.IsNullOrEmpty(user.Email) ||
                string.IsNullOrEmpty(user.Password))
                    return BadRequest("Username and/or Password not specified");
                if (await _goalRepository.LoginAsync(user))
                {
                    var secretKey = new SymmetricSecurityKey
                    (Encoding.UTF8.GetBytes(_configuration["Jwt:Secret"]));
                    var signinCredentials = new SigningCredentials
                  (secretKey, SecurityAlgorithms.HmacSha256);
                    var jwtSecurityToken = new JwtSecurityToken(
                        issuer: _configuration["Jwt:ValidIssuer"],
                        audience: _configuration["Jwt:ValidAudience"],
                        claims: new List<Claim>(),
                        expires:
DateTime.Now.AddMinutes(Convert.ToDouble(_configuration["Jwt:ExpiryInMinutes"])),
                        signingCredentials: signinCredentials
                    );
                    var token = new JwtSecurityTokenHandler().
                    WriteToken(jwtSecurityToken);
                    Response.Cookies.Append("X-Access-Token", token, new CookieOptions() {
HttpOnly = true, SameSite = SameSiteMode.Strict });
                    Response.Cookies.Append("UserId", user.Id.ToString());
                    return RedirectToAction("Index", "Home");
                }
            }
            catch
            {
                return BadRequest
                ("An error occurred in generating the token");
```

```
                }
                return Unauthorized();
            }
            #endregion
            #region Register
            [AllowAnonymous]
            [HttpGet]
            public IActionResult Register()
            {
                return View();
            }
            [AllowAnonymous]
            [HttpPost]
            public async Task<IActionResult> Register(User user)
            {
                if (!ModelState.IsValid)
                {
                    return View(user);
                }
                if (string.IsNullOrEmpty(user.Email) ||
                string.IsNullOrEmpty(user.Name) ||
                string.IsNullOrEmpty(user.Password))
                    return BadRequest("Заповніть всі поля");
                if (await _goalRepository.RegisterAsync(user))
                {
                    return RedirectToAction("Login");
                }
                else
                {
                    user.Email = "Такий email вже існує";
                    return View(user);
                }

            }
            #endregion
            public async Task<IActionResult> Logout()
            {
                Response.Cookies.Delete("X-Access-Token");

                return RedirectToAction("Index", "Home");
            }
        }
}
```

GoalController.cs

```
using Microsoft.AspNetCore.Authorization;
```

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using TodoList.DAL.Entities;
using TodoList.DAL.Repositories;

namespace TodoList.Controllers
{
    [Authorize]
    public class GoalController : Controller
    {
        private readonly ILogger<GoalController> _logger;
        private readonly GoalRepository _goalRepository;

        public GoalController(ILogger<GoalController> logger, GoalRepository goalRepository)
        {
            _logger = logger;
            _goalRepository = goalRepository;
        }

        public async Task<ActionResult> Index(int? id)
        {
            if (id != null)
            {
                var goal = await _goalRepository.GetGoalIncludeAsync(id.Value);
                if (goal != null)
                {
                    return View(goal);
                }
            }

            return NotFound();
        }

        #region Creation
        public IActionResult Create()
        {
            return View();
        }


        [HttpPost]
        public async Task<IActionResult> Create(Goal goal)
        {
            var userid = Convert.ToInt32(Request.Cookies["UserId"]);
            goal.UserId = userid;
```

```
    if (!ModelState.IsValid)
    {
        return View(goal);
    }

    await _goalRepository.CreateGoalAsync(goal);

    return RedirectToAction("Index", "Home");
}
#endregion


[HttpPost]
public async Task<IActionResult> Delete(int? id)
{
    if (id != null)
    {
        await _goalRepository.DeleteGoalAsync(id.Value);

        return RedirectToAction("Index", "Home");
    }

    return NotFound();
}


#region Edition
[HttpGet]
[Authorize]
public async Task<IActionResult> Edit(int? id)
{

    if (id != null)
    {
        var goal = await _goalRepository.GetGoalAsync(id.Value);
        if (goal != null)
        {
            return View(goal);
        }
    }

    return NotFound();
}
```

```
        [HttpPost]
        public async Task<IActionResult> Edit(Goal goal)
        {
            var userid = Convert.ToInt32(Request.Cookies["UserId"]);
            goal.UserId = userid;

            if (!ModelState.IsValid)
            {
                return View(goal);
            }

            await _goalRepository.EditGoalAsync(goal);

            return RedirectToAction("Index", "home");
        }
        #endregion

        [HttpPost]
        public async Task<IActionResult> SwitchStatus(int? id)
        {
            if (id != null)
            {
                await _goalRepository.SwitchGoalStatusAsync(id.Value);

                return RedirectToAction("Index", "Home");
            }

            return NotFound();
        }
    }
}
```

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Mvc.Rendering;
using TodoList.ViewModels;
using Microsoft.AspNetCore.Authorization;
using TodoList.DAL.Repositories;
using TodoList.DAL.Entities;
using Microsoft.Data.SqlClient;

namespace TodoList.Controllers
{
    [Authorize]
```

```
public class HomeController : Controller
{
   private readonly ILogger<HomeController> _logger;
   private readonly GoalRepository _goalRepository;


   public HomeController(ILogger<HomeController> logger, GoalRepository goalRepository)
   {
      _logger = logger;
      _goalRepository = goalRepository;
      _goalRepository.InitAsync();
   }

   [AllowAnonymous]
   [HttpGet]
   public async Task<ActionResult> Index(int? type, string? name, SortState sortOrder = SortState.NameAsc)
   {
      if (!User.Identity.IsAuthenticated)
      {
         return View();
      }
      else
      {
                                    var userid = Convert.ToInt32(Request.Cookies["UserId"]);

                                    var goal = await _goalRepository.GetGoalsAsync(type, name, userid, sortOrder);

                                    var types = await _goalRepository.GetTypesAsync();
                                    types.Insert(0, new DAL.Entities.Type { Name = "Все", Id = 0 });

                                    GoalListViewModel viewModel = new GoalListViewModel
                                    {
                                          Goals = goal,
                                          Types = new SelectList(types, "Id", "Name", type),
                                          Name = name,
                                          SortViewModel = new SortViewModel(sortOrder)
                                    };

                                    return View(viewModel);
                              }
   }

   public async Task<ActionResult> Filtration(int? type, string? name)
   {
      var userid = Convert.ToInt32(Request.Cookies["UserId"]);
```

```
            var goal = await _goalRepository.GetGoalsAsync(type, name, userid);

            var types = await _goalRepository.GetTypesAsync();
            types.Insert(0, new DAL.Entities.Type { Name = "Все", Id = 0 });

            GoalListViewModel viewModel = new GoalListViewModel
            {
                Goals = goal,
                Types = new SelectList(types, "Id", "Name", type),
                Name = name,
            };

            return View(viewModel);
        }

    }
}
```

SortHeaderTagHelper.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Mvc.ViewFeatures;
using Microsoft.AspNetCore.Razor.TagHelpers;
using TodoList.DAL.Entities;

namespace TodoList.TagHelpers
{
    public class SortHeaderTagHelper : TagHelper
    {
        public SortState Property { get; set; }
        public SortState Current { get; set; }
        public string? Action { get; set; }
        public bool Up { get; set; }

        [ViewContext]
        [HtmlAttributeNotBound]
        public ViewContext ViewContext { get; set; } = null!;

        IUrlHelperFactory urlHelperFactory;
        public SortHeaderTagHelper(IUrlHelperFactory helperFactory)
        {
            urlHelperFactory = helperFactory;
        }
```

```
        public override void Process(TagHelperContext context, TagHelperOutput output)
        {
            IUrlHelper urlHelper = urlHelperFactory.GetUrlHelper(ViewContext);
            output.TagName = "a";
            string? url = urlHelper.Action(Action, new { sortOrder = Property });
            output.Attributes.SetAttribute("href", url);

            if (Current == Property)
            {
                TagBuilder tag = new TagBuilder("i");
                tag.AddCssClass("glyphicon");

                if (Up == true)
                    tag.AddCssClass("glyphicon-chevron-up");
                else
                    tag.AddCssClass("glyphicon-chevron-down");

                output.PreContent.AppendHtml(tag);
            }
        }
    }
}
```

AccountViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace TodoList.ViewModels
{
    public class AccountModel
    {
        [Required(ErrorMessage = "Не вказано Ім'я")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Не вказаний пароль")]
        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
}
```

GoalListViewModel.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;
using TodoList.DAL.Entities;

namespace TodoList.ViewModels
{
    public class GoalListViewModel
    {
```

```
        public IEnumerable<Goal> Goals { get; set; } = new List<Goal>();
        public SelectList Types { get; set; } = new SelectList(new List<DAL.Entities.Type>(), "Id", "Name");
        public string? Name { get; set; }
        public SortViewModel SortViewModel { get; set; } = new SortViewModel(SortState.NameAsc);
    }
}
```

IndexViewModel.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;
using TodoList.DAL.Entities;

namespace TodoList.ViewModels
{
    public class IndexViewModel
    {
        public IEnumerable<Goal> Products { get; set; } = new List<Goal>();
        public SelectList Types { get; set; } = new SelectList(new List<DAL.Entities.Type>(), "Id", "Name");
        public string? Name { get; set; }
        public SortViewModel SortViewModel { get; set; } = new SortViewModel(SortState.NameAsc);
    }
}
```

SortViewModel.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;
using TodoList.DAL.Entities;

namespace TodoList.ViewModels
{
    public class SortViewModel
    {
        public SortState NameSort { get; set; }
        public SortState TimeSort { get; set; }
        public SortState TypeSort { get; set; }
        public SortState StatusSort { get; set; }
        public SortState Current { get; set; }
        public bool Up { get; set; }

        public SortViewModel(SortState sortOrder)
        {

            NameSort = SortState.NameAsc;
            TypeSort = SortState.TypeAsc;
            TimeSort = SortState.TimeAsc;
            StatusSort = SortState.StatusAsc;
            Up = true;

            if (sortOrder == SortState.NameDesc
```

```
            || sortOrder == SortState.TypeDesc
            || sortOrder == SortState.TimeDesc
            || sortOrder == SortState.StatusDesc)
        {
            Up = false;
        }

        switch (sortOrder)
        {
            case SortState.NameDesc:
                Current = NameSort = SortState.NameAsc;
                break;
            case SortState.TypeAsc:
                Current = TypeSort = SortState.TypeDesc;
                break;
            case SortState.TypeDesc:
                Current = TypeSort = SortState.TypeAsc;
                break;
            case SortState.TimeAsc:
                Current = TimeSort = SortState.TimeDesc;
                break;
            case SortState.TimeDesc:
                Current = TimeSort = SortState.TimeAsc;
                break;
            case SortState.StatusAsc:
                Current= StatusSort = SortState.StatusDesc;
                break;
            case SortState.StatusDesc:
                Current= StatusSort = SortState.StatusAsc;
                break;
            default:
                Current = NameSort = SortState.NameDesc;
                break;
        }
    }
  }
}
```

Login.cshtml

```
@model TodoList.DAL.Entities.User

<h2>Вхід на сайт</h2>

<form asp-action="Login" asp-controller="Account" asp-anti-forgery="true">
  <div class="validation" asp-validation-summary="ModelOnly"></div>
  <div>
```

```
      <div class="form-group">
        <label asp-for="Email">Введіть email</label>
        <input type="text" asp-for="Email" />
        <span asp-validation-for="Email" />
      </div>
      <div class="form-group">
        <label asp-for="Password">Введіть пароль</label>
        <input type="password" asp-for="Password" />
        <span asp-validation-for="Password" />
      </div>
      <div class="form-group">
        <input type="submit" value="Увійти" class="btn btn-outline-dark" />
      </div>
    </div>
</form>
<form asp-controller="Account" asp-action="Register">
      <input type="submit" value="Реєстрація" />
</form>
```

Register.cshtml

```
@model TodoList.DAL.Entities.User

<h2>Реєстрація на сайт</h2>

<form asp-action="Register" asp-controller="Account" asp-anti-forgery="true">
    <div class="validation" asp-validation-summary="ModelOnly"></div>
    <div>
      <div class="form-group">
        <label asp-for="Email">Введіть email</label>
        <input type="text" asp-for="Email" />
        <span asp-validation-for="Email" />
      </div>
      <div class="form-group">
        <label asp-for="Name">Введіть юзернейм</label>
        <input type="text" asp-for="Name" />
        <span asp-validation-for="Name" />
      </div>
      <div class="form-group">
        <label asp-for="Password">Введіть пароль</label>
        <input asp-for="Password" />
        <span asp-validation-for="Password" />
      </div>
      <div class="form-group">
        <input type="submit" value="Увійти" class="btn btn-outline-dark" />
      </div>
    </div>
```

```
</form>
Create.cshtml
@model TodoList.DAL.Entities.Goal

<h2>Нове завдання</h2>
<form asp-action="create" asp-controller="goal">
    <p>
        <label asp-for="Name">Назва</label><br />
        <input type="text" asp-for="Name" />
    </p>
    <p>
        <label asp-for="Time">Термін виконання</label><br />
        <input type="datetime-local" asp-for="Time" />
    </p>
    <p>
        <select asp-for="TypeId">
            <option value="">Оберіть категорію</option>
            <option value="1">Без категорії</option>
            <option value="2">Роботи по дому</option>
            <option value="3">Спорт</option>
            <option value="4">Робота</option>
        </select>
    </p>
    <p>
        <label asp-for="Description">Опис</label><br />
        <input type="text" asp-for="Description" />
    </p>
    <p>
        <input type="submit" value="Прийняти" />
    </p>
</form>
Edit.cshtml
@model TodoList.DAL.Entities.Goal

<h2>Редагування завдання</h2>
<form asp-controller="Goal" asp-action="Edit" asp-route-id="@Model.Id">
    <p>
        <label asp-for="Name">Назва</label><br />
        <input type="text" asp-for="Name" />
    </p>
    <p>
        <label asp-for="Time">Термін виконання</label><br />
        <input type="datetime-local" asp-for="Time" />
    </p>
    <p>
```

```
        <select asp-for="TypeId">
            <option value="">Оберіть категорію</option>
            <option value="1">Без категорії</option>
            <option value="2">Роботи по дому</option>
            <option value="3">Спорт</option>
            <option value="4">Робота</option>
        </select>
    </p>
    <p>
        <label asp-for="Description">Опис</label><br />
        <input type="text" asp-for="Description" />
    </p>
    <p>
        <input type="submit" value="Зберегти" />
    </p>
</form>
```

Index.cshtml

```
@model TodoList.DAL.Entities.Goal

@{
    ViewData["Title"] = "Goal";
}

<h1>Завдання @Model.Name</h1>
Категорія: @Model.Type?.Name <br />
Описание завдання: @Model.Description<br />
Термін виконнання: @Model.Time<br />
Статус: @if(Model.Status == true)
{
        <text>Виконано</text>
}else{
    <text>Ще не виконано</text>
}
```

Filtration.cshtml

```
@{
    ViewData["Title"] = "Sorting";
}
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@model GoalListViewModel
@using TodoList.ViewModels

<h2>Ваші завдання</h2>

<form method="get">
    <div>
```

```
        <label>Назва </label>
        <input asp-for="Name" />

        <label>Тип </label>
        <select name="Type" asp-items="Model.Types"></select>

        <input type="submit" value="Фільтр" />
    </div>
</form>

<table>
    <tr>
        <th>Назва</th>
        <th>Тип</th>
        <th>Час</th>
        <th>Статус</th>
    </tr>

    @foreach (var item in Model.Goals)
    {
        <tr>
            <td>
                <a asp-action="index" asp-controller="product" asp-route-id="@item.Id">@item.Name</a>
            </td>
            <td>@item.Type?.Name</td>
            @if(item.Time > DateTime.Now){
                TimeSpan ActualTime = (DateTime.Now - item.Time).Duration();
                string OutputTime = ActualTime.Days + " днів " + ActualTime.ToString(@"hh\:mm");
                <td>Залишилося @OutputTime</td>
            }
            else if(item.Time.Year < 2023) {<td></td>}
            else{<td>Прострочено</td>}
            @if(item.Status == true){
                <td> Виконано</td>
            }else{
                <td>Не виконано</td>
            }

        </tr>
    }
</table>
```

Index.cshtml

```
@{
    ViewData["Title"] = "Index";
```

```
}
@if(!User.Identity.IsAuthenticated)
{
   <h1>Unauthorized</h1>
}else
{
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@model GoalListViewModel
@using TodoList.ViewModels
@addTagHelper *, TodoList


<h2>Список завдань</h2>
<p><a asp-action="Create" asp-controller="Goal">Додати завдання</a></p>
<table class="table">
    <tr>
      <th>
        <sort-header action="Index"  up="@Model.SortViewModel.Up"
              current="@Model.SortViewModel.Current" property="@Model.SortViewModel.NameSort">
           Ціль
        </sort-header>
      </th>
      <th>
        <sort-header action="Index" up="@Model.SortViewModel.Up"
              current="@Model.SortViewModel.Current" property="@Model.SortViewModel.TimeSort">
           Термін виконання
        </sort-header>
      </th>
      <th>
        <sort-header action="Index" up="@Model.SortViewModel.Up"
              current="@Model.SortViewModel.Current" property="@Model.SortViewModel.TypeSort">
           Категорія
        </sort-header>
      </th>
      <th>
        <sort-header action="Index" up="@Model.SortViewModel.Up"
              current="@Model.SortViewModel.Current" property="@Model.SortViewModel.StatusSort">
           Статус
        </sort-header>
      </th>
      <th>
      </th>
      <th>
      </th>
   </tr>
```

```
    @foreach (var item in Model.Goals)
    {
        <tr>
            <td>
                <a asp-action="Index" asp-controller="Goal" asp-route-id="@item.Id">@item.Name</a>
            </td>
            @if(item.Time > DateTime.Now){
                TimeSpan ActualTime = (DateTime.Now - item.Time).Duration();
                string OutputTime = ActualTime.Days + " днів " + ActualTime.ToString(@"hh\:mm");
                <td>Залишилося @OutputTime</td>
            }
            else if(item.Time.Year < 2023) {<td></td>}
            else{<td>Прострочено</td>}
            <td>
                @item.Type.Name
            </td>
            <td>
                @if(item.Status == false)
                {
                <form asp-controller="Goal" asp-action="SwitchStatus" asp-route-id="@item.Id">
                    <input type="submit" value="Ще не виконано" />
                </form>
                }
                else
                {
                <form asp-controller="Goal" asp-action="SwitchStatus" asp-route-id="@item.Id">
                    <input type="submit" value="Готово" />
                </form>
                }
            </td>
            <td>
                <form asp-controller="Goal" asp-action="Edit" method="get"  asp-route-id="@item.Id">
                    <input type="submit" value="Редагувати" />
                </form>
            </td>
            <td>
                <form asp-controller="Goal" asp-action="Delete" method="post"  asp-route-id="@item.Id">
                    <input type="submit" value="Видалити" />
                </form>
            </td>
        </tr>
    }
</table>
}
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - Store</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="stylesheet" href="~/Store.styles.css" asp-append-version="true" />
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-
3">
      <div class="container-fluid">
    @if (User.Identity.IsAuthenticated)
    {
      <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">MyTodoList</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
            aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
                                </button>
      <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
        <ul class="navbar-nav flex-grow-1">
          <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">TaskList</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Filtration">Filtration</a>
          </li>
          <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Account" asp-action="Logout">Logout</a>
          </li>
        </ul>
      </div>
    }
    else
    {
      <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">MyTodoList</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
aria-controls="navbarSupportedContent"
            aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
```

```
                                    </button>
            <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
                <ul class="navbar-nav flex-grow-1">
                    <li class="nav-item">
                        <a class="nav-link text-dark" asp-area="" asp-controller="Account" asp-action="Login">Login</a>
                    </li>
                </ul>
            </div>
        }
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2022 - TodoList made by Кутнич Томаш
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

appsettings.json

```
{
  "AllowedHosts": "*",
  "Jwt": {
    "ValidAudience": "ABCXYZ",
    "ValidIssuer": "http://localhost:7223",
    "Secret": "thisisasecretkey@123",
    "ExpiryInMinutes": 180
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}
```

Program.cs

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using TodoList.DAL.Contexts;
using TodoList.DAL.Repositories;
using Task = System.Threading.Tasks.Task;

internal class Program
{
   private static void Main(string[] args)
   {
      var builder = WebApplication.CreateBuilder(args);

      // Add services to the container.
      builder.Services.AddControllersWithViews();



      string connection = "Server = (localdb)\\mssqllocaldb;Database = TodoListstoredb;Trusted_Connection=true";

      builder.Services.AddScoped<GoalRepository>();
;
      builder.Services.AddDbContext<GoalContext>(options => options.UseSqlServer(connection));

      builder.Services.AddAuthorization();
      builder.Services.AddAuthentication(options =>
      {
         options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
         options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
         options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
      })
      .AddJwtBearer(options =>
      {
         options.SaveToken = true;
         options.RequireHttpsMetadata = false;
         options.TokenValidationParameters = new TokenValidationParameters()
         {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateIssuerSigningKey = true,
            ValidateLifetime = true,
            ValidIssuer = builder.Configuration["Jwt:ValidIssuer"],
            ValidAudience = builder.Configuration["Jwt:ValidAudience"],
```

```
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Secret"])),
            ClockSkew = TimeSpan.Zero
        };
        options.Events = new JwtBearerEvents
        {
            OnMessageReceived = context =>
            {

                if (context.Request.Cookies.ContainsKey("X-Access-Token"))
                {
                    context.Token = context.Request.Cookies["X-Access-Token"];
                }

                return Task.CompletedTask;
            }
        };
    });

    var app = builder.Build();


    // Configure the HTTP request pipeline.
    if (!app.Environment.IsDevelopment())
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
        app.UseHsts();
    }

    app.UseAuthentication();

    app.UseRouting();

    app.UseAuthorization();

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    app.Run();
```

```
    }
}
```

TodoList.DAL

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Http.Abstractions" Version="2.2.0" />
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="7.0.5" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="7.0.5" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="7.0.5">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
  </ItemGroup>

</Project>
```

GoalContext.cs

```
using Microsoft.EntityFrameworkCore;
using TodoList.DAL.Entities;

namespace TodoList.DAL.Contexts
{
    public class GoalContext : DbContext
    {
        public DbSet<Goal> Goals { get; set; }
        public DbSet<Entities.Type> Types { get; set; }
        public DbSet<User> Users { get; set; }

        public GoalContext(DbContextOptions<GoalContext> options)
            : base(options)
        {
            Database.EnsureCreated();
        }
    }
}
```

Goal.cs

```
using System.ComponentModel.DataAnnotations;

namespace TodoList.DAL.Entities
```

```
{
    public class Goal
    {
        public int Id { get; set; }

        [Required(ErrorMessage ="Обов'язкове поле")]
        public string? Name { get; set; }
        public string? Description { get; set; }
        public Boolean Status { get; set; }
        [DataType(DataType.DateTime)]
        public DateTime Time { get; set; }
        public int? TypeId { get; set; }
        public Type? Type { get; set; }
        public int UserId { get; set; }
    }
}
```

Type.cs

```
namespace TodoList.DAL.Entities
{
    public class Type
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public List<Goal> Goals { get; set; } = new();
    }
}
```

User.cs

```
using System.ComponentModel.DataAnnotations;

namespace TodoList.DAL.Entities
{
    public class User
    {
        public int Id { get; set; }

        [Required(ErrorMessage = "Обов'язкове поле")]
        public string Email { get; set; }

        [Required(ErrorMessage ="Обов'язкове поле")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Обов'язкове поле")]
        public string Password { get; set; }
        public List<Goal> Goals { get; set; } = new();
    }
```

```
}
GoalRepository.cs
using Microsoft.AspNetCore.Http;
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TodoList.DAL.Contexts;
using TodoList.DAL.Entities;

namespace TodoList.DAL.Repositories
{
   public class GoalRepository
   {
      private readonly GoalContext _goalContext;

      public GoalRepository(GoalContext goalContext)
      {
         _goalContext = goalContext;
      }
      #region Goal
      public async Task<List<Goal>> GetGoalsAsync()
      {
         return await _goalContext.Goals.Include<Goal, Entities.Type>(pr => pr.Type).ToListAsync();
      }

      public async Task<List<Goal>> GetGoalsAsync(int? type, string? name, int userid)
      {
         IQueryable<Goal> goal = _goalContext.Goals.Include<Goal, Entities.Type>(g => g.Type).Where(g => g.UserId ==
userid);

         if (type != null && type != 0)
         {
            goal = goal.Where(g => g.TypeId == type);
         }
         if (!string.IsNullOrEmpty(name))
         {
            goal = goal.Where(g => g.Name!.Contains(name));
         }

         return await goal.ToListAsync();
      }
```

```
    public async Task<List<Goal>> GetGoalsAsync(int? type, string? name, int userid, SortState sortOrder =
SortState.NameAsc)
    {
        IQueryable<Goal> goal = _goalContext.Goals.Include<Goal, Entities.Type>(g => g.Type).Where(g => g.UserId ==
userid);

        if (type != null && type != 0)
        {
            goal = goal.Where(g => g.TypeId == type);
        }
        if (!string.IsNullOrEmpty(name))
        {
            goal = goal.Where(g => g.Name!.Contains(name));
        }

        goal = sortOrder switch
        {
            SortState.NameDesc => goal.OrderByDescending(s => s.Name),
            SortState.TimeAsc => goal.OrderBy(s => s.Time),
            SortState.TimeDesc => goal.OrderByDescending(s => s.Time),
            SortState.TypeAsc => goal.OrderBy(s => s.Type!.Name),
            SortState.TypeDesc => goal.OrderByDescending(s => s.Type!.Name),
            SortState.StatusAsc => goal.OrderBy(s => s.Status),
            SortState.StatusDesc => goal.OrderByDescending(s => s.Status),
            _ => goal.OrderBy(s => s.Name),
        };

        return await goal.ToListAsync();
    }

    public async Task<Goal> GetGoalIncludeAsync(int id)
    {
        return await _goalContext.Goals.Include<Goal, Entities.Type>(pr => pr.Type).FirstOrDefaultAsync(p => p.Id ==
id);
    }

    public async Task<Goal> GetGoalAsync(int id)
    {
        return await _goalContext.Goals.FirstOrDefaultAsync(p => p.Id == id);
    }

    public async Task CreateGoalAsync(Goal goal)
    {
        _goalContext.Goals.Add(goal);
```

```
        await _goalContext.SaveChangesAsync();
    }

    public async Task EditGoalAsync(Goal goal)
    {
        _goalContext.Goals.Update(goal);
        await _goalContext.SaveChangesAsync();
    }

    public async Task DeleteGoalAsync(int id)
    {
        Goal goal = new Goal { Id = id };
        _goalContext.Entry(goal).State = EntityState.Deleted;
        await _goalContext.SaveChangesAsync();
    }

    public async Task SwitchGoalStatusAsync(int id)
    {
        var goal = await GetGoalAsync(id);
        goal.Status = !goal.Status;
        await EditGoalAsync(goal);
    }
    #endregion

    public async Task<List<Entities.Type>> GetTypesAsync()
    {
        return await _goalContext.Types.ToListAsync();
    }

    #region User
    public async Task<bool> LoginAsync(User user)
    {
        var result = await _goalContext.Users.SingleOrDefaultAsync(u => u.Email == user.Email && u.Password ==
user.Password);
        user.Id = result.Id;
        return result != null;
    }

    public async Task<bool> RegisterAsync(User user)
    {
        var result = await _goalContext.Users.SingleOrDefaultAsync(u => u.Email == user.Email);
        if (result == null)
        {
            Entities.Type not_defined = new Entities.Type { Name = "Без категорії" };
            Goal initgoal = new Goal { Name = "Ознайомитися з платформою", Description = "Ознайомитися з
```

```
платформою, навчитися користуватися всіма можливостями", Type = not_defined, Status = false, UserId = user.Id };
            User newbe = new User { Email = user.Email, Name = user.Name, Password = user.Password};
            newbe.Goals.Add(initgoal);
            _goalContext.Users.Add(newbe);
            _goalContext.SaveChanges();
            return true;
        }
        return false;
    }
    #endregion

    public async Task InitAsync()
    {
        if (!_goalContext.Types.Any())
        {
            Entities.Type not_defined = new Entities.Type { Name = "Без категорії" };
            Entities.Type home = new Entities.Type { Name = "Роботи по дому" };
            Entities.Type sport = new Entities.Type { Name = "Спорт" };
            Entities.Type work = new Entities.Type { Name = "Робота" };

            Goal goal1 = new Goal { Name = "Ознайомитися з платформою", Description = "Ознайомитися з
платформою, навчитися користуватися всіма можливостями", Type = not_defined, Status = false, UserId = 1 };

            User tom = new User {Email = "test@gmail.com", Name = "Test", Password = "test"};
            tom.Goals.Add(goal1);

            _goalContext.Types.AddRange(not_defined, home, sport, work);
            _goalContext.Goals.Add(goal1);
            _goalContext.Users.Add(tom);
            _goalContext.SaveChanges();
        }
    }
  }
}
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Програма	
Презентація	