КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теорії та технології програмування

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

РОЗРОБКА WEB-ЗАСТОСУНКУ "MENTALHEALTHCARE"

Виконав студент 4-го курсу

Ткач Нікіта Сергійович

(підпис)

Науковий керівник:

асистент

Федорова Марія Вікторівна

(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«01» червня 2022 р.
протокол № 10
Завідувач кафедри
М. С. Нікітченко

(підпис)

Київ 2022

РЕФЕРАТ

Обсяг роботи 35 сторінок, 31 ілюстрацій, 4 таблиці, 15 джерел посилань.

МЕНТАЛЬНЕ ЗДОРОВ’Я, WEB-ЗАСТОСУНОК, ASP.NET CORE, ENTITY FRAMEWORK.

Предметом у даній роботі є Web-застосунок, створений нами засобами Microsoft SQL Server та ASP.NET Core, Entity Framework Core.

Метою роботи є створення web-додатку, що надав би можливість слідкувати та аналізувати стан власного ментального здоров’я, допоміг би контролювати себе у надзвичайних ситуаціях.

Інструменти розроблення: технології ASP.NET Core, Entity Framework, Microsoft SQL Server, Microsoft Visual Studio 2019, мова програмування C#, мова розмітки HTML, мова стилів CSS, мова програмування JavaScript.

Результати роботи: виконано загальний огляд та дослідження наявних додатків контролю ментального стану, та розроблено програмний продукт «MentalHealthCare».

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

EF – Entity Framework;

IDE – Integrated Design Environment, інтегроване середовище розробки;

SQL – Structured Query Language;

БД – база даних;

ОС – операційна система;

СУБД – система управління базами даних;

ВСТУП

Кожна людина кожен день відчуває спектр емоцій. Не завжди ці емоції позитивні. Коли відбувається щось жахливе або неочікуване людина переживає тривогу або шок і організм працює наче за програмою виживання. Далі емоції можуть чергуватися від злості, роздратування до ейфорії. Але стан ейфорії дуже енерговитратний для нашого організму, тому на зміну їй може прийти апатія, або навіть депресія.

Тому важливо слідкувати за своїм самопочуттям, та не уникати проблем зі своїм здоров'ям. Для цього треба аналізувати свій стан, та навчитися контролювати емоції, наприклад, за допомогою спеціального застосунку на телефоні або комп'ютері. До того ж, запис власних переживань – корисна психологічна практика.

Актуальність роботи та підстави для її виконання.

Ментальний стан українців в цей час не стабільний: ми перебуваємо в ненормальних обставинах. Перепади емоційних станів під час війни це типово. Постає питання, що треба зробити, щоб повернутись до адекватного стану. Розробка застосунку, що має допомогти справлятися зі своїми емоціями в цей скрутний для усіх час є дуже актуальною.

Мета й завдання роботи.

Метою роботи є створення Web-застосунку засобами ASP.NET Core, що надав би можливість слідкувати та аналізувати стан власного ментального здоров'я, допоміг би контролювати себе у надзвичайних ситуаціях.

Відповідно до мети ставляться наступні завдання.

- Дослідити предметну область розробки.
- Створити web-додаток «MentalHealthCare».
- Поглиблити та систематизувати знання у сфері веб-розробки.

Об'єкт, методи й засоби розроблення.

Об’єктом розроблення web-застосунку «MentalHealthCare» є процес розробки web-сайту для обраної сфери застосування. В якості інструменту створення web-додатку було обрано Microsoft Visual Studio 2019 IDE – інтегроване середовище розробки (IDE) мовою програмування C#, з використанням технологій ASP.NET Core, Entity Framework Core, що є безкоштовним, вільно поширюваним, з відкритим кодом.

ASP.NET має перевагу в швидкості в порівнянні з іншими технологіями, заснованими на скриптах, так як при першому зверненні код компілюється і поміщається в спеціальний кеш, і згодом тільки виконується, не вимагаючи витрат часу на оптимізацію.

Можливі сфери застосування.

Наш web-застосунок «MentalHealthCare» може бути використаний в особистих цілях для моніторингу та аналізу стану власного ментального здоров’я, для допомоги опанувати себе в надзвичайних ситуаціях.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд проблеми

З початком війни в Україні, дедалі більше людей стикаються з проблемами емоційного характеру, ментальний стан населення знаходиться в досить розкачаному стані.

Емоційні хвилі мають певний шаблон. Людина може відчувати зміни настрою протягом дня: спочатку їй сумно, за сумом йде тривога, далі вона може відчувати злість. Розглядаючи суспільство в цілому, можна побачити, що емоції змінюються наступним чином: спочатку людина розгублена, потім вона може відчувати тривогу, страх, паніку, невдовзі ці емоції замінять агресія та роздратування, далі буде фаза ейфорії, але вона не триватиме довго, її важливо використовувати конструктивно. І останній, найважчий з етапів – апатія, депресія. Апатія, втома та депресія зазвичай дуже швидко змінюють фазу ейфорії, і головне не застрягнути надовго в цьому етапі [14].

Після етапу апатії відбувається фаза стабілізації. Усі емоційні хвилі повторюватимуться, але оскільки людина з ними вже знайома, вони проходитимуть більш м'яко.

Людина думає, що буде у майбутньому, що робити у нових реаліях, вона починає звикати до невизначеності і тим часом розпочинає будувати плани на майбутнє. Усвідомлює, що все змінилося, і треба починати пристосовуватися до нових умов, починати жити відповідно їм [15].

Вкрай важливо прийти до фази адаптації, в якій людина буде діяти раціонально, свідомо та стабільно.

У стані адаптації слід зрозуміти, що людина знаходиться в нових умовах на невизначений термін. Треба вчитися працювати в цих реаліях та слідкувати за своїм емоційним станом.

1.2 Методи вирішення проблеми

Запис емоцій надає терапевтичну дію, допомагає впоратися з важкими ситуаціями та жити на повну [9].

Щоденник, який веде спортсмен містить плани тренувань та харчування, але результати одного дослідження свідчать, що якщо ви записуватимете в нього свої думки та почуття, це допоможе вам краще зрозуміти свої емоції та зміцнити психічне здоров'я.

Запис своїх думок і почуттів називається веденням щоденника. Доведено, що воно дуже корисне, особливо у стресових та тривожних ситуаціях, пов'язаних з труднощами або травмуючими подіями у житті. Такий вид "терапії листом" вперше використав доктор Джеймс Пеннебейкер, соціальний психолог Техаського університету в Остіні. Вона може стати цінним інструментом проживання та аналізу ситуацій та подій, з якими ви стикаєтесь [1].

Доктор Еліс Бойс каже, що ведення щоденника допомагає нам більш повно проживати ситуації, з якими ми стикаємося, а не постійно відмахуватися від емоцій [2].

Коли ви уникаєте думок та емоцій, вони утворюють незавершені цикли. Суть у тому, що ведення щоденника допомагає нам повніше проживати ситуації, із якими стикаємося, а чи не постійно закривати очі на складні емоцій, і потроху завершувати цикли, завдяки чому наш мозок може зациклюватися на них.

Як правило, не треба приділяти цьому надто багато часу. Було проведено невелике дослідження, в якому люди вели щоденник приблизно протягом 20 хвилин на день три дні поспіль, і навіть це дало певні позитивні результати у довгостроковій перспективі: вони змогли швидше оговтатися від стресової ситуації [10].

Джеймс Пеннебейкер, найвідоміший дослідник, який використовував цей підхід, навіть розробив кілька правил, яких ви можете дотримуватися. Він запропонував поставити таймер на 20 хвилин, відкрити блокнот або створити документ. Після запуску таймера почніть описувати емоції, які ви відчували протягом тижня, місяця або року. Не думайте про розділові знаки, помарок або зв'язність тексту. Просто щиро і без оцінки слідуйте за своїми думками. Пишіть собі, а чи не для якогось певного читача. Пишіть протягом декількох днів, а потім викиньте списані аркуші, покладіть їх у пляшку і киньте в море або закрийте документ, не зберігаючи його. Або якщо ви готові, заведіть блог або знайдіть літературного агента. Не має значення, що саме ви зробите. Сенс у тому, що ваші думки тепер не в голові, а на папері, і ви навчилися виходити за межі свого переживання, щоб його зрозуміти [1].

Як правило, це пов'язано з чимось дуже особистим. Фактично ви це робите, щоб прожити свої емоції. У ньому можна записувати свої враження (наприклад, від поїздки на Олімпіаду), щоб запам'ятати їх. Людські емоції можуть бути пов'язані з найважливішим моментом. У таких ситуаціях люди ведуть щоденник, щоб зберегти пам'ять, як вони його пережили [3].

Це не тільки спосіб упоратися зі стресом. Не звичайний спосіб боротьби з проблемами, коли замикаються емоційні «петлі». Це також можливість упоратися зі стресом від переживання.

Ми проживаємо свої емоції по-різному: просто дозволяємо думкам крутитись у голові, висловлюємо їх або записуємо. Це просто ще один спосіб проживання емоцій, який ми використовуємо, щоб трохи інакше структурувати думки. Протягом дня ми зазвичай зайняті важливими справами, тому завжди відганяємо думки про проблеми і не даємо собі можливості відчути свої емоції.

Ведення щоденника допоможе в будь-якій дуже важкій ситуації, і вам необхідно спеціально виділити час, щоб прожити емоції, що виникли. У таких випадках зазвичай є дві крайності: люди або закриваються від емоцій, або зациклюються на них. Тому, якщо ви весь день зайняті, ви можете

пригнічувати розчарування, гнів та інші неприємні сильні емоції, пов'язані з травмою чи тренером, який, на вашу думку, не вірить у вас чи пов'язані з іншими причинами. А потім ви лягаєте спати, і всі ці емоції звалюються на вас.

Люди намагаються уникати поганих думок або зациклюються на них, тому якщо протягом дня спеціально виділити час, щоб сконцентруватися на цих емоціях – тобто записати їх, це допоможе впоратися з перепадами настрою [8].

Тому, враховуючи необхідність мати дієвий спосіб виплеснути емоції, проаналізувати їх, та навчитися контролювати свій ментальний стан, було вирішено створити спеціальний застосунок, що допоможе в критичних ситуаціях.

РОЗДІЛ 2 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Для створення проекту було обрано зв’язку технологій ASP.NET Core + Microsoft SQL Server.

За середовище розробки проекту було обрано Visual Studio Community 2019 [7].

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів [4].

За допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі домінує, але тепер вже ми не обмежені тільки цією операційною системою. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll.

ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версії платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC [11].

ASP.NET Core характеризується розширюваністю. Фреймворк побудований з набору щодо незалежних компонентів. І ми можемо або використовувати вбудовану реалізацію цих компонентів, або розширити їх за

допомогою механізму спадкування, або зовсім створити і застосовувати свої компоненти зі своїм функціоналом.

Фреймворк тепер має свій легкий контейнер для впровадження залежностей, і більше немає необхідності застосовувати сторонні контейнери. Хоча при бажанні їх також можна продовжувати використовувати [13].

В якості інструментарію розробки ми можемо використовувати останні випуски Visual Studio, починаючи з версії Visual Studio 2015. Крім того, ми можемо створювати додатки в середовищі Visual Studio Code, яка є крос-платформної і може працювати як на Windows, так і на Mac OS X і Linux.

SQL Server є однією з найбільш популярних систем управління базами даних (СУБД) в світі. Дана СУБД підходить для самих різних проектів: від невеликих додатків до великих високонавантажених проектів [5].

SQL Server був створений компанією Microsoft. Перша версія вийшла в 1987 році. А поточною версією є версія 2019, яка вийшла в 2019 році і яка буде використовуватися в поточному керівництві [12].

SQL Server довгий час був винятково системою управління базами даних для Windows, проте починаючи з версії 16 ця система доступна і на Linux.

SQL Server характеризується такими особливостями як:

- Продуктивність. SQL Server працює дуже швидко.
- Надійність і безпека. SQL Server надає шифрування даних.
- Простота. З даної СУБД відносно легко працювати і вести адміністрування.

Центральним аспектом в MS SQL Server, як і в будь-який СУБД, є база даних. База даних являє сховище даних, організованих певним способом. Нерідко фізично база даних представляє файл на жорсткому диску, хоча таке відповідність необов'язково. Для зберігання і адміністрування баз даних застосовуються системи управління базами даних (database management system). І якраз MS SQL Server є однією з такою СУБД.

РОЗДІЛ 3 ПРИЗНАЧЕННЯ ТА ЦІЛІ СИСТЕМИ

3.1 Призначення системи

Задачею даної кваліфікаційної роботи є реалізація системи, що надавала б можливість аналізувати та слідкувати за станом власного ментального здоров’я, допомагала контролювати себе у надзвичайних ситуаціях.

Розробка повинна задовольняти наступним основним вимогам:

- надавати зручне середовище для постійного користування застосунком;
- надавати можливість додавати, зберігати, видаляти щоденний емоційний стан;
- надавати можливість здійснювати аналіз емоційного стану протягом місяця;
- надавати рекомендації щодо дій в надзвичайних ситуаціях.

3.2 Цілі системи

Застосунок «MentalHealthCare» розробляється з метою:

- надання моніторингу власного емоційного стану;
- надання інформації про дії в надзвичайних ситуаціях для контролю за ментальним здоров’ям;
- надання можливості вести контроль за емоційним станом.

РОЗДІЛ 4 ВИМОГИ ДО СИСТЕМИ

4.1 Вимоги до структури та функціонування системи

Web-застосунок «MentalHealthCare» повинен бути зручний у використанні, мати інтуїтивно-зрозумілий користувачеві дизайн та функціонал і мати просту для розуміння навігацію. Додаток повинен бути розроблений з українським інтерфейсом.

Система повинна мати можливість захисту даних, авторизацію та аутентифікацію користувачів.

Застосунок повинен пропонувати корисну інформацію для самодопомоги.

Також повинна бути реалізована додаткова інформаційна сторінка з усією потрібною інформацією для допомоги Збройним Силам України.

4.2 Технічні вимоги

Інтернет-браузери MS Internet Explorer 5.5 і вище, Mozilla 1.7 і вище, Opera 7.54 і вище.

На довільні некоректні дії користувача, пов’язані з введенням невірних даних, не заповненням обов’язкових полів введення в формах та інші, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки українською мовою, в межах загального дизайну сайту.

Операційна система: Windows, Linux, macOS.

Джерелом даних для Системи повинна бути інформаційна система Microsoft SQL Server.

РОЗДІЛ 5 ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

5.1 Логічна структура бази даних

Для реалізації системи «MentalHealthCare» була використана СУБД Microsoft SQL Server. Схема бази даних на рисунку …...

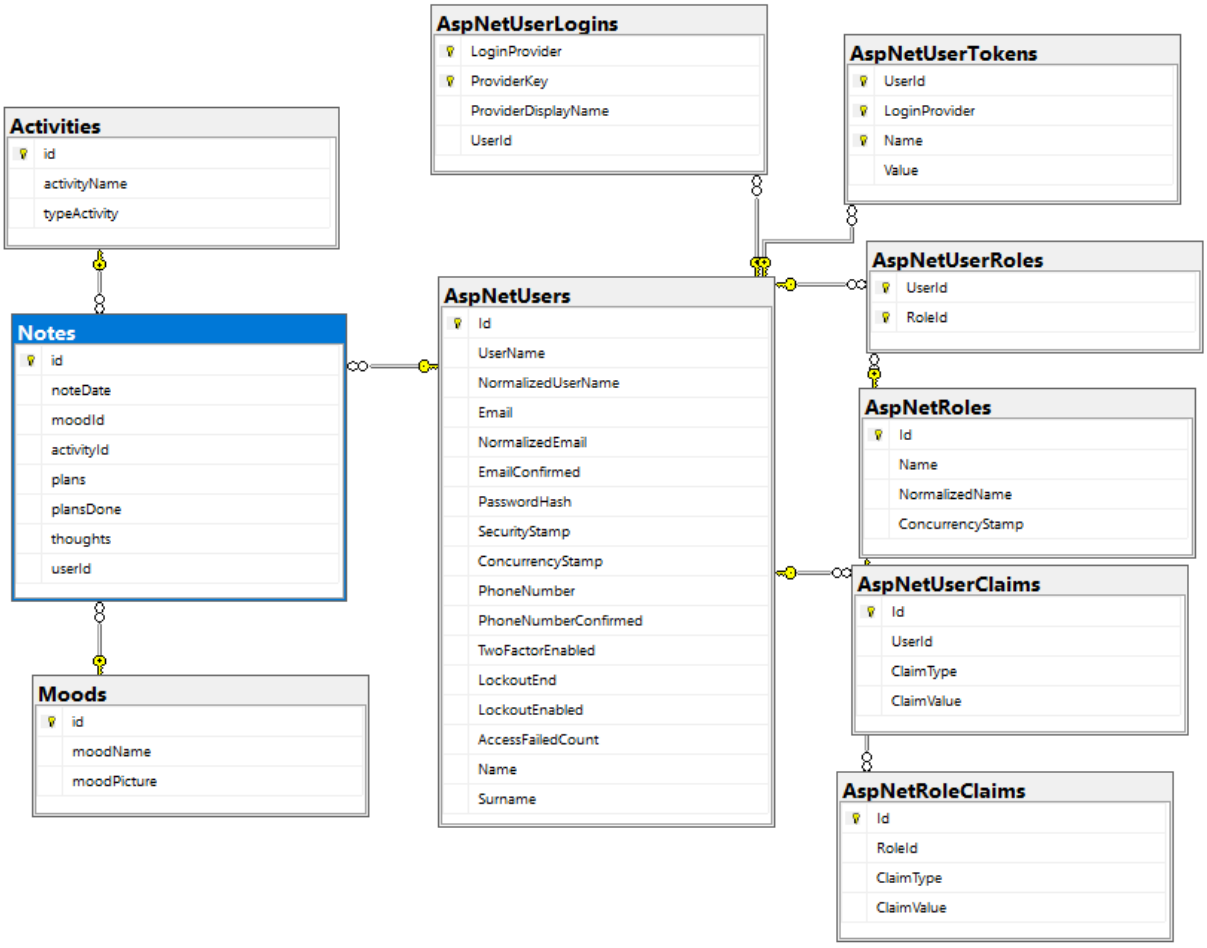

Рисунок 5.1 – Схема бази даних системи MentalHealthCare

5.2 Опис таблиць

У базі даних наявні таблиці, що описують записи користувача (таблиця 5.1), таблиця-словник активностей (таблиця 5.2), таблиця-словник настроїв (таблиця 5.3), та таблиця користувачів (таблиця 5.4), інші таблиці створені автоматично за допомогою ASP.NET Identity.

Таблиця 5.1 Notes

Атрибут	Тип	Опис
ID	Int	Ідентифікатор
noteDate	datetime	Дата
moodId	Int	Ідентифікатор
activityId	Int	Ідентифікатор
plans	nvarchar	Плани на наступний день
plansDone	bool	Чи були виконані плани
Thoughts	Nvarchar	Допис користувача
userId	nvarchar	Ідентифікатор

Таблиця 5.2 Activities

Атрибут	Тип	Опис
ID	Int	Ідентифікатор
activityName	nvarchar	Назва активності
typeActivity	bool	Тип активності

Таблиця 5.3 Moods

Атрибут	Тип	Опис
ID	Int	Ідентифікатор
moodName	nvarchar	Назва настрою
moodPicture	nvarchar	Картинка настрою

Таблиця 5.4 Users

Атрибут	Тип	Опис
ID	Int	Ідентифікатор
Name	nvarchar	І’мя користувача
Surname	nvarchar	Прізвище користувача
Email	nvarchar	Пошта користувача
noteId	int	Ідентифікатор
PasswordHash	nvarchar	Хеш паролю

РОЗДІЛ 6 РЕАЛІЗАЦІЯ СИСТЕМИ

Для взаємодії з MS SQL Server через Entity Framework через пакетний менеджер Nuget додамо в проект пакет Microsoft.EntityFrameworkCore.SqlServer.

Для підключення до бази даних, нам треба задати параметри підключення. Для цього змінимо файл appsettings.json, за замовчуванням він містить тільки налаштування логування (рисунок 6.1).

Рисунок 6.1 – Файл appsettings.json

Вбудована система AspNet Identity була перевизначена. Додано нові поля до моделі та нові таблиці (рисунок 6.2).

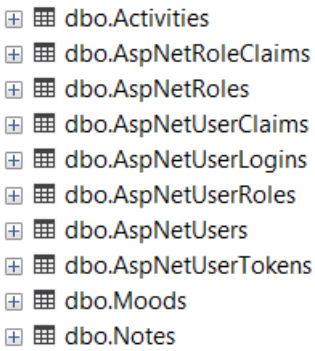

Рисунок 6.2 – База даних додатку

Створені моделі даних та контекст даних Code First підходом (рисунки 6.3 – 6.7).

```
public class Activity
{
    ссылка: 1
    public int id { get; set; }
    [Display(Name = "Активність")]
    Ссылок: 3
    public string activityName { get; set; }
    Ссылок: 0
    public bool typeActivity { get; set; }

    Ссылок: 2
    public virtual ICollection<Note> Notes { get; set; }
}
```

Рисунок 6.3 – модель даних Activity

```
public class Mood
{
    [Display(Name = "Настрій")]
    ссылка: 1
    public int id { get; set; }
    [Display(Name = "Настрій")]
    Ссылок: 3
    public string moodName { get; set; }
    [Display(Name = "Настрій")]
    ссылка: 1
    public string moodPicture { get; set; }

    Ссылок: 2
    public virtual ICollection<Note> Notes { get; set; }
}
```

Рисунок 6.4 – модель даних Mood

```
10  public class Note
11  {
        Ссылок: 10
12      public int id { get; set; }
13      [Display(Name = "Дата")]
14      [DisplayFormat(DataFormatString = "{0:dd/MM/yyyy}")]
        Ссылок: 12
15      public DateTime noteDate { get; set; }
16      [Display(Name = "Настрій")]
        Ссылок: 8
17      public int moodId { get; set; }
18      [Display(Name = "Вагоме заняття протягом дня")]
        Ссылок: 8
19      public int activityId { get; set; }
20      [Display(Name = "Плани на наступний день")]
21      [Required(ErrorMessage = "Поле не повинно бути порожнім")]
        Ссылок: 12
22      public string plans { get; set; }
23      [Display(Name = "Чи були виконані минулі плани")]
        Ссылок: 8
24      public bool plansDone { get; set; }
25      [Display(Name = "Думки протягом дня")]
26      [Required(ErrorMessage = "Поле не повинно бути порожнім")]
        Ссылок: 11
27      public string thoughts { get; set; }
28
29      [Display(Name = "Email")]
30      [ForeignKey("User")]
        Ссылок: 8
31      public string userId { get; set; }
32
33      [Display(Name = "Користувач")]
        Ссылок: 5
34      public virtual User User { get; set; }
35      [Display(Name = "Настрій")]
        Ссылок: 10
36      public virtual Mood Mood { get; set; }
37      [Display(Name = "Вагоме заняття протягом дня")]
        Ссылок: 9
38      public virtual Activity Activity { get; set; }
```

Рисунок 6.5 – модель даних Note

```
public class User : IdentityUser
{
    Ссылок: 3
    public string Name { get; set; }
    ссылка: 1
    public virtual string Surname { get; set; }

    Ссылок: 0
    public virtual ICollection<Note> Notes { get; set; }

}
```

Рисунок 6.6 – модель даних User

```
public class ApplicationDbContext:IdentityDbContext<User>
{
    Ссылок: 0
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
        Database.EnsureCreated();
    }

    Ссылок: 5
    public DbSet<Mood> Moods { get; set; }
    Ссылок: 5
    public DbSet<Activity> Activities { get; set; }
    Ссылок: 7
    public DbSet<Note> Notes { get; set; }
}
```

Рисунок 6.7 – Контекст даних

Для роботи з AspNet Identity модель даних User представляє користувача і успадковується від класу IdentityUser, переймаючи всі його властивості. Крім того, були додані поля ім’я, прізвище та ідентифікатор допису користувача (рисунок 6.6).

Щоб реалізувати реєстрацією і авторизацією користувачів створюємо моделі даних LoginViewModel та RegisterViewModel (рисунок 6.8-6.9).

```
public class LoginViewModel

{
    [Required]
    [Display(Name = "Email")]
    Ссылок: 4
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Пароль")]
    Ссылок: 4
    public string Password { get; set; }

    [Display(Name = "Запам'ятати?")]
    ссылка: 1
    public bool RememberMe { get; set; }
    Ссылок: 5
    public string ReturnUrl { get; set; }
}
```

Рисунок 6.9 – модель LoginViewModel

```
public class RegisterViewModel
{
    [Required]
    [Display(Name = "Email")]
    [DataType(DataType.EmailAddress)]
    Ссылок: 5
    public string Email { get; set; }

    [Required]
    [Display(Name = "Ім'я")]
    Ссылок: 4
    public string Name { get; set; }

    [Required]
    [Display(Name = "Прізвище")]
    Ссылок: 4
    public string Surname { get; set; }

    [Required]
    [Display(Name = "Пароль")]
    [DataType(DataType.Password)]
    [StringLength(16, ErrorMessage = "{0} повинен мати мінімум {2} и максимум {1} символів.", MinimumLength = 5)]
    Ссылок: 4
    public string Password { get; set; }

    [Required]
    [Compare("Password", ErrorMessage = "Паролі не співпадають")]
    [Display(Name = "Підтвердження паролю")]
    [DataType(DataType.Password)]
    Ссылок: 3
    public string PasswordConfirm { get; set; }
}
```

Рисунок 6.9 – модель RegisterViewModel

Для роботи з обліковими записами користувачів створюємо новий AccountController і визначаємо в ньому методи для реєстрації та авторизації користувачів (рисунок 6.10-6.11).

За допомогою методу CreateAsync користувач додається в базу даних. Як параметр передається сам користувач і його пароль.

Даний метод повертає об'єкт IdentityResult, за допомогою якого можна дізнатися успішність виконаної операції. Цілком можливо, що передані значення не задовольняють вимогам, і тоді користувача не буде додано до бази даних. У разі вдалого додавання за допомогою методу SignInAsync встановлюємо аутентифікаційні куки для доданого користувача. У цей метод передається об'єкт користувача, що ідентифікує, і логічне значення, яке вказує, чи треба зберігати куки протягом тривалого часу. Далі виконуємо переадресацію на головну сторінку програми.

```
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        User user = new User { Email = model.Email, UserName = model.Email, Name = model.Name, Surname = model.Surname };
        var result = await _userManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await _signInManager.SignInAsync(user, false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError(string.Empty, error.Description);
            }
        }
    }
    return View(model);
```

Рисунок 6.10 – метод реєстрації користувачів

```
public async Task<IActionResult> Login(LoginViewModel model)
{
    if (ModelState.IsValid)
    {
        var result =
            await _signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);
        if (result.Succeeded)
        {
            if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.ReturnUrl))
            {
                return Redirect(model.ReturnUrl);
            }
            else
            {
                return RedirectToAction("Index", "Home");
            }
        }
        else
        {
            ModelState.AddModelError("", "Неправильний логін чи (та) пароль");
        }
    }
    return View(model);
}
```

Рисунок 6.11 – метод авторизації користувачів

Для того, щоб виводити інформацію на веб-сторінку, щоб користувачі могли додавати записи та переглядати їх – потрібно створити метод (дію контролера) і представлення, яке відповідатиме за відображення потрібної інформації. Розглянемо на прикладі NotesController на рисунках 6.12 – 6.16.

```
public class NotesController : Controller
{
    private readonly ApplicationDbContext _context;

    Ссылок: 0
    public NotesController(ApplicationDbContext context)
    {
        _context = context;
    }
    // GET: Notes
    Ссылок: 3
    public async Task<IActionResult> Index(string searchString)
    {
        ViewData["CurrentFilter"] = searchString;

        IQueryable<Note> applicationDbContext = _context.Notes.Include(n => n.Activity).Include(n => n.Mood).Include(n => n.User);

        if (!(searchString==null))
        {
            applicationDbContext = applicationDbContext.Where(s => s.noteDate == Convert.ToDateTime(searchString));
        }

        return View(await applicationDbContext.AsNoTracking().ToListAsync());
    }

    // GET: Notes/Details/5
    Ссылок: 0
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }
```

Рисунок 6.12 – NotesController 1 частина

```
        var note = await _context.Notes
            .Include(n => n.Activity)
            .Include(n => n.Mood)
            .Include(n => n.User)
            .FirstOrDefaultAsync(m => m.id == id);
        if (note == null)
        {
            return NotFound();
        }

        return View(note);
    }

    // GET: Notes/Create
    Ссылок: 0
    public IActionResult Create()
    {
        ViewData["activityId"] = new SelectList(_context.Activities, "id", "activityName");
        ViewData["moodId"] = new SelectList(_context.Moods, "id", "moodName");
        ViewData["userId"] = new SelectList(_context.Users, "Id", "Email");
        return View();
    }
```

Рисунок 6.13 – NotesController 2 частина

```
public async Task<IActionResult> Create([Bind("id,noteDate,moodId,activityId,plans,plansDone,thoughts,userId")] Note note)
{
    if (ModelState.IsValid)
    {
        _context.Add(note);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["activityId"] = new SelectList(_context.Activities, "id", "activityName", note.activityId);
    ViewData["moodId"] = new SelectList(_context.Moods, "id", "moodName", note.moodId);
    ViewData["userId"] = new SelectList(_context.Users, "Id", "Email", note.userId);
    return View(note);
}

// GET: Notes/Edit/5
Ссылок: 0
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var note = await _context.Notes.FindAsync(id);
    if (note == null)
    {
        return NotFound();
    }
    ViewData["activityId"] = new SelectList(_context.Activities, "id", "activityName", note.activityId);
    ViewData["moodId"] = new SelectList(_context.Moods, "id", "moodName", note.moodId);
    ViewData["userId"] = new SelectList(_context.Users, "Id", "Email", note.userId);
    return View(note);
```

Рисунок 6.14 – NotesController 3 частина

```
Ссылок: 0
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var note = await _context.Notes.FindAsync(id);
    if (note == null)
    {
        return NotFound();
    }
    ViewData["activityId"] = new SelectList(_context.Activities, "id", "activityName", note.activityId);
    ViewData["moodId"] = new SelectList(_context.Moods, "id", "moodName", note.moodId);
    ViewData["userId"] = new SelectList(_context.Users, "Id", "Email", note.userId);
    return View(note);
}

// POST: Notes/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to, for
// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
Ссылок: 0
public async Task<IActionResult> Edit(int id, [Bind("id,noteDate,moodId,activityId,plans,plansDone,thoughts,userId")] Note note)
{
    if (id != note.id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
```

Рисунок 6.15 – NotesController 4 частина

```
    var note = await _context.Notes
        .Include(n => n.Activity)
        .Include(n => n.Mood)
        .Include(n => n.User)
        .FirstOrDefaultAsync(m => m.id == id);
    if (note == null)
    {
        return NotFound();
    }

    return View(note);
}

// POST: Notes/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
Ссылок: 0
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var note = await _context.Notes.FindAsync(id);
    _context.Notes.Remove(note);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

ссылка: 1
private bool NoteExists(int id)
{
    return _context.Notes.Any(e => e.id == id);
}
```

Рисунок 6.16 – NotesController 5 частина

Для зручності використання додатку були реалізовані пошук дописів за датою та посторінкове відображення записів. Для реалізації пошуку за датою були додані зміни в контролері NotesController (рисунок 6.17) та у представленні сторінки дописів (рисунок 6.18).

```
ViewData["CurrentFilter"] = searchString;

IQueryable<Note> applicationDbContext = _context.Notes.Include(n => n.Activity).Include(n => n.Mood).Include(n => n.User)

if (!(searchString==null))
{
    applicationDbContext = applicationDbContext.Where(s => s.noteDate == Convert.ToDateTime(searchString));
}
```

Рисунок 6.17 – пошук за датою NotesController

```
<form asp-action="Index" method="get">
    <div class="form-actions no-color">
        <p>
            Пошук за датою: <input type="text" name="SearchString" value="@ViewData["CurrentFilter"]" />
            <input type="submit" value="Пошук" class="btn btn-primary" /> |
            <a asp-action="Index" class="btn btn-secondary">Показати усі записи</a>
        </p>
    </div>
</form>
```

Рисунок 6.18 – пошук за датою у представлені Index

Щоб додати посторінковий перегляд до сторінки перегляду записів користувача, створюємо клас PaginatedList, який використовує оператори Skip і Take для фільтрації даних на сервері замість того, щоб завжди отримувати всі рядки таблиці (рисунок 6.19).

```
public class PaginatedList<T> : List<T>
{

    public int PageIndex { get; private set; }

    public int TotalPages { get; private set; }


    public PaginatedList(List<T> items, int count, int pageIndex, int pageSize)
    {
        PageIndex = pageIndex;
        TotalPages = (int)Math.Ceiling(count / (double)pageSize);

        this.AddRange(items);
    }


    public bool HasPreviousPage => PageIndex > 1;


    public bool HasNextPage => PageIndex < TotalPages;


    public static async Task<PaginatedList<T>> CreateAsync(IQueryable<T> source, int pageIndex, int pageSize)
    {
        var count = await source.CountAsync();
        var items = await source.Skip((pageIndex - 1) * pageSize).Take(pageSize).ToListAsync();
        return new PaginatedList<T>(items, count, pageIndex, pageSize);
    }
```

Рисунок 6.19 – клас PaginatedList

Потім внесимо додаткові зміни в метод Index у контролері (рисунок 6.20) і додаємо кнопки сторінок до представлення Index (рисунок 6.21).

```
public async Task<IActionResult> Index(string searchString, int? pageNumber, string currentFilter)
{
    ViewData["CurrentFilter"] = searchString;

    IQueryable<Note> applicationDbContext = _context.Notes.Include(n => n.Activity).Include(n => n.Moo

    if (!(searchString==null))
    {
        applicationDbContext = applicationDbContext.Where(s => s.noteDate == Convert.ToDateTime(search
    }

    if (searchString != null)
    {
        pageNumber = 1;
    }
    else
    {
        searchString = currentFilter;
    }
```

Рисунок 6.20 – посторінкове відображення NotesController

```
@{
    var prevDisabled = !Model.HasPreviousPage ? "disabled" : "";
    var nextDisabled = !Model.HasNextPage ? "disabled" : "";
}

<a asp-action="Index"
   asp-route-sortOrder="@ViewData["CurrentSort"]"
   asp-route-pageNumber="@(Model.PageIndex - 1)"
   asp-route-currentFilter="@ViewData["CurrentFilter"]"
   class="btn btn-default @prevDisabled">
    Назад
</a>
<a asp-action="Index"
   asp-route-sortOrder="@ViewData["CurrentSort"]"
   asp-route-pageNumber="@(Model.PageIndex + 1)"
   asp-route-currentFilter="@ViewData["CurrentFilter"]"
   class="btn btn-default @nextDisabled">
    Далі
</a>
```

Рисунок 6.21 – посторінкове відображення у представлені Index

Для самоаналізу власного ментального здоров’я було створено діаграми для аналізу зроблених записів користувачем. Було використано бібліотеку Google Charts. Дані ми будемо передавати зі сторони клієнта. Створюємо API контролер ChartsController і реалізовуємо потрібні методи (рисунок 6.22-6.23).

```
public class ChartsController : ControllerBase
{
    private readonly ApplicationDbContext _context;
    Ссылок: 0
    public ChartsController(ApplicationDbContext context)
    {
        _context = context;
    }
    [HttpGet("JsonData")]
    Ссылок: 0
    public JsonResult JsonData()
    {
        var moods = _context.Moods.Include(b => b.Notes).ToList();
        List<object> moodNotes = new List<object>();
        moodNotes.Add(new[] { "Настрій", "Кількість записів" });
        foreach (var c in moods)
        {
            moodNotes.Add(new object[] { c.moodName, c.Notes.Count() });
        }
        return new JsonResult(moodNotes);
```

Рисунок 6.22 – ChartsController 1 частина

```
[HttpGet("JsonData1")]
Ссылок: 0
public JsonResult JsonData1()
{
    var activities = _context.Activities.Include(b=>b.Notes).ToList();
    List<object> activityNotes = new List<object>();
    activityNotes.Add(new[] { "Активності", "Кількість записів" });
    foreach (var c in activities)
    {
        activityNotes.Add(new object[] { c.activityName, c.Notes.Count()});
    }
    return new JsonResult(activityNotes);
}
```

Рисунок 6.23 – ChartsController 2 частина

Для відображення діаграм на сторінці було змінено представлення Index – написані відповідні скрипти (рисунок 6.24).

```
<script type="text/javascript" src="https://gstatic.com/charts/loader.js"></script>
<script>
    google.charts.load('current', { 'packages': ['corechart'] });
    google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
        $.get('/api/Charts/JsonData', function (jsonData) {
            data = google.visualization.arrayToDataTable(jsonData, false);
            var option = {
                title: "Статистика настроїв за місяць",
                width: 400,
                height: 300
            };
            chart = new google.visualization.PieChart(document.getElementById('chart1'));
            chart.draw(data, option);
        })
    }
</script>

<script>
    google.charts.load('current', { 'packages': ['corechart'] });
    google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
        $.get('/api/Charts/JsonData1', function (jsonData1) {
            data = google.visualization.arrayToDataTable(jsonData1, false);
            var option = {
                title: "Статистика активностей за місяць",
                width: 400,
                height: 300
            };
            chart = new google.visualization.PieChart(document.getElementById('chart2'));
            chart.draw(data, option);
        })
    }
```

Рисунок 6.25 – скрипт для відображення діаграм представлення Index

РОЗДІЛ 7 ІНСТРУКЦІЯ КОРИСТУВАЧА

Починаючи роботу з додатком, користувач повинен зареєструватися в системі (рисунок 7.1), або, якщо він вже зареєстрований – авторизуватися (рисунок 7.2).

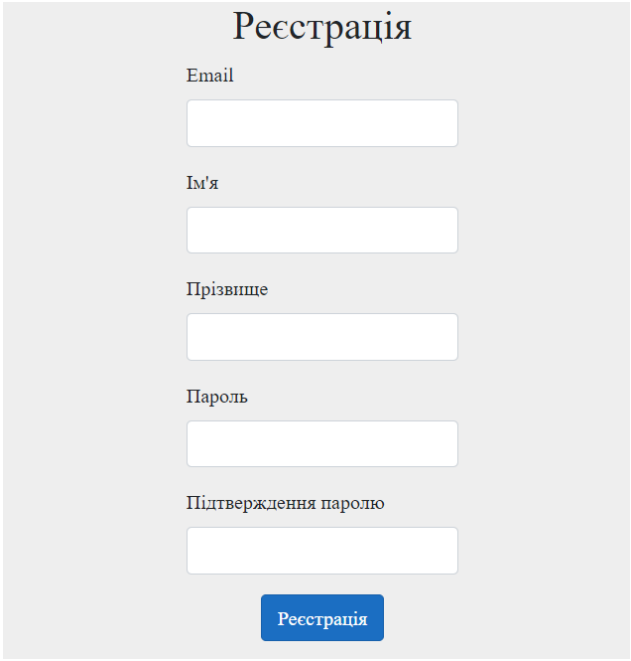

Рисунок 7.1 – Вікно реєстрації

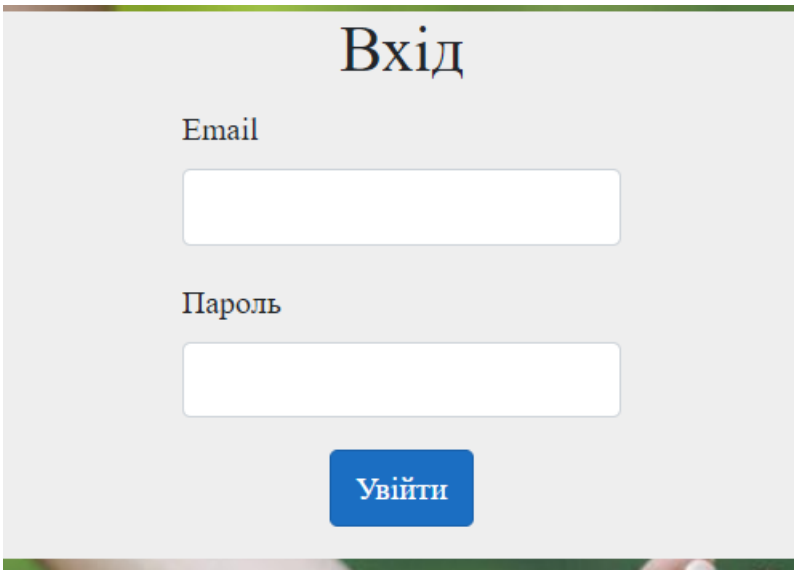

Рисунок 7.2 – Вікно авторизації

Увійшовши в свій акаунт, користувач може додавати, змінювати, або редагувати свої записи (рисунок 7.3).

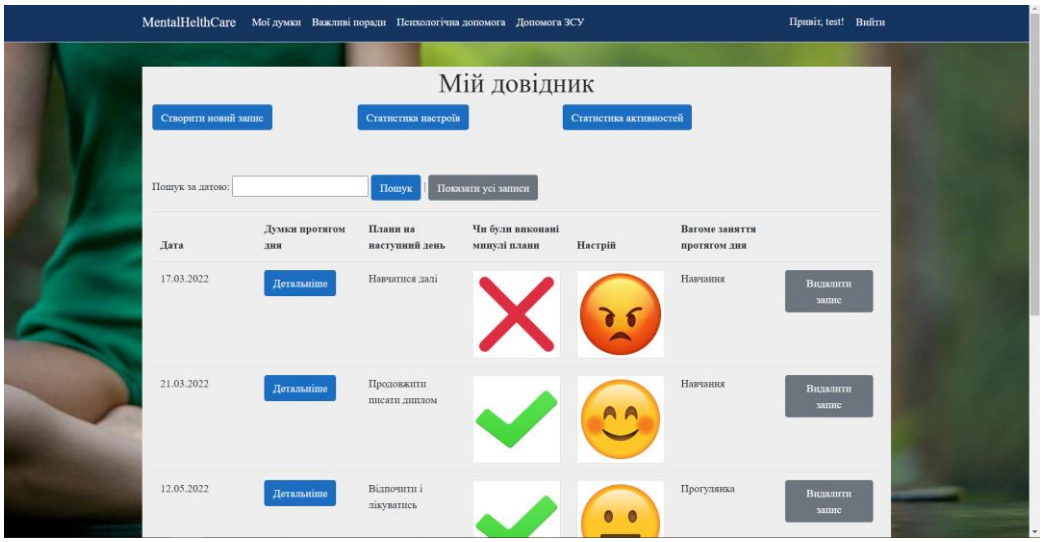

Рисунок 7.3 – Сторінка «Мої думки»

Користувачу доступна статистика своїх записів, а саме: статистика настроїв та активностей (рисунок 7.4).

Рисунок 7.4 – статистика записів користувача

Якщо при створенні нового запису користувач викладе думки, які можуть нанести шкоду власному здоров’ю, то система попередить користувача (рисунок 7.5) та надасть усі контакти для отримання психологічної допомоги (рисунок 7.6).

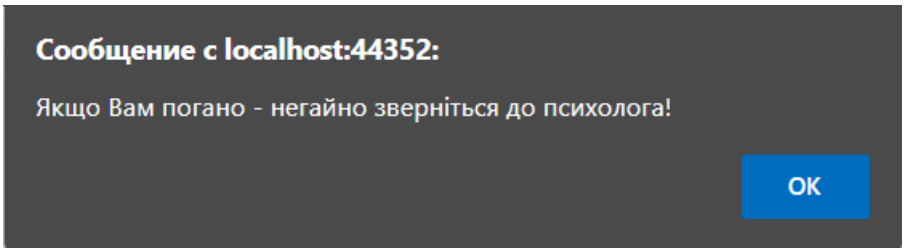

Рисунок 7.5 – Попередження системи

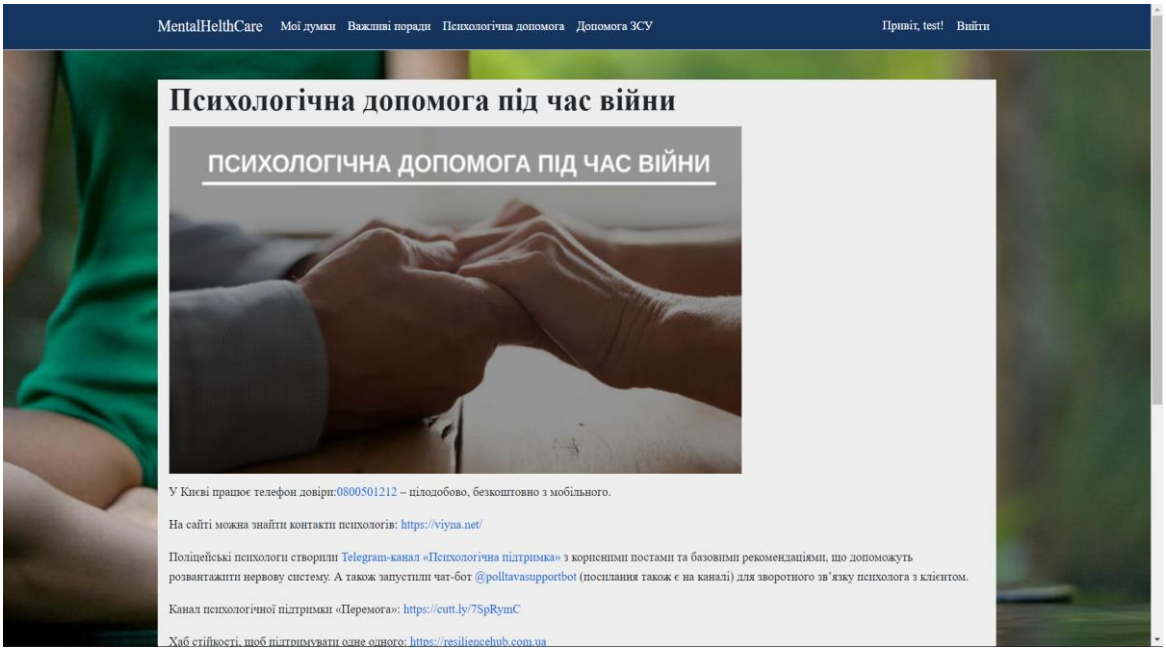

Рисунок 7.6 – сторінка «Психологічна допомога»

ВИСНОВКИ

В ході виконання кваліфікаційної роботи розглянуто питання, пов’язані з необхідністю розробки застосунку для запису та самоаналізу власних емоцій, а також описано процес розробки вищезгаданого застосунку.

Після аналізу предметної області даної системи та її компонентів визначено постановку задачі та основні вимоги, яких необхідно дотриматись в процесі реалізації.

Протягом виконання роботи було досліджено та закріплено знання за необхідними для роботи технологіями, а також покращено уміння в роботі з ASP.NET Core, а також вивчено методи конструювання простих інформаційних систем.

В процесі виконання кваліфікаційної роботи автором було реалізовано застосунок «MentalHealthCare». Було розроблено зручний та зрозумілий інтерфейс користування застосунком.

Після здійснення поставлених задач було досягнуто мети виконання кваліфікаційної роботи – розроблено застосунок для запису та самоаналізу власних емоцій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Social Psychology Network [Електронний ресурс] – Режим доступу до ресурсу: https://pennebaker.socialpsychology.org/.
2. Dr Alice Boyes [Електронний ресурс] – Режим доступу до ресурсу: https://www.aliceboyes.com/
3. The Healthy Mind Toolkit [Електронний ресурс] – Режим доступу до ресурсу: http://healthymindtoolkit.com/.
4. ASP.NET documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0.
5. Microsoft SQL Server documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15.
6. JavaScript documentation [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/javascript/
7. Visual Studio Community 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/
8. Emotional and physical health benefits of expressive writing [Електронний ресурс] – Режим доступу до ресурсу: https://www.cambridge.org/core/journals/advances-in-psychiatric-treatment/article/emotional-and-physical-health-benefits-of-expressive-writing/ED2976A61F5DE56B46F07A1CE9EA9F9F#B3/

9. Writing Can Help Injuries Heal Faster [Електронний ресурс] – Режим доступу до ресурсу: https://www.scientificamerican.com/article/writing-can-help-injuries-heal-faster/
10. Using Diary Methods in Psychological Research [Електронний ресурс]- Режим доступу до ресурсу: http://www.columbia.edu/~nb2229/docs/Iida,%20Shrout,%20Laurenceau%20&%20Bolger%20%282012%29.pdf
11. The Model View Controller Pattern [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/
12. SQL Server 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019
13. Overview to ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0
14. Емоційний стан під час війни [Електронний ресурс] – Режим доступу до ресурсу: https://forbes.ua/inside/pid-chas-viyni-bagato-ukraintsiv-stikayutsya-z-perepadami-emotsiy-psikhologinya-poyasnyue-yak-navchitis-keruvati-emotsiyami-ta-pereyti-do-etapu-adaptatsii-06032022-4273
15. Контроль емоційного стану [Електронний ресурс] – Режим доступу до ресурсу: https://mon.gov.ua/ua/news/kontrol-emocijnogo-stanu-pid-chas-vijni-poyasnennya-psihologiv