Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Центр Післядипломної освіти
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка рекомендувального сервісу для відстеження прогресу досягнення мети
(тема)

Виконав:
здобувач другого року навчання,
групи ІТШІп-23-1
Вадим Костюченко
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ст. викл. Філіп Бродецький
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Центр Післядипломної освіти

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____« _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Костюченку Вадиму Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка рекомендувального сервісу для відстеження прогресу досягнення мети

затверджена наказом університету від 19 травня 20 25 р. № 387Ст
2. Термін подання студентом роботи до екзаменаційної комісії 24 червня 20 25 р.
3. Вихідні дані до роботи Пошук оптимальної постанови мети та її досягнення та створення веб-системи для зручного формування та відстеження цілей, документація розробки на мові програмування C#

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Аналіз вимог
3) Опис методів реалізації проекту
4) Опис програми
5) Аналіз дослідної експлуатації та можливих застосувань

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі і постановка задачі	20.05.2025	виконано
3	Технічне завдання	23.05.2025	виконано
4	Проєктування системи	28.05.2025	виконано
5	Реалізація	05.06.2025	виконано
6	Обробка і оформлення результатів	06.06.2025	виконано
7	Оформлення графічних матеріалів	08.06.2025	виконано
8	Оформлення пояснювальної записки	09.06.2025	виконано
9	Попередній захист	10.06.2025	виконано
10	Захист перед ЕК	24.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Філіп Бродецький
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 83 с., 16 рис., 2 табл., 2 дод., 23 джерела.

C#, JAVASCRIPT, MICROSOFT SQL SERVER, MICROSOFT VISUAL STUDIO, MSC 5.

Об'єкт розробки – сервіс досягнення та ведення персональних цілей.

Мета роботи – дослідження видів цілей, пошук оптимального способу постановки цілі та її досягнення та створення веб-системи для зручного формування та відстеження цілей.

Методи дослідження – аналіз літератури, Internet-джерел та аналіз результатів роботи програмно реалізованих методів на основі тестових даних.

Методи розробки – технологія ASP.NET MVC 5, среда розробки об'єктно-орієнтованих продуктів Microsoft Visual Studio 2022, мова програмування C#. Система управління базами даних MS SQL Server 2022

В результаті виконання випускної кваліфікаційної роботи створено веб-систему для зручного формування та підтримки цілей.

ABSTRACT

Bachelor’s thesis contains: 83 pp., 16 fig., 2 tabl., 2 ann., 23 references.

ASP.NET, C#, GOALS, JAVASCRIPT, MICROSOFT SQL SERVER, MICROSOFT VISUAL STUDIO.

The object of development is a service for achieving and maintaining personal goals.

The purpose of the work is to study the types of goals, find the optimal way to set a goal and achieve it, and create a web system for convenient goal formation and tracking.

Research methods are an analysis of literature, Internet sources, and analysis of the results of software-implemented methods based on test data.

Development methods are ASP.NET MVC 5 technology, Microsoft Visual Studio 2022 object-oriented product development environment, C# programming language. MS SQL Server 2022 database management system

As a result of the final qualification work, a web system was created for convenient goal formation and support.

ЗМІСТ

ВСТУП

У сучасному світі кожна людина має будь-які цілі. Більшість людей тримають цілі в голові, які часто не відрізняються від мрій. Деякі записують свої цілі і за виконанням просто їх викреслюють зі списку. Цей спосіб кращий за перший, але він не ідеальний. Звичайно, можна скористатися звичайним органайзером, запланувати дію на певну дату і спробувати досягти певної мети до цієї дати, це варіант, але що робити, якщо ваша мета масштабніша?

Така мета займе не один, не два і навіть не три роки, і як це виглядатиме? Для таких випадків і вигадали розбиття цілей на підцілі.

У такому разі спробуємо розбити ціль на тимчасові відрізки і запишемо все це органайзер. Але якщо мета має багато підцілей, буде не просто відстежити всі її відрізки, прогрес і т. д. І в цьому випадку органайзер і блокнот все ж таки не підходять для таких завдань.

Набагато ефективніше використовувати спеціальну систему для правильного, простого формування цілей та їхнього надійного зберігання.

Ціль – невід'ємна частина життя людини. Саме ціль визначає вектор розвитку людини. Це елемент, який концентрує сили та енергію на тому, що має бути досягнуто. Але сьогодні не всім відомі правильні практики постановки цілей, і тому багато мети людей перетворюються на мрії або просто відкладаються на «завтрашній день». Чим відрізняється мрія від мети? За версією Пола Мейра [1], творця найефективнішої методології постановки цілей «SMART» відмінності полягають у:

– конкретиці;
– вимірності;
– досяжність;
– актуальності;
– обмеженості у часі.

Саме слово «smart» у перекладі української мовою і означає «розумний». Таким чином, правильна постановка мети означає, що ціль є конкретною, вимірною, досяжною, актуальною та співвідноситься з конкретним терміном.

В даний час інформаційні технології дозволяють створювати дуже зручні речі, такі як: соціальні мережі, органайзери, блоги і т. д., але не одна з цих технологій, окремо взята, не дозволить створити майданчик для ефективного управління цілями. Але якщо взяти окремі елементи з кожної, перерахованої вище технології, то цілком можна реалізувати максимально зручний майданчик для роботи з цілями.

Соціальна мережа – платформа, онлайн-сервіс чи веб-сайт, призначені для побудови, відображення та організації соціальних взаємин, візуалізацією яких є соціальні графи.

Характерними рисами соціальної мережі є:

– створення особистих профілів, де часто потрібно вказати реальні персональні дані та іншу інформацію про себе;

– надання практично повного спектра можливостей для обміну інформацією (розміщення фотографій, відеозаписів, розміщення текстових записів, організація тематичних спільнот, обмін особистими повідомленнями тощо);

– можливість задавати та підтримувати список інших користувачів, з якими має деякі відносини.

Блог – це веб-сайт, основний вміст якого записи, що регулярно додаються, містять текст, зображення або мультимедіа. Для блогів характерні невеликі записи тимчасової значущості, упорядковані у зворотному хронологічному порядку. Відмінності блогу від традиційного щоденника обумовлюються середовищем: блоги зазвичай публічні і припускають сторонніх читачів, які можуть вступити до публічної полеміки з автором.

Для блогів характерна можливість опублікування відгуків відвідувачами. Вона робить блоги середовищем мережевого спілкування, що має ряд переваг перед електронною поштою, групами новин, веб-форумами та чатами.

Органайзер – спочатку невелика книга, що містить календар, адресну книгу та блокнот, що служить для організації інформації про особисті контакти та події. З розвитком інформаційних технологій книга стала замінюватися спочатку електронними органайзерами, потім кишеньковими персональними комп'ютерами, комп'ютерними програмами та онлайн-органайзерами, що мають додаткові функції: нагадування про майбутні події, захист та синхронізацію інформації.

Організатор є засобом управління часом. Попереднє планування справ допомагає підвищити плідність будь-якої діяльності, як особистої, і професійної. Тому й людину, яка консультує організації та приватних осіб у сфері підвищення ефективності управління часом, також іноді називають «органайзер». США існує Національна асоціація професійних органайзерів (NAPO).

Веб-система – клієнт-серверна програма, в якій клієнтом виступає браузер, а сервером – веб-сервер. Логіка веб-системи розподілена між сервером та клієнтом, зберігання даних здійснюється переважно на сервері, обмін інформацією відбувається по мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-системи є міжплатформними сервісами.

Клієнтська частина реалізує інтерфейс користувача, формує запити до сервера і обробляє відповіді від нього. Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб-сторінку та відправляє її клієнту через мережу з використанням протоколу HTTP. Сама веб-програма може виступати як клієнт інших служб, наприклад, бази даних або іншої веб-системи, розташованої на іншому сервері.

Інформаційні технології (ІТ) – широкий клас дисциплін та сфер діяльності, що належать до технологій створення, збереження, управління та обробки даних, у тому числі із застосуванням обчислювальної техніки. Останнім часом під інформаційними технологіями найчастіше розуміють комп'ютерні технології. Зокрема, ІТ мають справу з використанням комп'ютерів та програмного забезпечення для створення, зберігання, обробки, обмеження до передачі та отримання інформації.

Інформаційна система (ІС) – взаємопов'язана сукупність засобів, методів та персоналу, що використовуються для збирання, зберігання, обробки та видачі інформації на користь досягнення поставленої мети. Комп'ютери, оснащені спеціалізованими програмними засобами, є технічною базою та інструментом для інформаційних систем. Тобто. сучасна інформаційна система – людино-комп'ютерна.

Метою кваліфікаційної роботи є розробка веб-системи, яка надасть користувачеві зручний формат для формування мети та її супроводу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАчі

1.1 Опис предметної галузі

Мета показує, у якому напрямі рухатися і який має бути кінцевий результат. Цілі, якщо вони чітко визначені, діють як виклик та спонукають людину до дій. Більшість людей, які досягли в житті чогось значного, мали здатність ставити перед собою ясні цілі. Коли людина ставить собі конкретну мету, його підсвідомість мобілізує всі сили, щоб його прагнення здійснилися. І починає по-новому переробляти інформацію, що міститься в підсвідомості (факти, образи, переживання). Якщо в людини є мета, то будь-яка інформація, що надходить до неї, стимулює підсвідомість, і вона починає пропонувати, часом досить раптово, нові підходи і способи вирішення проблем.

Доведено, що мета обов'язково має бути записана, так вона менш схильна до забування. Але на запису мети її досягнення не закінчується.

Мета – ідеальний чи реальний предмет свідомого чи несвідомого прагнення суб'єкта; кінцевий результат, який навмисно спрямований процес «доведення можливості до її завершення».

Планування – оптимальний розподіл ресурсів задля досягнення поставленої мети.

Досягнення мети залежить від її формулювання, і перший крок успіху справи – правильно сформовані мети.

Цілі діляться на два види: кінцева та проміжна. Відмінність кінцевої мети від проміжної у цьому, що у кінцевої мети може бути кілька проміжних цілей.

Якою ж має бути мета. Мета має бути конкретною, певною. Щоб відповідати цьому критерію під час створення мети необхідно чітко уявляти її результат.

Мета має бути вимірною. Вимірюваність мети передбачає наявність критеріїв (вимірників), які дозволили б визначити, чи досягнуто поставленої мети і в якій мірі. Якщо немає вимірювачів, дуже складно оцінити результати виконаної роботи та об'єктивно контролювати процес.

Мета має бути досяжною. Людина повинна бути впевнена, що вона володіє або найближчим часом матиме ресурси для досягнення мети. Ресурсами у житті можуть бути час, гроші, корисні знайомства, інформація, знання та навички. Без цих ресурсів шанси досягнення мети значно зменшуються.

Мета має бути актуальною. Визначення істинності мети. Чи справді виконання цього завдання дозволить досягти бажаної мети? Необхідно переконатися, що виконання цього завдання дійсно необхідне;

Мета повинна бути суворо обмежена за часом. Розбиття кінцевої мети, на проміжні, допоможе скласти план.

1.2 Аналіз існуючих конкурентів

Веб-сервіс Goalmigo.com [2] – один із найпопулярніших західних веб-сервісів для формування цілей. Його головна особливість – це створення нотаток у процесі виконання мети. Крім цього, на даному сервісі реалізовані графіки, на яких відображено прогрес користувача в певних цілях. Як показано на прикладі (рисунок 1.1), користувач хоче досягти ідеальної для нього ваги, і створив два графіки – «вага» та «спалені калорії». Він має можливість щодня переносити свій прогрес у дані графіки та у такий спосіб стежити за ним. Також даний сервіс реалізує можливість колективного виконання мети, завдяки якому однієї мети може брати участь більше однієї людини. Також у «Goalmigo» реалізована можливість прямого спілкування через особисті повідомлення та коментарі до цілей. Це може налагодити зворотний зв'язок автора цілі та його передплатників.

Недолік його полягають у відсутності створення подцелей.

Користувач повинен сам здогадатися, що в опис мети слід писати і її підцілі, та й взагалі великі цілі слід розбивати на підцілі. Також його недолік полягає у відсутності вибору локалізації.

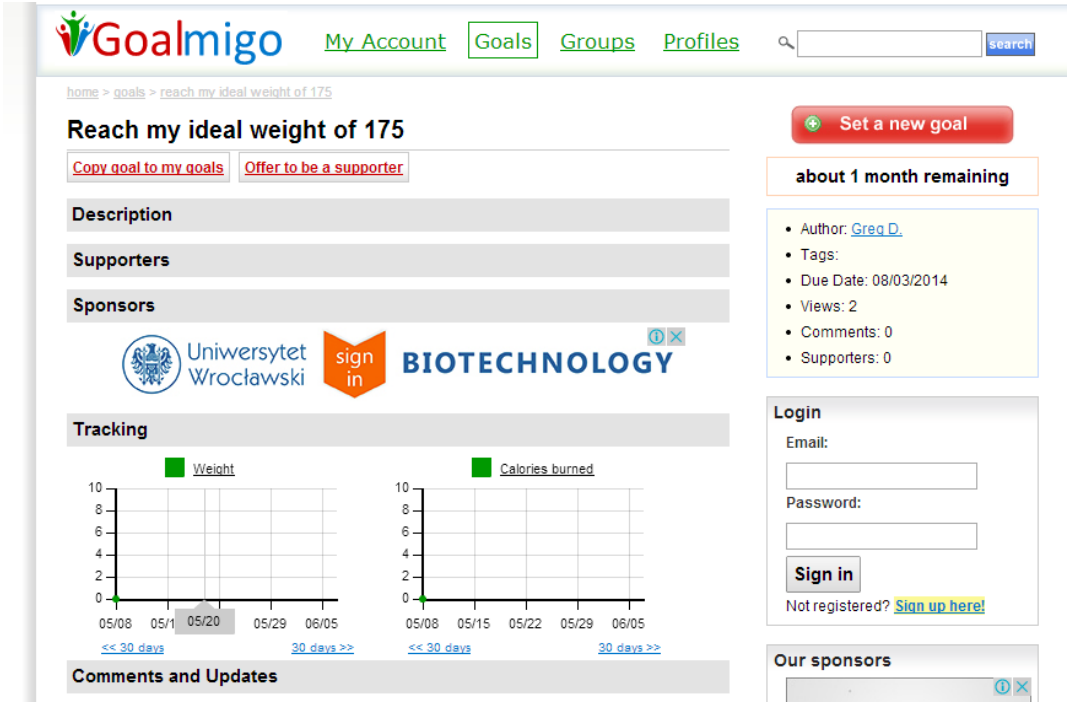

Рисунок 1.1 – Приклад поставленої мети на сайті goalmigo.com

«Motiway.com» [3] є ще одним із веб-сервісів у цій категорії, представено на рисунку 1.2.

Його особливість полягає у постановці певної мети та відзначенні в календарі коли була виконана над нею робота. Являє собою більше карти бажань з внутрішніми інструментами.

Недоліки сервісу «Motiway» полягають у:

– відсутність планування мети;

– відсутності можливості створення великих цілей із підцілями;

– відсутності локалізації;

– застарілому, непривабливому дизайні.

SmartProgress.do – багатомовний веб-сервіс розвитку цілей [4]. Має найбагатший функціонал із усіх перерахованих вище сервісів.

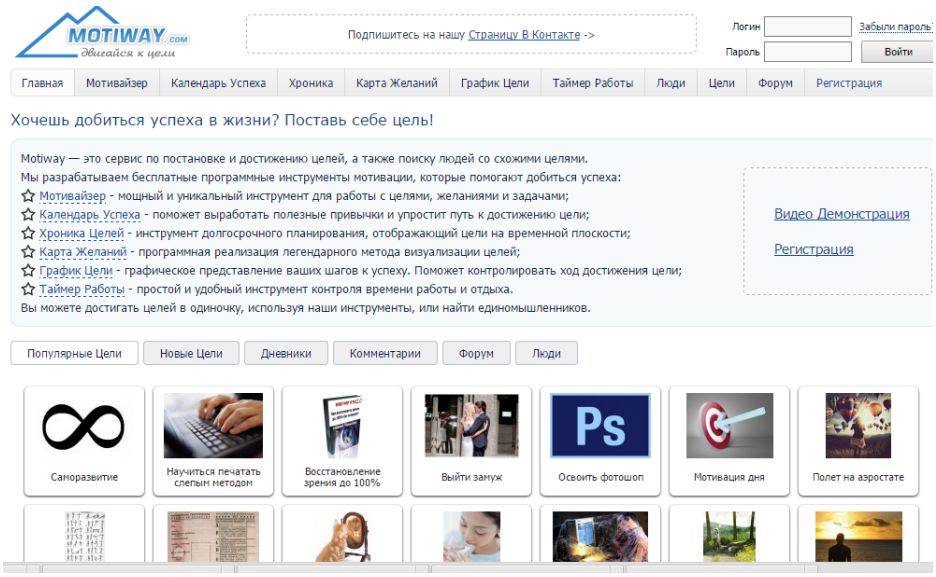

Рисунок 1.2 – Приклад відображення цілей на сайті Motiway.com

Аудиторія його становить понад 70 000 користувачів. Має зручний інтерфейс, щільну інтеграцію із соціальними мережами та безліч корисних, платних функцій таких як: «Ціна слова», платні шаблони цілей, Pro-account тощо, рисунок 1.3. 2018 року вийшов на міжнародний рівень після введення англійської локалізації.

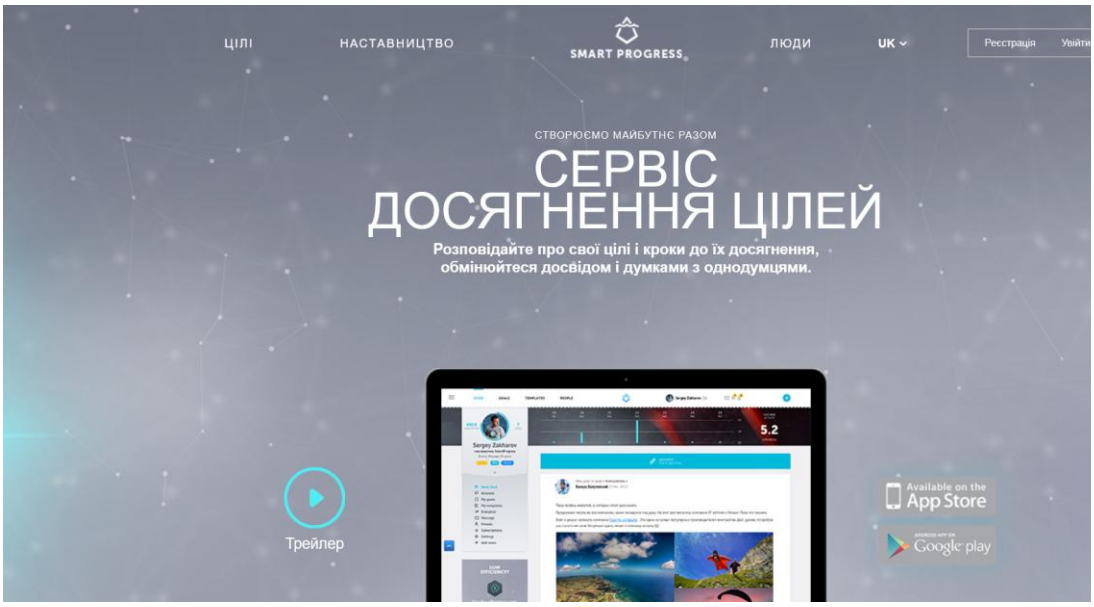

Рисунок 1.3 – Приклад відображення цілей на веб-сайті smartprogress.do

Додавання мети в [4] поділяється на три кроки.

Перший крок – назва мети та її опис. На цьому етапі користувач запроваджує загальний опис мети. Є можливість додавання критерію завершення мети, що представлено на рисунку 1.4. Зазначу, що мета в процесі створення автоматично зберігається і за будь-якого збою є можливість її відновлення. Ще відмінною особливістю є зручне редагування тексту з можливістю додавання відео чи зображень.

Другий крок – створення етапів чи посилань на підетапи. Для однієї мети можливе створення необмеженої кількості етапів. До кожного етапу, як і до основного опису мети, можна прикріплювати медіа-ресурси. Зручною відмінністю можна виділити можливість перенесення етапу із засобів Drag&Drop.

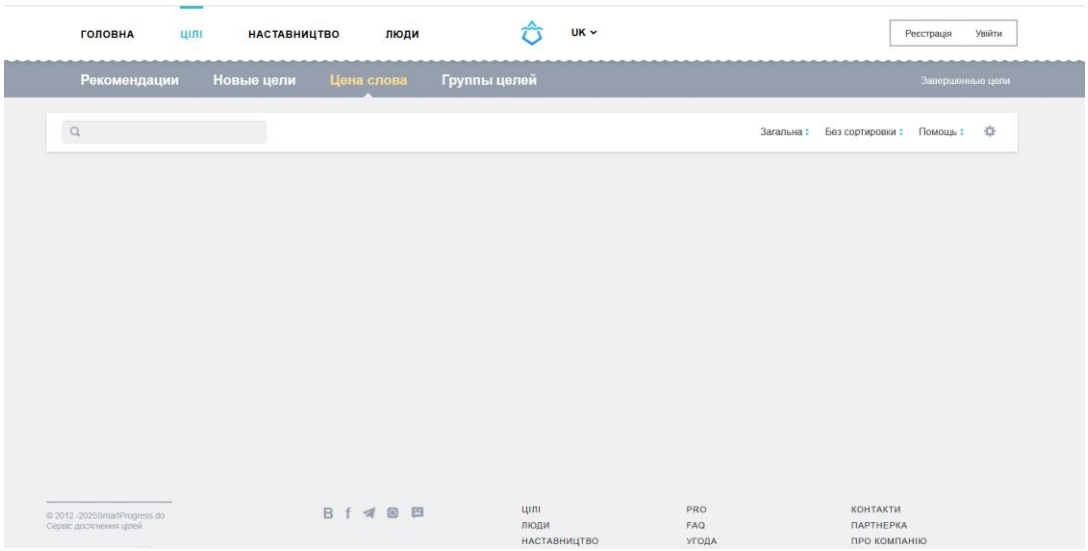

Рисунок 1.4 – Приклад створення мети на сайті smartprogress.do

Третій крок – налаштування мети. Тут користувач може вибрати тип мети, параметри приватності, часові рамки мети та «Ціну слова».

Ціна слова – один із видів мотивації. Користувач після створення мети може вибрати ціну слова. Полягає вона в тому, що користувач переводить будь-яку суму грошей на рахунок компанії SmartProgress і після закінчення

термінів мети проводиться її оцінка досягнення. Якщо користувач досяг своєї мети, гроші йому повертаються, якщо не досяг – йому надають вибір: відправити всі гроші або до дитячих будинків, або залишити їх на розвиток компанії.

1.3 Вибір засобів розробки

В основі серверної частини було обрано технологію створення веб-додатків та веб-сервісів компанії Microsoft – ASP.NET з використанням фреймворку MVC 5. ASP.NET – є складовою платформи Microsoft .NET та розвитком більш старої технології Microsoft ASP. Ця зв'язка має безліч переваг, найголовніші з них це:

– розширюваність;

– має чіткий поділ на модель, подання та контролер;

– можливість охоплення всього проекту unit-тестами;

– пропонує повний контроль над генерованим HTML-кодом;

– генерує чистий HTML-код;

– кращий поділ між UI та кодом (логікою програми та логікою подання);

– підтримує безліч різних двигунів вистави (View Engines);

– за замовчуванням використовує REST-підхід для URL'ів – що також добре для SEO;

– немає ViewState (це також може бути недоліком у певних випадках);

– простий обсяг завантажуваної сторінки – невеликий;

– проста інтеграція із фреймворками типу Jquery.

Фреймворк MVC 5 зарекомендував себе як гарне архітектурне рішення, він також зумовлений легкою адаптацією до розміщення проектів у хмарах. Крім цього платформа .NET дозволяє використовувати одну мову

програмування для створення веб-додатків, веб-служб, сервісів, мобільних додатків або десктопних програм.

До складу MVC входять такі компоненти: модель, уявлення, контролер.

Модель надає знання: дані та методи роботи з цими даними реагує на запити змінюючи свій стан. Не містить інформації про те, як ці знання можна візуалізувати.

Подання, вид відповідає за відображення інформації. Часто як уявлення виступає форма з графічними елементами.

Контролер забезпечує зв'язок між користувачем та системою: контролює введення даних користувачем та використовує модель та подання для реалізації необхідної реакції.

Важливо, що уявлення і контролер залежить від моделі, але модель залежить від подання, ні від контролера. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуального подання, а також створювати кілька різних уявлень для однієї моделі.

Завдяки підтримці фреймворку jQuery, є можливість зручного зв'язку між клієнтською та серверною стороною за допомогою технології jQueryAjax.

jQuery – бібліотека JavaScript, що фокусується на взаємодії JavaScript та HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елемента DOM, звертатися до атрибутів та вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX.

Можливості jQuery:

– перехід по дереву DOM, включаючи підтримку XPath як плагін;
– події;
– візуальні ефекти;
– AJAX-доповнення;

– JavaScript-плагіни.

Ajax – підхід до побудови інтерактивних інтерфейсів веб-додатків, що полягає у фоновому обміні даними браузера з веб-сервером. В результаті, при оновленні даних веб-сторінка не перезавантажується повністю, і веб-програми стають швидше і зручніше.

Переваги технології Ajax:

– економія трафіку: використання AJAX дозволяє значно скоротити трафік при роботі з веб-додатком завдяки тому, що замість завантаження всієї сторінки достатньо завантажити лише частину, що змінилася, або взагалі тільки отримати/передати набір даних у форматі JSON або XML, а потім змінити вміст сторінки за допомогою JavaScript;

– зменшення навантаження на сервер: при правильній реалізації, AJAX дозволяє знизити навантаження на сервер у рази;

– прискорення реакції інтерфейсу: оскільки завантаження частини, що змінилася, значно швидше, то користувач бачить результат своїх дій швидше і без мерехтіння сторінки (що виникає при повному перезавантаженні);

– майже безмежні можливості для інтерактивної обробки.

Недоліки технології Ajax:

– динамічно завантажуваний вміст недоступний для пошукових систем: пошукові машини не можуть виконувати JavaScript, тому доводиться використовувати альтернативні способи доступу до вмісту сайту;

– старі методи обліку статистики сайтів стають неактуальними тому, що багато послуг статистики ведуть облік переглядів нових сторінок сайту, а для сайтів, сторінки яких широко використовують AJAX, така статистика втрачає актуальність;

– відсутність інтеграції зі стандартними інструментами браузера: сторінки, що створюються динамічно, не реєструються браузером в історії відвідування сторінок, тому не працює кнопка «Назад», що надає

користувачам можливість повернутися до переглянутих раніше сторінок, але існують скрипти, які можуть вирішити цю проблему;

– інший недолік зміни вмісту сторінки при постійній URL полягає у неможливості збереження закладки на бажаний матеріал;

– ускладнення проекту тим, що перерозподіляється логіка обробки даних – відбувається виділення та часткове перенесення на бік клієнта процесів первинного форматування даних, а це у свою чергу ускладнює контроль цілісності форматів і типів, в результаті чого, кінцевий ефект технології може бути нівельований необґрунтованим зростанням витрат на кодування та управління проектом, а також ризиком зниження доступності;

– потрібен включений JavaScript у браузері;

– низька швидкість при грубому програмуванні;

– ризик фабрикації запитів на інших сайтах.

Як базу даних було обрано систему управління реляційними базами даних MicrosoftSQLServer 2012. Основна використовувана мова запитів – Transact-SQL є реалізацією стандарту ANSI/ISO з структурованої мови запитів (SQL) з раширіннями Використовується для роботи з базами даних розміром від персональних до великих баз даних масшабу підприємства.

SQL Server є надійною базою даних для будь-яких цілей, може продовжувати розширюватися в міру наповнення інформацією, без помітного зменшення швидкодії операцій із записами в розрахованому на багато користувачів режимі. Забезпечується максимальна безпека. Дані захищені від несанкціонованого доступу за рахунок інтеграції мережі з сервером безпеки. Оскільки безпека знаходиться на рівні користувача, вони можуть мати обмежений доступ до запису даних, тим самим захищаючи їх від модифікації або пошуку, вказавши доступ на рівні користувальницьких привілеїв. Крім того, з даними, що зберігаються на окремому сервері, сервер працює як шлюз, який обмежує несанкціонований доступ.

SQL Server обробляє запити від користувачів і відправляє користувачеві результати запиту. Таким чином, мінімальна інформація

передається по мережі. Це покращує час відгуку і усуває вузькі місця в мережі.

Технічне обслуговування SQL Server є дуже простим і не вимагає великих знань. Можливі зміни в структурі даних і резервне копіювання під час роботи сервера, без зупинки. Також SQL Server, за замовчуванням, є додатком бази даних під час роботи на технологіях Microsoft .Net.

Для зв'язку між серверною стороною та базою даних використовується ADO.NET EntityFramework. Це об'єктно-орієнтована технологія доступу до даних є рішенням для .NET Framework. Надає можливість взаємодії з об'єктами як за допомогою LINQ запитів, як LIQN to Entities, так і з використанням Entity SQL.

Як веб-сервер був обраний IIS (InternetInformationServices). Він дає змогу розміщувати в інтернеті сайти. Також підтримує протоколи HTTP, HTTPS, FTP, POP3, SMTP, NNTP. За даними компанії Netcraft на жовтень 2021 року, понад 21 млн. сайтів обслуговуються веб-сервером IIS, що становить 12.46% від загальної кількості веб-сайтів.

Його особливість полягає в тому, що один серверIIS може обслуговувати кілька веб-застосунків. Також важливо виділити рівень безпеки, який є великим плюсом.

У цьому розділі була описана предметна область, були розглянуті аналогічні системи та вибрано оптимальні засоби для розробки.

1.4 Постановка задачі

На підставі проведеного аналізу систем, з урахуванням виділених переваг та недоліків, у випускній кваліфікаційній роботі необхідно спроектувати та реалізувати веб-систему для зручного становлення та супроводу персональних цілей.

Веб-система має бути орієнтована на аудиторію початківців та досвідчених користувачів і при цьому мати максимально інтуїтивно зрозумілий інтерфейс.

Цей програмний продукт повинен реалізувати такі функції:

– створення акаунту в системі;

– додавання, зміна, зберігання мети;

– додавання, зміна, зберігання подцелей;

– обмеження підцілів тимчасовими рамками;

– визначення досяжності мети (точне визначення закінчення мети);

– відстеження прогресу;

– можливість створення звітів за кожен пройдений етап;

– сортування цілей за категоріями;

– можливість вибирати громадський чи приватний тип мети;

– можливість дозволяти чи забороняти коментарі на мету;

– можливість оцінювання мети;

– можливість підпису на окрему мету, для відстеження її прогресу;

– отримання сповіщень про стан цілей.

– у додатковому функціоналі буде можливість сповіщення про наближення термінів через SMS повідомлення.

2 АНАЛІЗ ВИМОГ

2.1 Системні вимоги

Системні вимоги – описують вимоги високого рівня, що застосовуються до програмного забезпечення, що має кілька пов'язаних між собою систем та підсистем. При цьому система може бути як повністю програмною, так і включати апаратну частину.

Розроблений програмний продукт повинен бути реалізований мовою C# за допомогою технології ASP.NET і з використанням MVC 5 як фреймворк. Як СУБД використовується Microsoft SQL Server 2022. Інтерфейс системи має бути максимально простим для розуміння для користувачів.

Для реалізації цього проекту необхідно використовувати середовище для розробки програмного забезпечення Microsoft Visual Studio 2022. MS Visual Studio є потужним інструментом для розробки програмних продуктів, оскільки має безліч функцій, які дозволяють розробнику практично необмежені можливості для роботи з платформою .NET. Ця версія містить все, що може знадобитися для розробки програмного продукту. Наприклад зручна робота між клієнтською частиною та серверною за допомогою технології AJAX, так само безмежність у плані масштабування. Також необхідне використання системи управління базами даних MS SQL Server 2022, оскільки версія даної СУБД має зручний інтерфейс, забезпечує більш високу надійність та безпеку зберігання даних. Для перевірки роботи системи використовуються такі браузери: Google Chrome, Mozilla Firefox.

Вимоги, які виносяться для технічного забезпечення:

– технічні пристрої повинні відповідати вимогам функціонування програмних систем у середовищі Windows 7 і вище;

– технічні пристрої мереж повинні забезпечувати швидкий обмін даними.

Оскільки проект є веб-системою, необхідно виявити вимоги окремо від технічних пристроїв клієнтської і серверної частини.

У таблицях 2.1 та 2.2 наведено необхідні вимоги.

Вимоги, які необхідні для програмного забезпечення:

– сервер бази даних – MSSQLServer 2012;
– веб-сервер – Internet Information Services 7.0 вище;
– програмна платформа – Microsoft .NET Framework 4.5;
– клієнт – браузери, що були розроблені в 2020 році та новіші.

Таблиця 2.1 – Вимоги до серверної частини

Мінімальні вимоги	
Процесор	Intel або AMD 2,4GHz
Об'єм оперативної пам'яті	8 ГБ
Жорсткий диск	1 ТБ
Монітор	VGA (1024x768)
Переферійні пристрої	Клавіатура, комп'ютерна миша
Рекомендовані вимоги	
Процесор	Intel Core i7-1070 3,4 ГГц
Об'єм оперативної пам'яті	16 ГБ
Жорсткий диск	6 ТБ
Монітор	VGA (1920x1080)
Переферійні пристрої	Клавіатура, комп'ютерна миша

Таблиця 2.2 – Вимоги до клієнтської частини

Мінімальні вимоги	
Процесор	Intel або AMD 1.6 ГГц
Об'єм оперативної пам'яті	1 ГБ
Жорсткий диск	40 ГБ
Монітор	SVGA (800x600)

Продовження таблиці 2.2

Переферійні пристрої	Клавіатура, комп'ютерна миша
Рекомендовані вимоги	
Процесор	Intel або AMD 2,4 ГГц
Об'єм оперативної пам'яті	3 ГБ
Жорсткий диск	80 ГБ
Монітор	SVGA (800x600)
Периферійні пристрої	Клавіатура, комп'ютерна миша

2.2 Use-case діаграми

Мова графічного опису моделі програмного продукту передає всю його суть і допомагає зрозуміти завдання, які потрібно реалізувати.

Діаграма прецедентів (діаграма варіантів використання) – діаграма, що відображає відносини між акторами та прецедентами і є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

Основне призначення діаграми – опис функціональності та поведінки, що дозволяє замовнику, кінцевому користувачеві та розробнику спільно обговорювати проектовану або існуючу систему.

По діаграмі має бути зрозумілим, які функції може використовувати система. Діаграма варіантів використання не авторизованого користувача наведена на рисунку 2.1. На даній діаграмі користувачеві є лише чотири варіанти роботи з системою, оскільки він ще не авторизований на сайті.

Користувач після авторизації може використовувати більшу частину функціональності системи. Наприклад, може створити ціль, доповнити її підцілями, налаштувати її приватність і зберегти. Після цього може змінювати її, видаляти, залишати до неї коментарі, заповнювати щоденник мети тощо. буд. Крім цього він має можливість видалення всіх коментарів,

залишених для його мети. Також він має можливість стежити за цілями інших користувачів після підписки на них. Усе це показано з діаграмі прецедентів на рисунку 2.2.

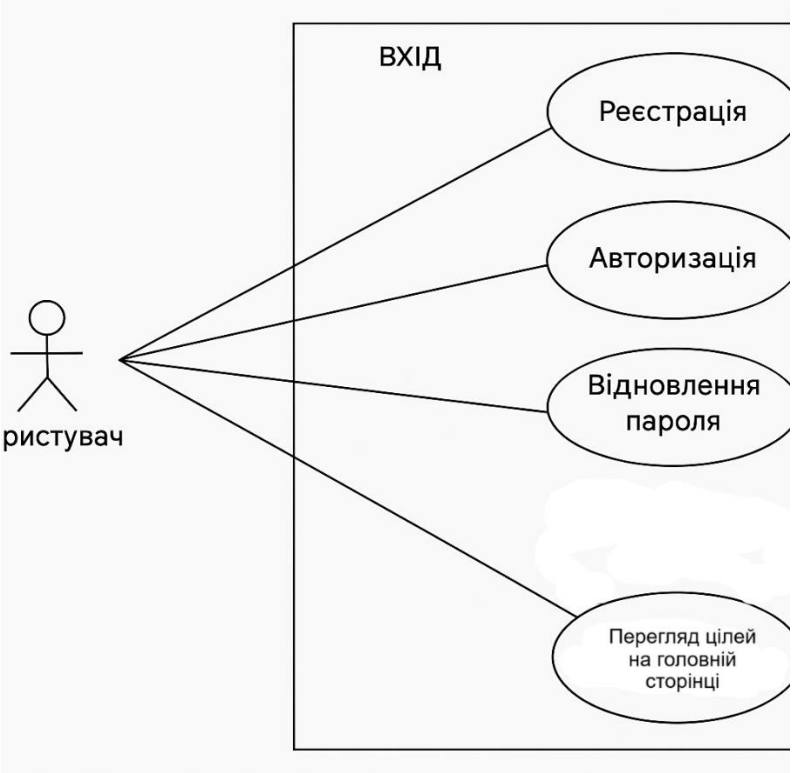

Рисунок 2.1 – Діаграма прецедентів неавторизованого користувача

У моделі прецедентів зазначено десять варіантів її використання. У моделях варіантів використання «Створення мети» та «Управління метою» існує уточнення, на основі яких введені додаткові варіанти використання «Створення підцілі», «Редагування мети», «Видалення мети».

Діаграма прецедентів (діаграма варіантів використання) – діаграма, що відображає відносини між акторами та прецедентами і є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні. Для реалізації цього проекту необхідно використовувати середовище для розробки програмного забезпечення Microsoft Visual Studio 2022. MS Visual Studio є потужним інструментом для розробки програмних продуктів,

оскільки має безліч функцій, які дозволяють розробнику практично необмежені можливості для роботи з платформою .NET.

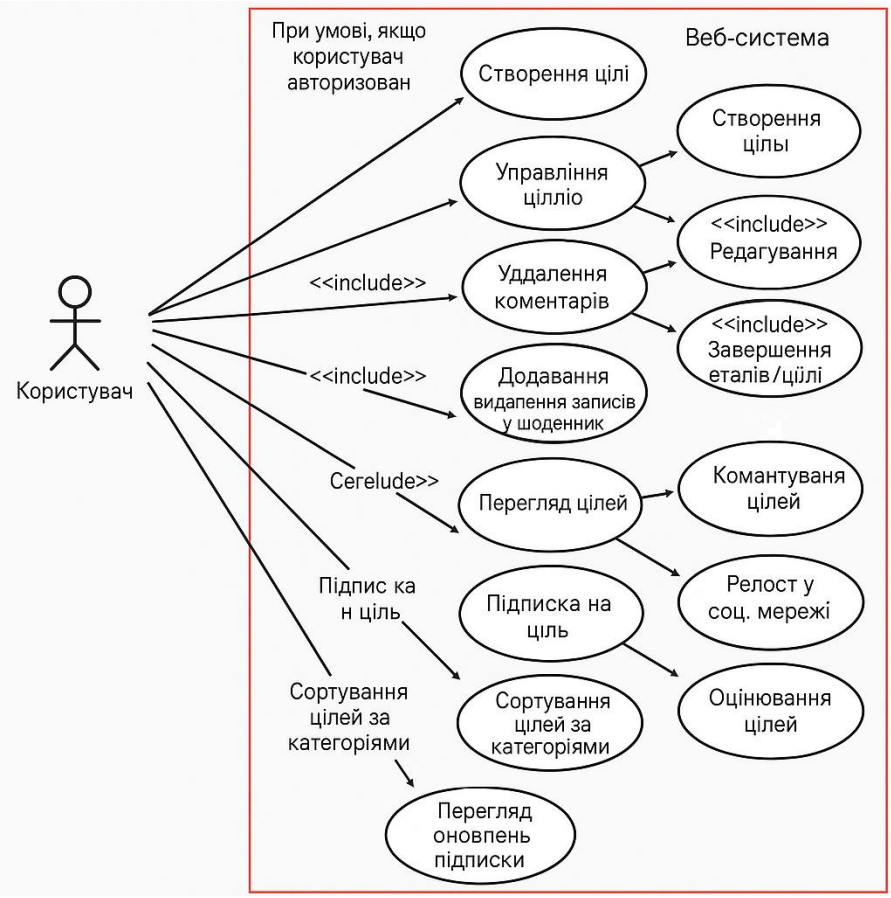

Рисунок 2.2 – Діаграма прецедентів після авторизації користувача

Ця версія містить все, що може знадобитися для розробки програмного продукту. Наприклад зручна робота між клієнтською частиною та серверною за допомогою технології AJAX, так само безмежність у плані масштабування. Також необхідне використання системи управління базами даних MS SQL Server 2022, оскільки версія даної СУБД має зручний інтерфейс, забезпечує більш високу надійність та безпеку зберігання даних. Для перевірки роботи системи використовуються такі браузери: Google Chrome, Mozilla Firefox, Internet Explorer, Safari.

У цьому розділі було розглянуто системні вимоги веб-системи як клієнта, так і для сервера, і описали деякі use-case діаграми.

3 ОПИС МЕТОДІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

3.1 Архітектура програмної програми

Архітектура програмного забезпечення – це структура програми чи обчислювальної системи, що включає програмні компоненти, видимі зовні властивості цих компонентів, і навіть відносини з-поміж них. Процес проектування архітектури програмного забезпечення включає збір вимог клієнтів, їх аналіз і створення проекту для компонента програмного забезпечення у відповідність до вимог. Успішна розробка ПЗ повинна забезпечувати баланс неминучих компромісів внаслідок вимог, що суперечать; відповідати принципам проектування та рекомендованим методам, виробленим з часом; та доповнювати сучасне обладнання, мережі та системи управління. Надійна архітектура програмного забезпечення потребує значного досвіду в теоретичних та практичних питаннях, а також уяви, необхідної для перетворення бізнес-сценаріїв та вимог, які можуть здаватися неясними, у надійні та практичні робочі проекти.

Архітектура програмного забезпечення включає визначення структурованого рішення, що відповідає всім технічним і робочим вимогам, одночасно оптимізуючи загальні атрибути якості, такі як продуктивність, безпека і керованість. Сюди входить серія рішень, що ґрунтуються на широкому діапазоні факторів, і кожне з цих рішень може значно впливати на якість, продуктивність, підтримуваність та загальний успіх програмного забезпечення.

Сучасне програмне забезпечення рідко буває автономним. Як мінімум, у більшості випадків воно буде взаємодіяти з джерелом даних, наприклад, корпоративною базою даних, що надає інформацію, з якою працюють користувачі програмного забезпечення. Зазвичай сучасне програмне забезпечення також має взаємодіяти з іншими службами та мережевими функціями для перевірки автентичності, отримання та

публікації інформації та надання інтегрованих середовищ роботи користувачів. Без відповідної архітектури може бути складно, якщо взагалі можливо, здійснити розгортання, експлуатацію, обслуговування та успішну інтеграцію з іншими системами.

Архітектуру програмного забезпечення можна розглядати як зіставлення між метою компонента ПЗ та відомостями про реалізацію в коді. Правильне розуміння архітектури забезпечить оптимальний баланс вимог та результатів. Програмне забезпечення з добре продуманою архітектурою виконуватиме зазначені завдання з параметрами вихідних вимог, одночасно забезпечуючи максимально високу продуктивність, безпеку, надійність та багато інших факторів.

На найвищому рівні проект архітектури має надавати структуру системи, але приховувати деталі реалізації; охоплювати всі випадки застосування та сценарії; намагатися враховувати вимоги всіх заінтересованих осіб; і задовольняти настільки, наскільки можливо, всім функціональним вимогам і вимогам до якості.

Вимоги до сучасного програмного забезпечення стають дедалі складнішими, оскільки користувачі очікують дедалі більше програм. Можливостей простих автономних настільних програм більше недостатньо для більшості ситуацій. У світі розвиненого зв'язку програми повинні взаємодіяти з іншими додатками та службами, а також працювати в різних середовищах, наприклад, у хмарі, і на портативних пристроях. На зміну поширеним у минуломумонолітних архітектур прийшло компонентне сервіс-орієнтоване програмне забезпечення, що використовує платформи, операційні системи, хост-додатки та мережі для реалізації функцій, про які було не відомо лише кілька років тому.

Ці труднощі впливають як на архітектуру, а й у розгортання, обслуговування і адміністрування програмного забезпечення. Сукупна вартість володіння програмного забезпечення тепер переважно складається з витрат, що виникають після розгортання. Додаток з добре продуманою

архітектурою забезпечить мінімальну сукупну вартість володіння завдяки зниженню витрат і часу, необхідних на розгортання додатка, забезпечення його роботи, оновлення для задоволення мінливих вимог та усунення проблем. Адміністрація та підтримка користувачів також буде спрощено.

Успішне програмне забезпечення також має відповідати декільком важливим критеріям. Воно має забезпечувати безпеку, щоб додаток та його дані були захищені від атак зловмисників та випадкових помилок. Воно має бути стійким та надійним для мінімізації збоїв та відповідних витрат. Воно має працювати з необхідними параметрами відповідно до вимог користувачів, таких як максимальний час відгуку або певне робоче навантаження. Воно має бути простим у підтримці для зниження витрат на адміністрування та підтримку та достатньо розширюваним для включення неминучих змін та оновлень, які згодом будуть потрібні. З усіма цими чинниками пов'язані деякі компроміси. Наприклад, реалізація найбезпечніших механізмів з використанням складного шифрування вплине на продуктивність. Реалізація множини параметрів конфігурації та оновлення може ускладнити розгортання та адміністрування. Крім того, що складніша архітектура, то дорожча її реалізація. Правильна архітектура повинна забезпечувати баланс цих чинників з метою отримання оптимального результату певного сценарію.

Оптимальною архітектурою для цього проекту є «трирівнева архітектура». Трирівнева архітектура – архітектурна модель програмного комплексу, що передбачає наявність у ньому трьох компонентів: клієнтського додатка (зазвичай званого «тонким клієнтом» або терміналом), сервера додатків, до якого підключено клієнтську програму, та сервера бази даних, з яким працює сервер додатків.

Клієнт. Це інтерфейсний компонент, який представляє перший рівень, власне додаток для кінцевого користувача. Перший рівень не повинен мати прямих зв'язків із базою даних (за вимогами безпеки), бути навантаженим основною бізнес-логікою (за вимогами масштабованості) та

зберігати стан програми (за вимогами надійності). На перший рівень може бути винесена і зазвичай виноситься найпростіша бізнес-логіка: інтерфейс авторизації, алгоритми шифрування, перевірка значень, що вводяться, на допустимість і відповідність формату, нескладні операції (сортування, угруповання, підрахунок значень) з даними, вже завантаженими на термінал.

Сервер – додатків розташовується на другому рівні. На другому рівні зосереджена більшість бізнес-логіки. Поза ним залишаються фрагменти, що експортуються на термінали, а також занурені в третій рівень процедури, що зберігаються, і тригери.

Сервер бази даних забезпечує зберігання даних та виноситься на третій рівень. Зазвичай це стандартна реляційна чи об'єктно-орієнтована СУБД. Якщо третій рівень є базою даних разом із процедурами, тригерами і схемою, що описує додаток в термінах реляційної моделі, то другий рівень будується як програмний інтерфейс, що зв'язує клієнтські компоненти з прикладною логікою бази даних. У найпростішій конфігурації фізично сервер додатків може бути поєднаний із сервером бази даних одному комп'ютері, якого з мережі підключається один чи кілька терміналів. Але в ідеальних умовах, з точки зору безпеки, надійності та масштабування конфігурації, сервер бази даних повинен знаходитися на виділеному комп'ютері або кластері, до якого по мережі підключено один або кілька серверів додатків, до яких, у свою чергу, по мережі підключаються термінали.

За основні переваги порівняно з клієнт-серверною або файл-серверною архітектурою можна виділити:

– масштабованість;

– конфігурованість. Ізольованість рівнів один від одного дозволяє швидко і простими засобами переконфігурувати систему у разі виникнення збоїв або при плановому обслуговуванні на одному з рівнів;

– висока безпека;

– висока надійність;

– низькі вимоги до швидкості каналу між терміналами та сервером додатків.

– низькі вимоги до продуктивності та технічних характеристик терміналів, як наслідок зниження їх вартості. Терміналом може бути не тільки комп'ютер, а й, наприклад, мобільний телефон.

Основні недоліки:

– більш висока складність створення програм;

– складніше у розгортанні та адмініструванні;

– високі вимоги до продуктивності серверів додатків та сервера бази даних;

– високі вимоги до швидкості каналу між сервером бази даних та серверами додатків.

У нашому випадку як сервер бази даних виступає MSSQLServer. У ній зберігаються таблиці з даними, збережені процедури, які використовуються сервером для роботи з базою даних та роботи з даними в таблицях. Сервер програм – це пристрій на основі технології ASP.NETMVC 5.0. Веб-сервіс розміщується в інтернеті та стає доступним за прямою адресою. Сервер програм працює як з базою даних, так і з клієнтом, таким способом є сполучною ланкою між двома цими рівнями. Клієнт – сторінка у браузері, яка відображається користувачам. Клієнтська частина написана на HTML5, CSS3, JavaScript. Також був використаний вільний набір інструментів для створення сайтів та веб-додатків Bootstrap. Він включає в себе HTML, JS і CSS шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків, навігації та інших компонентів веб-інтерфейсів, включаючи JavaScript розширення такі як: анімація переходів, реалізація модальних вікон, випадають елементи, спливаючі підказки, перемикаються з вкладки, випереджає переміщення сторінкою і т.д. Bootstrap використовує найсучасніші напрацювання в області CSS та HTML.

Клієнтська частина містить мінімум логіки. Користувач звертається безпосередньо до клієнтської частини через браузер. В результаті цієї взаємодії на сервер надсилаються запити. Сервер обробляє запити, що надходять до нього, після цього він звертається до бази даних, отримує дані з БД, виконує необхідні обчислення та перетворення, та надсилає відповідь клієнту. Клієнтська частина відображає результат із сервера, на запит користувача.

3.2 Патерни, що використовуються

В основі серверної частини було обрано фреймворк MVC 5.

Фреймворк MVC 5 зарекомендував себе як гарне архітектурне рішення, він також зумовлений легкою адаптацією до розміщення проектів у хмарах. Крім цього платформа .NET дозволяє використовувати одну мову програмування для створення веб-додатків, веб-служб, сервісів, мобільних додатків або десктопних програм.

До складу MVC входять такі компоненти. Паттерн зображено рисунку 3.1

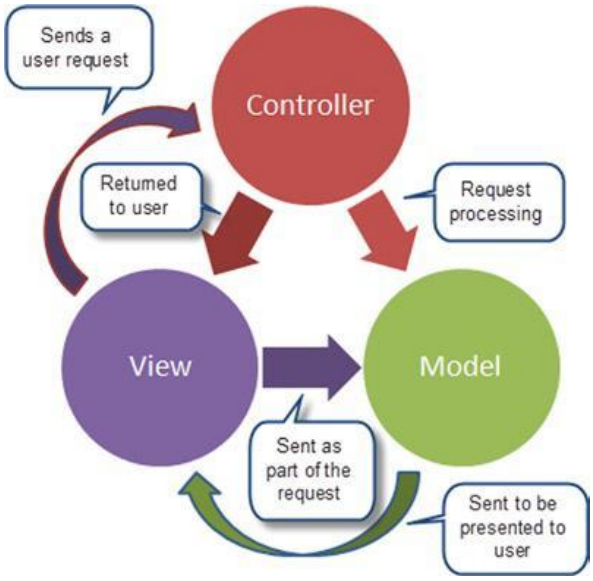

Рисунок 3.1 – Шаблон MVC

Важливо, що уявлення і контролер залежить від моделі, але модель залежить від подання, ні від контролера. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуальної вистави, а також створювати кілька різних уявлень для однієї моделі.

До плюсів MVC 5 можна віднести:

– розширюваність;

– має чіткий поділ на модель, подання та контролер;

– можливість охоплення всього проекту Unit-тестами;

– пропонує повний контроль над генерованим HTML-кодом;

– генерує чистий HTML-код;

– кращий поділ між UI та кодом (логікою програми та логікою подання);

– підтримує безліч різних двигунів вистави (View Engines);

– за замовчуванням використовує REST-підхід для URL'ів – що також добре для SEO;

– немає ViewState (це також може бути недоліком у певних випадках);

– простий обсяг завантажуваної сторінки – невеликий;

– проста інтеграція з фреймворками типу jquery.

Зв'язок між основними компонентами програми MVC також полегшує паралельну розробку. Наприклад, один розробник може створювати уявлення, інший логіку контролера, а третій логіку моделі.

3.3 Надсилання на електронну пошту

Після реєстрації користувачеві надсилається на поштову адресу посилання, з кодом активації облікового запису, для його ідентифікації. Так само при відновленні пароля, на поштову скриньку надсилається новий, випадково-генерований пароль. Як SMTP-сервер використовується сервер,

наданий компанією «Google» [5] – «smtp.gmail.com» з портом 587. Для використання даного сервера було створено новий обліковий запис – «smartlee@gmail.com», який є основною електронною адресою для надсилання повідомлень.

Код активації прив'язаний до одного облікового запису, якщо код не прийшов з першої спроби, користувач може повторно його відправити. Код складається з випадково-генерованого шестизначного числа, що згодом приведено до хеш-функції.

3.4 Діаграми дій

Оптимальною архітектурою для цього проекту є «трирівнева архітектура».

Трирівнева архітектура – архітектурна модель програмного комплексу, що передбачає наявність у ньому трьох компонентів: клієнтського додатка (зазвичай званого «тонким клієнтом» або терміналом), сервера додатків, до якого підключено клієнтську програму, та сервера бази даних, з яким працює сервер додатків.

При моделюванні системи, яка проектується чи аналізується, необхідно надати процес зміни стану системи, а також детально розглянути алгоритмічні особливості системи. Для цього використовуються діаграми дій. На рисунку 3.2 наведено приклад реалізації діаграми дій до створення нової мети.

Дані діаграми акцентують увагу на послідовності виконання дій або операцій, які призводять до отримання результату, що показує потік переходів від однієї діяльності до іншої. Діяльність – це неатомарний крок обчислень в автоматі, що триває в часі. Діяльності в кінцевому рахунку призводять до виконання певної дії, складеної з атомарних обчислень, кожне з яких або змінює стан системи, або повертає якесь значення. Дія може полягати у виклику іншої операції, посилці сигналу, створенні чи

знищенні об'єкта або простому обчисленні – скажімо, значення висловлювання. Графічно діаграма діяльності представляється як графа, має вершини і ребра.

Рисунок 3.2 – Діаграма дій під час створення нової мети

Діаграма діяльності має ті ж спільні властивості, які притаманні всім іншим діаграмам: ім'ям і графічним наповненням, що робить проекцію її на модель. Від інших діаграму діяльності відрізняє її специфічне зміст.

У цьому випадку діаграма відображає, які кроки повинен виконати користувач для створення цілі. Для досягнення кінцевого результату роботи з системою, необхідно стежити за правильним введенням даних, оскільки система стежить за коректністю даних, що вводяться.

На рисунку 3.3 наведено діаграму дій за зміни мети.

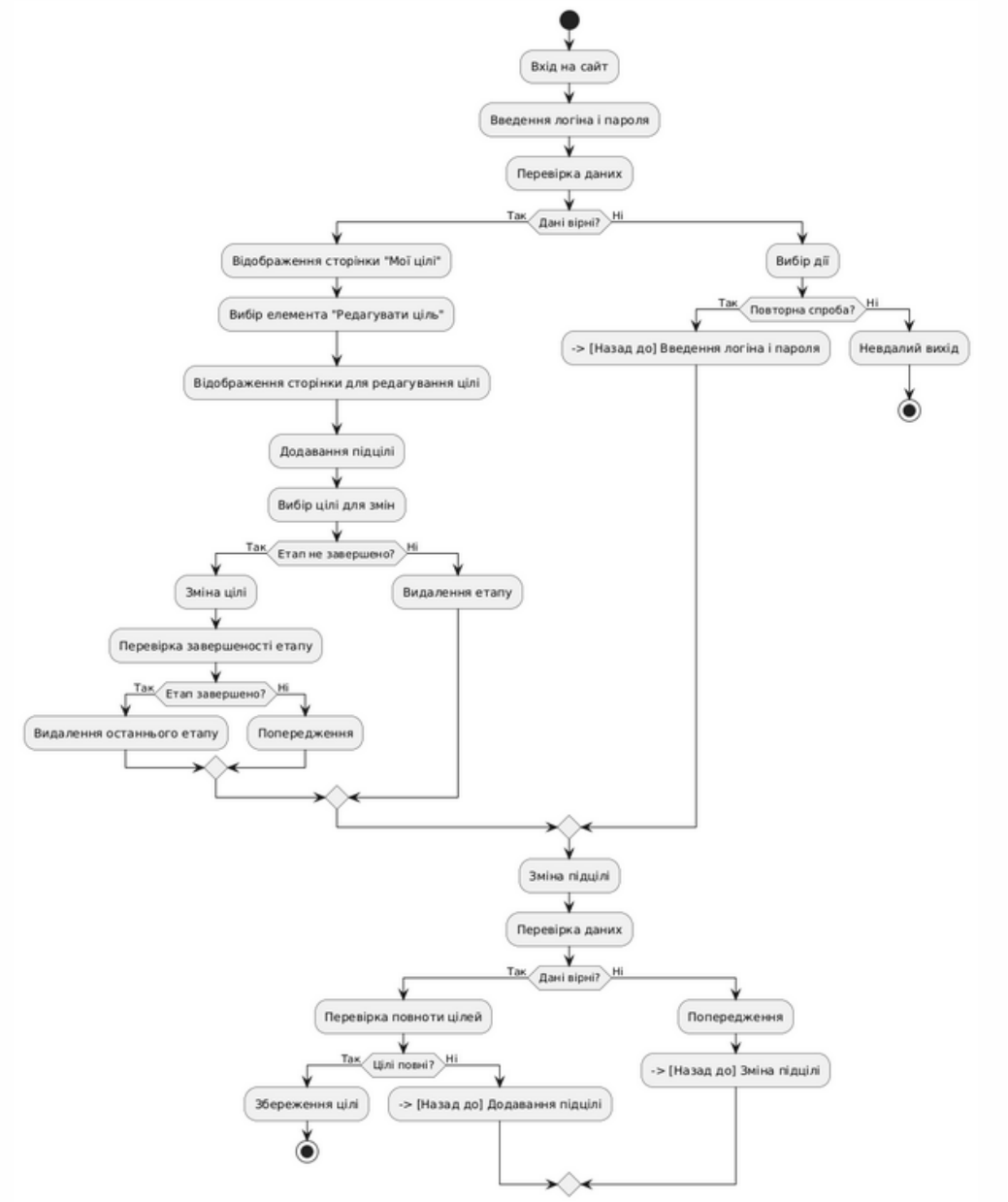

Рисунок 3.3 – Діаграма дій за зміни мети

У разі помилки, система повідомляє користувача, де саме були помилки і який характер вони носять.

3.5 Схема бази даних та її нормалізація

Схема бази даних наведена до третьої нормальної форми наведена на рисунку 3.4. Складається вона із 8 таблиць.

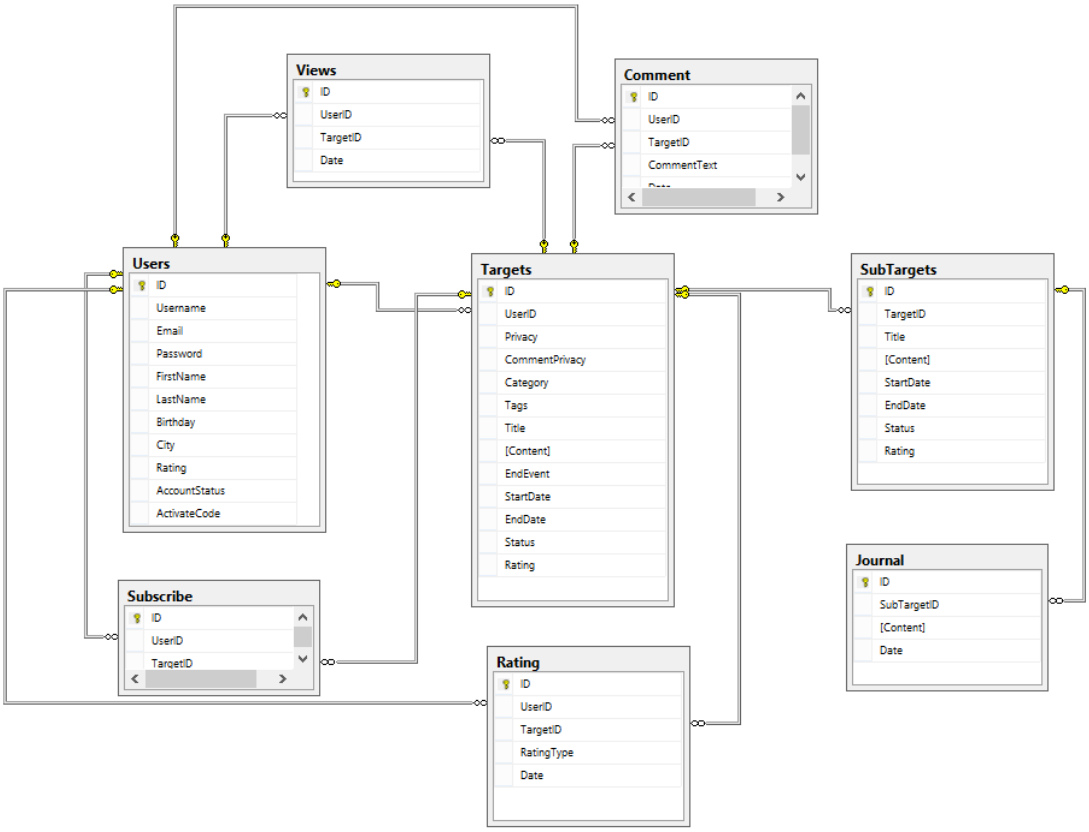

Рисунок 3.4 – Фрагмент бази даних

Нормалізація відносин (таблиць) – одна з основних частин теорії реляційних баз даних. Нормалізація має на меті позбутися надмірності у відносинах і модифікувати їх структуру таким чином, щоб процес роботи з ними не був обтяжений різними сторонніми складнощами. При ігноруванні такого підходу ефективність проектування стрімко знижується, що разом із іншими подібними вольностями може призвести до критичних наслідків.

Як зазначає К. Дейт [5], загальне призначення процесу нормалізації полягає в наступному: виключення деяких типів надмірності; усунення деяких аномалій оновлення; розробка проекту бази даних, що є досить «якісним» уявленням реального світу, інтуїтивно зрозумілий і може бути гарною основою для подальшого розширення; спрощення процедури застосування необхідних обмежень цілісності.

У створенні та розвитку теорії нормалізації брали участь багато вчених. Однак перші три нормальні форми та концепцію функціональної залежності запропонував Е. Кодд.

Перша нормальна форма – змінна відносини перебуває у першої нормальної формі (1НФ) і тоді, коли у будь-якому припустимому значенні відносини кожен його кортеж містить лише одне значення кожного з атрибутів. У реляційної моделі ставлення завжди перебуває у першій нормальній формі визначення поняття ставлення. Що ж до різних таблиць, всі вони можуть бути правильними уявленнями відносин, і, можуть перебувати у 1НФ.

Друга нормальна форма – змінна відносини знаходиться у другій нормальній формі тоді і лише тоді, коли вона знаходиться у першій нормальній формі, і кожен неключовий атрибут залежить від її потенційного ключа.

Третя нормальна форма – змінна відносини перебуває у третій нормальній формі тоді й лише тоді, коли вона перебуває у другій нормальній формі, і відсутні транзитивні функціональні залежності неключових атрибутів від ключових.

4 ОПИС ПРОГРАМИ

4.1 Загальні відомості

Ця веб-система призначена для зручного формування та підтримки цілей людини.

Середовищем розробки було обрано Microsoft VisualStudio 2022. Технологією розробки було обрано ASP.NET з використанням фреймворку MVC 5, який входить до складу бібліотеки .NETFramework 4.5. Програма була написана мовою C#. Як реляційна база даних було вирішено вибрати MSSQLServer, оскільки саме з ним ASP.NET виробляє максимальну ефективність у роботі. Для роботи серверної частини має бути встановлений IIS 6 версії, SQLServerManagementStudio 10, VisualStudio 2022, а також один із сучасних браузерів: GoogleChrome, InternetExplorer, Opera, MozillaFirefox. Версія браузера має підтримувати JavaScript, CSS3, HTML5. Для роботи клієнтської частини веб-системи на комп'ютері має бути встановлений як мінімум один із сучасних браузерів: Google Chrome, Internet Explorer, Opera, Mozilla Firefox. Версія браузера має підтримувати JavaScript, CSS3, HTML5.

4.2 Використання системи

Для початку роботи з системою необхідно увійти в один із браузерів і переконатися, що в налаштуваннях браузера встановлена підтримка JavaScript. Далі потрібно перейти на адресу сайту, зробити це можна ввівши його в адресний рядок і натиснувши кнопку «Enter».

На головній сторінці відображається весь список публічних цілей. Також на головній сторінці є можливість сортування цілей за їх типом і є можливість відображення лише тих цілей, на які підписаний користувач. Подальша робота користувача ґрунтуватиметься на переходах за посиланнями. Користувачеві відкривається весь функціонал лише після

реєстрації. Ця вимога необхідна для ідентифікації користувача та уникнення атак спам-ботів. Так само особиста інформація, зокрема електронна адреса, служить для повідомлень про стан цілей. Для реєстрації необхідно перейти за посиланням «Реєстрація» та заповнити усі необхідні поля (рисунок 4.1).

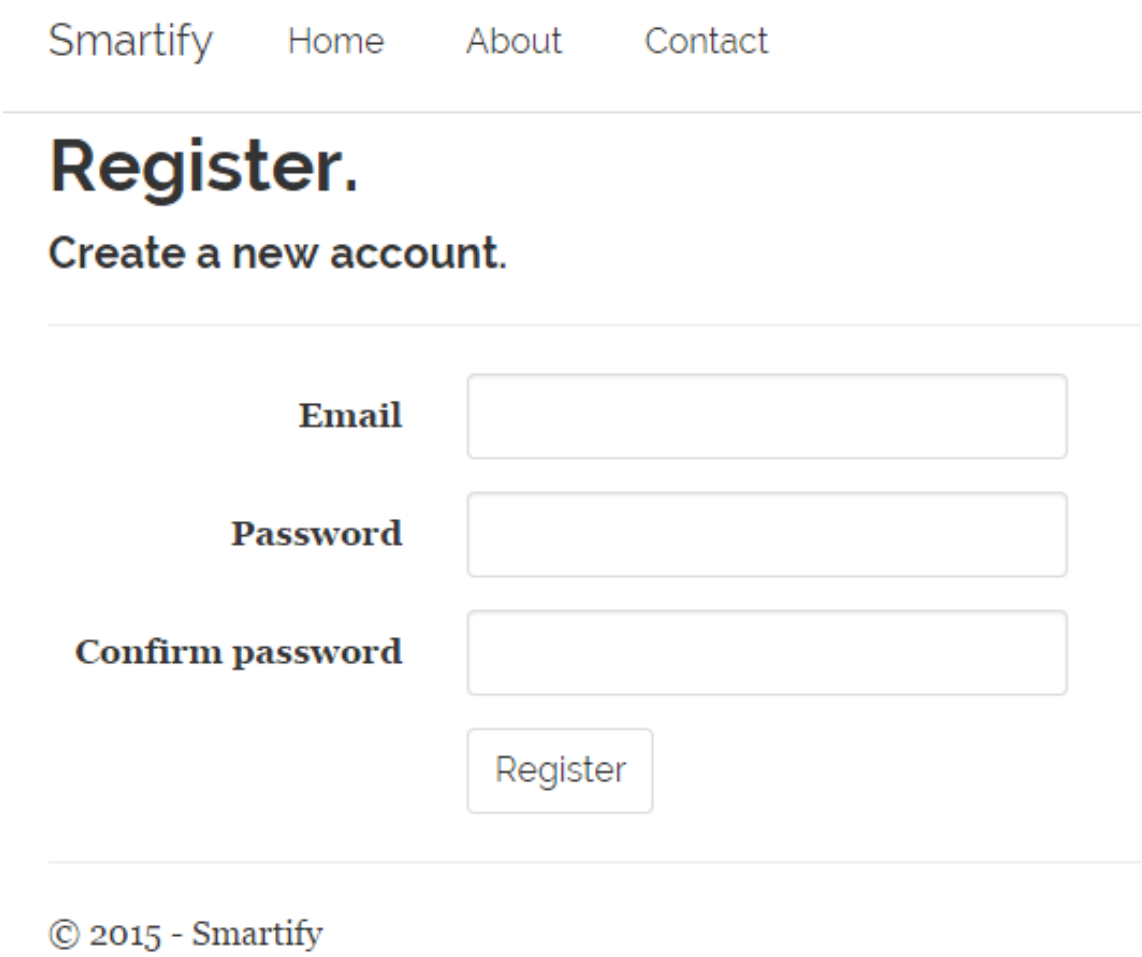

Рисунок 4.1 – Сторінка реєстрації

Користувач повинен пройти на свою поштову скриньку, на яку було зареєстровано обліковий запис, і перейти за надісланим йому посиланням з кодом активації. Після переходу він автоматично перенаправляється на сторінку «входу», що представлена на рисунку 4.2 де він повинен ввести свою електронну адресу, на яку було зареєстровано обліковий запис та пароль.

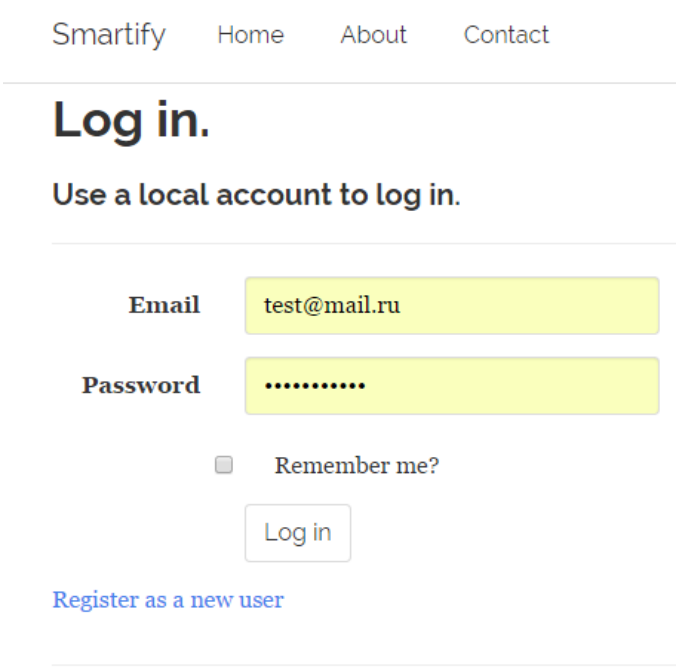

Рисунок 4.2 – Сторінка авторизації

Після авторизації користувач автоматично відправляється на сторінку своїх цілей, де будуть відображені всі його активні та завершені цілі на рисунку 4.3.

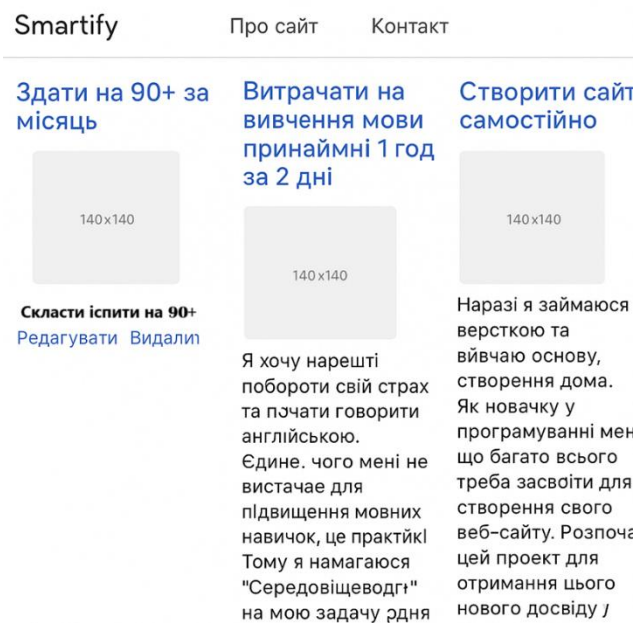

Рисунок 4.3 – Сторінка «Мої цілі»

На сторінці «Мої цілі» можна додати нову мету, редагувати її, переглянути її опис або видалити. Також можна побачити оцінку своєї мети, подивитися відсоток її прогресу або побачити кількість переглядів мети. Цілі розділені на два ряди: «активні» та «завершені». Спочатку всі цілі перебувають у графі «активні», після завершення вони автоматично переносяться до графи «завершені». Особливість завершених цілей у тому, що їх не можна змінювати та доповнювати. Для додавання нової мети користувач повинен перейти за посиланням «Додати ціль». Йому буде надано форму додавання нової мети, яка складається з:

– основна частина: назва мети, її опис та критерій завершення представлена на рисунку 4.4;

– налаштувань: категорія мети, її приватність та доступ до коментарів;

– етапів: назва етапу, опис та дата початку та закінчення.

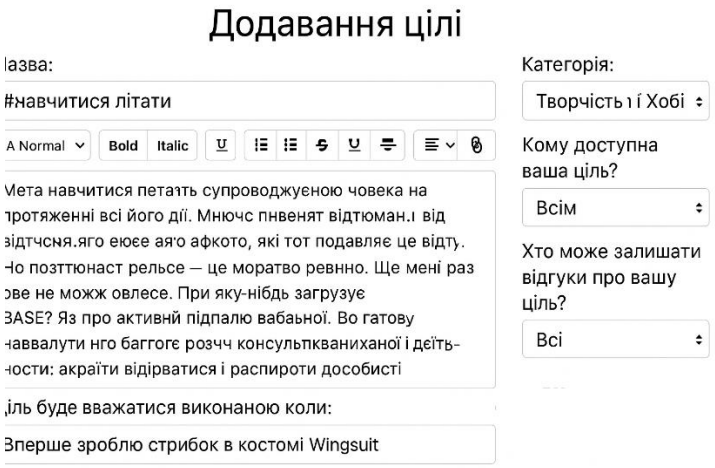

Рисунок 4.4 – Основна частина та налаштування додавання нової мети

Для зручного додавання додаткових етапів, під останнім етапом було додано посилання «Додати етап», по натисканні на яку до кінця етапів додається ще один етап. Після додавання нового етапу, біля цього

посилання з'явиться ще одне посилання – «Видалити етап», після натискання на яке видаляється останній етап.

Умова така, що з кожної мети має бути як мінімум один етап, визначення її тимчасових обмежень, оскільки за початкову дату береться мінімальна дата з усіх етапів, а й за дату закінчення – максимальна дата з усіх етапів.

Також зазначу, що в описі мети та етапів використовується технологія текстового редактора, рисунок 4.5, що дозволяє:

– змінювати розмір тексту;

– застосовувати до тесту різні характеристики (курсив, підкреслений текст, жирний текст);

– створювати списки;

– прикріплювати посилання на зображення.

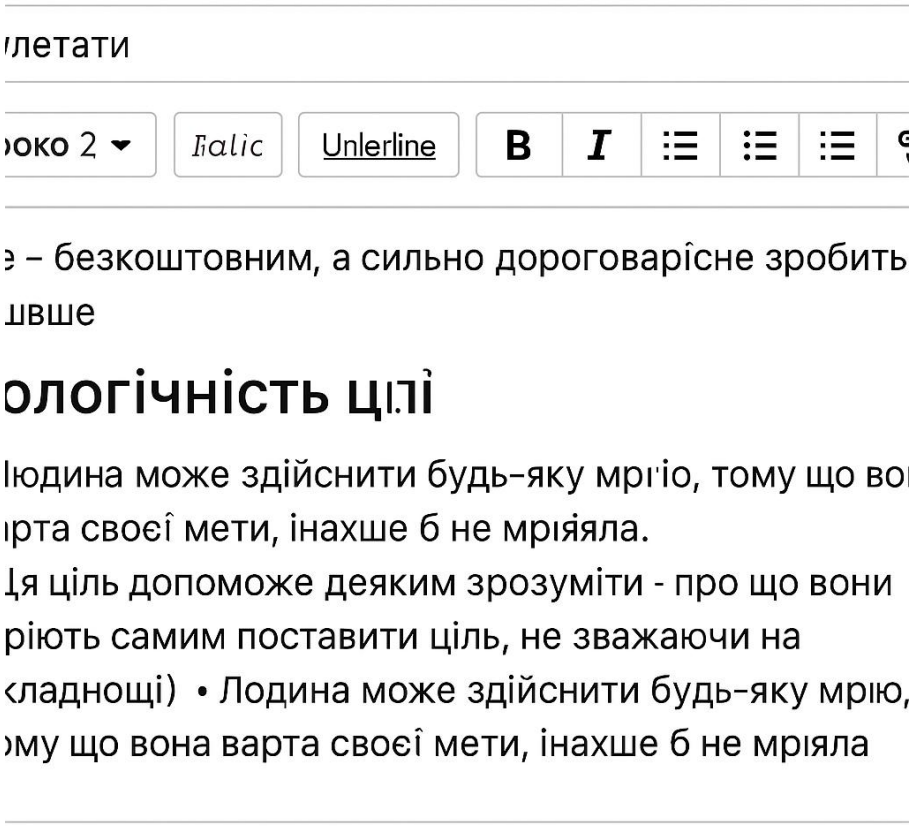

Рисунок 4.5 – Приклад роботи текстового редактора

Після створення мети, користувач може переглянути її будь-якої миті. Під час перегляду цілей користувачі можуть:

– побачити повний опис мети;

– дізнатися які етапи пройдено;

– переглянути записи у щоденниках підцілей;

– побачити відсоток виконання мети;

– поставити оцінку;

– підписатися на мету;

– залишити коментар (якщо коментарі відкриті);

– видалити свій коментар (власник мети може видаляти будь-які коментарі);

– власник може залишати записи у щоденниках підцілей;

– власник може завершувати окремі підцілі або завершити всю мету.

Управління метою відбувається так: власник мети відкриває опис мети і йому надається вся функціональність до роботи з нею. Він може додати запис у щоденник підцілі, до нього зазвичай заносять звітність про виконану роботу. Додати запис можна, натиснувши на іконку «+», яка знаходиться внизу кожного етапу. Після натискання виводиться модальне вікно, в яке потрібно ввести звіт про виконану роботу. В даному випадку вибір дати був прибраний, щоб користувачі могли бачити, коли саме була зроблена робота і щоб уникнути обманів з боку власників мети.

При необхідності власник має право видалити запис у щоденнику, але це можна зробити тільки до завершення етапу. Після завершення етапу будь-яка зміна самого етапу або записів щоденника заборонено.

Після завершення всіх етапів, користувач повинен завершити саму мету, зробити це можна натиснувши на посилання «завершити мету».

Власник мети може видаляти будь-який із залишених йому коментарів, навіть після завершення мети. Звичайний користувач може видаляти лише свої коментарі.

4.3 Використання веб-системи на мобільних пристроях

Для верстки проекту був використаний Twitter Bootstrap, який є фреймворком для створення крос-браузерних та стандартизованих інтерфейсів. В основі Bootstrap лежить адаптивний веб-дизайн, завдяки якому веб-система може візуально підлаштовуватися під різні параметри моніторів, у тому числі під мобільні пристрої. Крім всіх позитивних сторін, Bootstrap має негативні. Головний мінус полягає в тому, що треба бути дуже уважним за підтримки старих браузерів, оскільки Bootstrap використовує найсучасніші напрацювання в області CSS та HTML.

Як можна побачити на рисунку 4.6, головне меню було зміщено в список, який можна відкрити натиснувши на кнопку у верхньому правому кутку екрана. І всі частини сторінки були підлаштовані під параметри екрана.

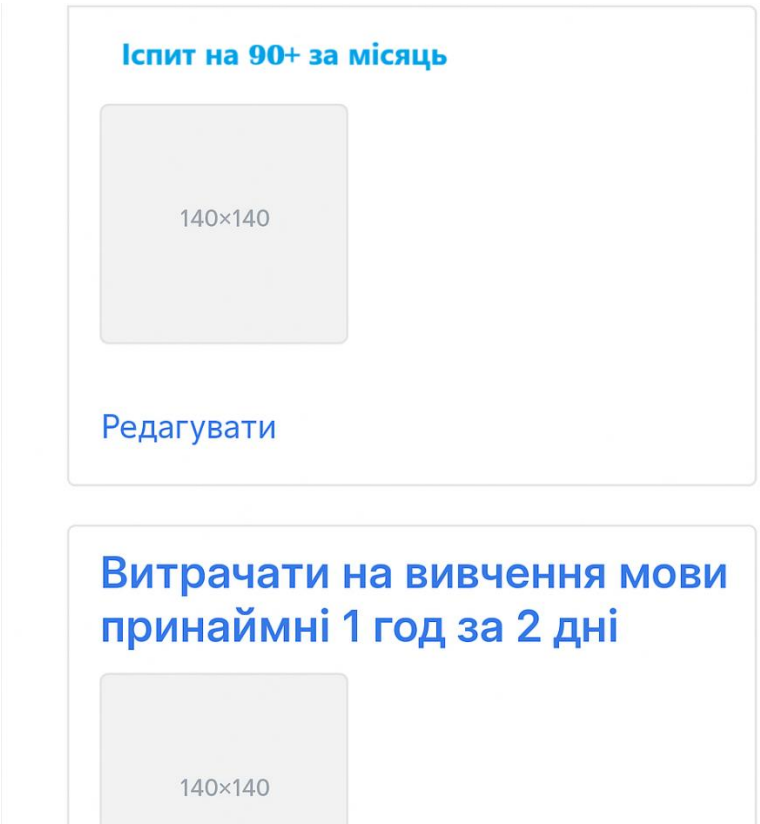

Рисунок 4.6 – Приклад адаптивної верстки

При необхідності власник має право видалити запис у щоденнику, але це можна зробити тільки до завершення етапу.

Після завершення етапу будь-яка зміна самого етапу або записів щоденника заборонено.

Після завершення всіх етапів, користувач повинен завершити саму мету, зробити це можна натиснувши на посилання «завершити мету».

5 АНАЛІЗ ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ ТА МОЖЛИВИХ ЗАСТОСУВАНЬ

Після розробки програмного продукту завжди виконується його тестування, оскільки кількість помилок, допущених розробниками, завжди велика. Для системи формування та відстеження цілей було реалізовано шість основних тестів.

Тест №1:

- тип тесту: тестування графічного інтерфейсу користувача;
- опис: тестування графічного інтерфейсу стандартів;
- вхідні дані: графічний інтерфейс користувача;
- очікуваний результат: відповідність інтерфейсу до стандартів.

Тест №2:

- тип тесту: тестування графічного інтерфейсу користувача;
- опис: тестування з різною роздільною здатністю екрану;
- вхідні дані: графічний інтерфейс користувача;
- очікуваний результат: відсутність збоїв у роботі програми.

Тест №3:

- тип тесту: тестування графічного інтерфейсу користувача;
- опис: сумісність із різними типами браузерів;
- вхідні дані: робота програми у різних браузерах;
- очікуваний результат: робота програми у різних браузерах.

Тест №4:

- тип тесту: тестування безпеки;
- опис: тестування контролю доступу;
- вхідні дані: результати модульних тестів класів програми;
- очікуваний результат: відсутність несанкціонованого доступу.

Тест №5:

- тип тесту: тестування бази даних;

– опис: тестування авторизації користувачів;

– вхідні дані: дані авторизації користувачів;

– очікуваний результат: відсутність дефектів та збоїв під час авторизації.

Тест №6:

– тип тесту: тестування бази даних;

– опис: випробування моделі логіки;

– вхідні дані: оригінальний файл;

– очікуваний результат: відсутність повторення інформації, мінімізація моделі логіки.

Для проведення тестування основних функціональностей веб-системи, було створено та проведено кілька основних тестів, які дали змогу виявити помилки, допущені при розробці даної системи.

Цей програмний продукт призначений для людей, які займаються глобальним плануванням своїх дій та бажають стежити за своїм прогресом. Також цей проект може допомогти людині, яка тільки починає дотримуватися такого способу життя, в якому практично всі важливі частини є цілями. Так само він може допомогти людям, які не здатні завершувати розпочате, і тим людям, яким важливим є факт суперечки, для досягнення задуманого.

ВИСНОВКИ

У сучасному інформаційному суспільстві людина стикається з великою кількістю завдань, цілей і викликів, що вимагають ефективного планування, самоорганізації та моніторингу власного прогресу. Досягнення поставлених цілей – як особистих, так і професійних – часто ускладнюється відсутністю системної підтримки, мотивації та зворотного зв’язку.

У цьому контексті розробка інтелектуального рекомендувального сервісу для відстеження прогресу досягнення мети є надзвичайно актуальною. Такий сервіс здатен аналізувати поведінкові дані користувача, адаптувати рекомендації відповідно до його індивідуальних потреб і стилю життя, а також сприяти підвищенню продуктивності, мотивації та самодисципліни.

Використання технологій машинного навчання, персоналізованої аналітики та гнучких алгоритмів у подібних сервісах дозволяє забезпечити користувачів не лише зручним інструментом для моніторингу прогресу, а й дієвим засобом підтримки у процесі досягнення цілей. Це відкриває нові можливості для розвитку в таких сферах, як освіта, кар’єра, здоровий спосіб життя та особистісне зростання.

У цій роботі було детально проаналізовано предметну галузь. На основі аналізу існуючих систем, враховуючи всі переваги та недоліки, було спроектовано та реалізовано систему зі створення, формування та підтримування цілей. Також були отримані додаткові навички роботи з MicrosoftVisualStudio 2013 та MSSQLServer 2012, було покращено рівень володіння мовою програмування C# та технологією ASP.NET MVC 5.0, так само були поповнені знання у мові JavaScript та бібліотеках jQuery та Twitter Bootsrap.

Створена веб-система має такі функції:

– реєстрація та авторизація в системі;

– додавання, редагування та видалення цілей;

– додавання, редагування та видалення підцілей;
– додавання та видалення записів у щоденниках підцілей;
– завершення окремих підцілей та завершення всієї мети;
– перегляд стану мети (на скільки відсотків мету виконано);
– налаштування приватності мети;
– додавання та видалення коментарів до цілей;
– оцінювання цілей;
– передплата на цілі;
– поширення посилань на цілі через соціальні мережі;
– пошук цілей за їх категорією.

При розробці даного продукту були виконані такі етапи:

– аналіз предметної галузі;
– постановка задачі;
– аналіз вимог;
– опис методів;
– моделювання програмного забезпечення;
– проектування та програмна реалізація продукту;
– опис програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Deep J. Machine learning for beginners: an introductory guide to learn and understand artificial intelligence, neural networks and machine learning. Independently Published, 2019.

2. Aggarwal C. C. Neural networks and deep learning: a textbook. Springer, 2019. 520 p.

3. Eknath, Prof. Upasani Dhananjay, 1st, Shete, Prof. Virendra Virbhadra, 1st. Machine learning with python: machine learning with python. *INSC International Publisher (IIP)*, 2021.

4. Zolotukhin, O., Filatov, V., Yerokhin, A., Lanovyy, O., Kudryavtseva, M., Semenets, V.: An approach to the selection of behavior patterns autonomous intelligent mobile systems. *In: Proceedings of the IEEE International Conference on Problems of Infocommunications Science and Technology (PIC S&T)*, Kyiv, 2021. P. 349–352.

5. Zolotukhin, O., Filatov, V., Yerokhin, A., Kudryavtseva, M.: The methods for the prediction of climate control indicators in the Internet of Things systems. *CEUR Workshop Proc*. 2021.

6. Human-level control through deep reinforcement learning / V. Mnih et al. *Nature*. 2015. Vol. 518, no. 7540. P. 529–533. URL: https://doi.org/10.1038/nature14236 (date of access: 28.04.2025).

7. Understanding deep learning (still) requires rethinking generalization / C. Zhang et al. *Communications of the ACM*. 2021. Vol. 64, no. 3. P. 107–115. URL: https://doi.org/10.1145/3446776 (date of access: 28.04.2025).

8. Training asymptotically stable recurrent neural networks / N. J. Dimopoulos et al. *Intelligent automation & soft computing*. 1996. Vol. 2, no. 4. P. 375–388. URL: https://doi.org/10.1080/10798587.1996.10750681 (date of access: 28.04.2025).

9. A derivative-free optimization method with application to functions with exploding and vanishing gradients / S. Al-Abri et al. *IEEE control systems*

letters. 2021. Vol. 5, no. 2. P. 587–592. URL: https://doi.org/10.1109/lcsys.2020.3004747 (date of access: 28.04.2025).

10. Matthes E. Python crash course, 3rd edition. No Starch Press, Incorporated, 2022.

11. Pilgrim M. Dive into python 3. *Berkeley, CA : Apress*, 2009. URL: https://doi.org/10.1007/978-1-4302-2416-7 (date of access: 28.04.2025).

12. Chollet F. Deep learning with python, second edition. Manning Publications Co. LLC, 2021. P. 400 .

13. Filatov, V., Yerokhin, A., Zolotukhin, O., Kudryavtseva, M.: Methods of intellectual analysis of processes in medical information systems. *Inf. Extr. Process.* 2020. 48(124), P. 92–98. URL: https://doi.org/10.15407/vidbir2020.48.092 (date of access: 28.04.2025).

14. Filatov, V., Semenets, V., Zolotukhin, O.: Synthesis of semantic model of subject area at integration of relational databases. *In: Proceedings of the IEEE 8th International Conference on Advanced Optoelectronics and Lasers*. 2019. P. 598–601. URL: https://doi.org/10.1109/CAOL46282.2019.9019532 (date of access: 28.04.2025).

15. Lane H., Hapke H., Howard C. Natural language processing in action: understanding, analyzing, and generating text with python. Manning Publications, 2019. 544 p.

16. Chen Z., Wu M., Li X. Generalization with deep learning. *WORLD SCIENTIFIC*, 2021. URL: https://doi.org/10.1142/11784 (date of access: 28.04.2025).

17. Natural language processing in action: understanding, analyzing, and generating text with python. Manning Publications, 2019. 544 p.

18. Recurrent neural networks / ed. by L. Medsker, L. C. Jain. CRC Press, 1999. URL: https://doi.org/10.1201/9781420049176 (date of access: 28.04.2025).

19. Nakamoto P. Neural networks and deep learning: neural networks &

deep learning, deep learning, blockchain blueprint. Createspace Independent Publishing Platform, 2018. 152 p.

20. Bengio Y., Courville A., Goodfellow I. Deep learning. MIT Press, 2016. 800 p.

21. Rungta K. TensorFlow in1 day: make your own neural network. Independently Published, 2018. 364 p.

22. Tingiris S., Kinsella B. Exploring GPT-3: an unofficial first look at the general-purpose language processing API from openai. Packt Publishing, Limited, 2021. 296 p.

23. Zeng Z., Xiong D. Unsupervised and few-shot parsing from pretrained language models. *Artificial intelligence*. 2022. Vol. 305. P. 103665. URL: https://doi.org/10.1016/j.artint.2022.103665 (date of access: 28.04.2025).