МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ВЕБЗАСТОСУНОК ДЛЯ ПСИХОЛОГІЧНОЇ ПІДТРІМКИ ПІД ЧАС ВІЙНИ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21910206

Виконала студентка 4-го курсу, групи 402
_____ А. В. Гончаренко
«19» червня 2023 р.

Керівник: канд. техн. наук, доцент
_____Г. В. Горбань
«19» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«23» листопада 2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студентці групи 402 факультету комп'ютерних наук Гончаренко Аліні Володимирівні.

1. Тема кваліфікаційної роботи «Вебзастосунок для психологічної підтримки під час війни».

Керівник роботи: Горбань Гліб Валентинович, канд. техн. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «26» березня 2023 р. № 59

2. Строк представлення кваліфікаційної роботи студентом «28» червня 2023 р.

3. Вхідні (початкові) дані до роботи: розроблений вебзастосунок для психологічної підтримки під час війни, для реалізації знадобилось розробити фронтенд та серверну частину застосунку, також мовою для реалізації поставленої мети було обрано AngularJS, як фронтенд, та node.js, як мову для відповідник серверної частини.

Очікуваний результат: вебзастосунок для психологічної підтримки під час війни.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– дослідити об’єктну та предметну галузь;

– проаналізувати аналоги, що вже існують;

– розробити веб застосунок для психологічної підтримки під час війни;

– створити користувацький інтерфейс;

– розробка структури бази даних;

– проєктування програмних інтерфейсів для взаємодії з серверною частиною.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Оцінка умов праці».

7. Консультанти розділів роботи.

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Боженко А. Л., викладач.	

Керівник роботи __канд. техн. наук, доцент. Горбань Г. В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Гончаренко А. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Вебзастосунок для психологічної підтримки під час війни

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження	26.10.2022	30.10.2022	
2	Отримання завдання на виконання БКР	25.11.2022	25.11.2022	
3	Складання календарного плану роботи на весь період виконання БКР	08.12.2022	10.12.2022	
4	Отримання завдання на переддипломну практику	01.05.2023	01.05.2023	
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	
7	Виконання БКР: аналіз сучасного стану задачі відслідковування очей, огляд існуючих технологій, розробка ПЗ	15.05.2023	19.06.2023	
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	
10	Подання БКР рецензенту	15.06.2023	17.06.2023	
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробив студент __Гончаренко А. В._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __ канд. техн. наук, доцент. Горбань Г. В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«10» _____грудня_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студентки групи 402 ЧНУ ім. Петра Могили

Гончаренко Аліни Володимирівни

Тема: «Вебзастосунок для психологічної підтримки під час війни»

Актуальність бакалаврської роботи полягає в розробці вебзастосунку для психологічної підтримки під час війни, який може бути корисним інструментом для людей, які переживають емоційний стрес у зв'язку з воєнним конфліктом. Він надає різноманітні можливості для отримання психологічної підтримки, навчання корисним стратегіям подолання стресу та отримання додаткової допомоги у випадках потреби.

Об’єктом кваліфікаційної роботи є психологічна онлайн підтримка під час війни.

Предметом кваліфікаційної роботи є технології розробки вебзастосунку психологічної підтримки.

Метою кваліфікаційної роботи є підвищення ефективності надання психологічної підтримки шляхом зменшення часу організацій психологічних консультацій у вебзастосунку підтримки під час війни.

Пояснювальна записка бакалаврської роботи складається зі вступу, чотирьох розділів, висновків та двух додатків. У вступі визначається актуальність теми, сформульовані мета, об’єкт, предмет та завдання бакалаврської роботи.

У першому розділі було проаналізовано предметну сферу були досліджені технології, які можуть бути використані для створення вебзастосунку.

У другому розділі було обрано технології для реалізації вебзастосунку.

У третьому розділі наведені вхідні данні та структури системи та проєктування інформаційної систем.

У четвертому розділі були проведена програмна реалізація та розробка документації, та інформаційні діаграми. В спеціальній частині. В спеціальній частині присвячений охороні праці на підприємстві. У висновках наведено аналіз виконаної роботи та отриманих результатів дослідження та розроблення.

Бакалаврська кваліфікаційна робота містить 76 сторінок, 39 рисунків, 1 таблицю, 23 використаних джерел та 2 додатків.

Ключові слова: AngularJS, MongoDB, Restfulapi, node.js, , психологічна підтримка.

ABSTRACT

for bachelor’s qualification work of a student of 402 group at Petro Mohyla Black Sea National University

Honcharenko Alina Volodymyrivna

Topic: "Web application for psychological support during war"

The relevance of the bachelor's work lies in the development of a web application for psychological support during war, which can be a useful tool for individuals experiencing emotional stress due to armed conflict. It provides various opportunities for receiving psychological support, learning helpful stress coping strategies, and accessing additional assistance when needed.

The object of the qualification work is online psychological support during the war.

The subject of the qualification work is the technology of developing a psychological support web application.

The goal of the qualification work is to increase the effectiveness of providing psychological support by reducing the time for organizing psychological consultations in the web application for support during war.

The explanatory note of the bachelor's work consists of an introduction, four chapters, conclusions, and two appendices. The introduction defines the relevance of the topic and formulates the aim, object, subject, and tasks of the bachelor's work.

The first chapter analyzed the subject area and examined the technologies that can be used for creating the web application.

The second chapter selected the technologies for implementing the web application.

The third chapter presents the input data and system structures, as well as the design of the information system.

The fourth chapter involved the software implementation and development of documentation and information diagrams. The special section is dedicated to

occupational health and safety in the enterprise. The conclusions provide an analysis of the work performed and the results obtained from research and development.

The bachelor's qualification work consists of 76 pages, 39 figures, 1 table, 23 references, and 2 appendices.

Keywords: AngularJS, MongoDB, Restful API, node.js, psychological support.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	–	Application Programming Interface
JS	–	JavaScript
DI	–	Dependency Injection
CLI	–	Command-Line Interface
NPM	–	Node Package Manager
TS	–	TypeScript

ВСТУП

Війна виснажує нас фізично і психологічно. Незважаючи на те, що наша психіка здатна адаптуватися під будь-які складності, кожен день бойових дій в країні пережити дуже складно. В цей непростий час слід підтримувати себе, рідних і близьких. Це життєво необхідно для того, щоб зберегти здоров'я – фізичне і психологічне.

Під час війни для психологічної підтримки можуть використовувати різноманітні технології, які допомагають покращити самопочуття та забезпечити доступ до необхідної допомоги.

Розробка вебзастосунку для психологічної підтримки під час війни є важливою та цінною ініціативою, оскільки вона може допомогти забезпечити доступ до необхідної психологічної підтримки тим, хто перебуває у військових конфліктах.

Актуальність кваліфікаційної роботи полягає в розробці вебзастосунку для психологічної підтримки під час війни, який може бути корисним інструментом для людей, які переживають емоційний стрес у зв'язку з воєнним конфліктом. Він надає різноманітні можливості для отримання психологічної підтримки, навчання корисним стратегіям подолання стресу та отримання додаткової допомоги у випадках потреби.

Об’єктом кваліфікаційної роботи є психологічна онлайн підтримка під час війни.

Предметом кваліфікаційної роботи є технології розробки вебзастосунку психологічної підтримки.

Метою кваліфікаційної роботи є підвищення ефективності психологічної підтримки шляхом зменшення часу організацій психологічних консультацій у вебзастосунку підтримки під час війни.

Завдання для досягнення поставленої мети:

– дослідити об’єктну та предметну галузь;
– проаналізувати аналоги, що вже існують;

– розробити вебзастосунок для психологічної підтримки під час війни;

– створити користувацький інтерфейс;

– розробка структури бази даних;

– проєктування програмних інтерфейсів для взаємодії з серверною частиною.

1. АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Опис предметної сфери

Вебзастосунок для психологічної підтримки під час війни - це онлайн-платформа, розроблена для допомоги людям, які переживають емоційні труднощі та психологічний стрес у зв'язку з війною. Враховуючи складну природу воєнного конфлікту, такий вебзастосунок має спеціальні функції та ресурси, спрямовані на підтримку користувачів у цих умовах. Основні елементи вебзастосунку для психологічної підтримки під час війни включають [1]:

– консультування та підтримка: користувачі мають можливість звернутися до професійних психологів або кваліфікованих консультантів для отримання конфіденційної підтримки та ради. Це може включати онлайн-консультування через відеозв'язок або повідомлення;

– психологічні ресурси: вебзастосунок може містити багато ресурсів, таких як статті, пам'ятки, інформація про стратегії подолання стресу, методи релаксації та інші корисні матеріали. Це допомагає користувачам отримати необхідну інформацію та навички для керування своїм емоційним станом;

– групова підтримка: платформа може надавати можливість приєднатися до онлайн-спільноти або форуму, де люди, які переживають подібні ситуації, можуть обмінюватися досвідом, надавати підтримку одне одному та спілкуватися;

– психологічні вправи та стратегії: вебзастосунок може містити різноманітні психологічні вправи, методики та стратегії, що допомагають користувачам керувати своїми емоціями та подолати стрес. Наприклад, це може бути медитація, дихальні вправи, психологічні ігри та інші методи;

– підтримка з боку соціальних служб: вебзастосунок може мати функцію підключення до соціальних служб для надання додаткової підтримки

у випадках, коли користувачі потребують більш серйозної допомоги, наприклад, у випадках післятравматичного стресу або депресії;

– безпека та конфіденційність: оскільки платформа забезпечує психологічну підтримку, безпека та конфіденційність є важливими аспектами розробки такого вебзастосунку. Для цього можуть використовуватися різні заходи безпеки, такі як шифрування даних, резервне копіювання даних та інші методи захисту;

– повідомлення та нагадування: вебзастосунок може надавати користувачам підтримку у вигляді повідомлень та нагадувань. Наприклад, користувачі можуть отримувати повідомлення з психологічними порадами або нагадування про важливі події.

Вебзастосунок для психологічної підтримки під час війни може бути корисним інструментом для людей, які переживають емоційний стрес у зв'язку з воєнним конфліктом. Він надає різноманітні можливості для отримання психологічної підтримки, навчання корисним стратегіям подолання стресу та отримання додаткової допомоги у випадках потреби.

1.2 Аналіз існуючих рішень для реалізації проєкту

AngularJS — це набір інструментів для створення фреймворку, який найбільше підходить для розробки ваших програм. Він повністю розширюваний і добре працює з іншими бібліотеками. Кожну функцію можна змінити або замінити відповідно до вашого унікального робочого процесу розробки та потреб функцій.

Незважаючи на те, що HTML ефективний для створення статичних документів, він недостатній, коли справа доходить до визначення динамічних переглядів у вебзастосунках. Однак AngularJS дозволяє покращити мову HTML спеціально для вашої програми. Це призводить до надзвичайно потужного, легкого для розуміння та ефективного середовища розробки.

Інші фреймворки усувають недоліки HTML, абстрагуючись від HTML, CSS і/або JavaScript, або забезпечуючи обов’язковий спосіб маніпулювання DOM. Жоден із них не вирішує кореневої проблеми, що HTML не був розроблений для динамічних переглядів [2].

Переваги платформи:

– легка початкова точка входу;
– обширний вибір бібліотек;
– інтуїтивний інтерфейс;
– підтримка різних платформ;
– зручне використання фізичних обчислень;
– безкоштовна доступність.

Недоліки:

– великі розміри: AngularJS вимагає завантаження великих бібліотек, що може призвести до збільшення часу завантаження сторінок та споживання більше мережевого трафіку;
– складність вивчення: вивчення AngularJS може бути викликом для новачків. Фреймворк має значну кількість концепцій, директив та власної відмінної від звичайного підходу до розробки вебзастосунків;
– потреба у підтримці старших версій: зазвичай, коли нова версія Angular виходить, підтримка старих версій швидко зменшується. Це може створити проблеми для вебзастосунків, розроблених на попередніх версіях AngularJS;
– важкість тестування: AngularJS має вбудовану систему залежностей та складний життєвий цикл компонентів, що може ускладнити процес тестування та зміну взаємодій між компонентами;
– обмежена підтримка мобільних платформ: оригінальний AngularJS не має нативної підтримки для розробки мобільних застосунків, що

може вимагати додаткових зусиль для створення перехідних рішень для підтримки мобільних платформ.

Варто враховувати ці недоліки при виборі AngularJS для розробки вебзастосунків і зважити на вимоги та потреби вашого проєкту.

Node.js - це відкрита середовище виконання JavaScript на стороні сервера, яке дозволяє розробникам створювати вебзастосунки та мережеві застосунки. Воно базується на двигуні V8 JavaScript, розробленому компанією Google, і забезпечує високу продуктивність та швидкодію [3].

Основні переваги Node.js:

– висока продуктивність: Node.js використовує асинхронну, подієву модель програмування, що дозволяє ефективно обробляти багатопотокові запити та забезпечує високу швидкодію обробки даних;

– спільна мова на фронтенді та бекенді: Node.js дозволяє використовувати JavaScript як на стороні клієнта (фронтенді), так і на стороні сервера (бекенді), що спрощує розробку та підтримку проєкту;

– розширена екосистема: Node.js має велику кількість сторонніх модулів та бібліотек, доступних через пакетний менеджер npm. Це дозволяє розробникам швидко використовувати готові рішення та розширювати функціональність своїх застосунків;

– масштабованість: Node.js добре підходить для побудови масштабованих застосунків, оскільки він може обробляти велику кількість одночасних підключень без значного збільшення витрат ресурсів;

– активна спільнота: node.js має велику та активну спільноту розробників, що забезпечує підтримку, документацію та постійний розвиток екосистеми.

Node.js знаходить застосування у різних сферах, включаючи розробку вебсерверів, API, реал-тайм застосунків, мікросервісів та іншого типу застосунків, де важливо швидко.

Node.js також має свої недоліки:

– однопоточність: node.js працює в одному потоці, що означає, що він може мати проблеми з обробкою великої кількості обчислювально інтенсивних операцій. Це може призвести до блокування іншої роботи, якщо виконується довготривала операція;

– не підходить для CPU-інтенсивних завдань: через однопотоковий характер Node.js, він не є оптимальним вибором для виконання важких обчислювальних завдань, які вимагають використання багатьох ядер процесора;

– відсутність стандартів: оскільки Node.js є релятивно новою технологією, він не має однозначних стандартів розробки. Це може призвести до розпливчатості коду і виникнення проблем з сумісністю між різними модулями та бібліотеками;

– складність управління станом застосунку: за замовчуванням Node.js не надає засобів для управління станом застосунку. Це може призвести до складнощів у веденні стану застосунку та синхронізації даних між клієнтом та сервером;

– потенційні проблеми зі обробкою: у деяких випадках можуть виникати проблеми зі обробкою Node.js застосунків на багатоядерних серверах через однопотоковий характер платформи.

Ці недоліки не означають, що Node.js не є потужним інструментом, але вони мають бути враховані при виборі платформи для конкретного проєкту.

При порівнянні Angular з іншими рішеннями для розробки вебзастосунку треба розуміти що не всі платформи мають достатній розвиток архітектури.

React є популярним вибором для розробки веб-додатків, включаючи психологічну підтримку. Він має широку спільноту розробників та багато готових компонентів, що полегшує розробку інтерактивних інтерфейсів.

Однак, в порівнянні з Angular, React може потребувати додаткових бібліотек та рішень для реалізації функцій, таких як маршрутизація або керування станом додатку.

Vue.js є легким, простим у вивченні та швидким фреймворком. Він підходить для швидкої розробки прототипів або менших проектів психологічної підтримки. Завдяки простоті синтаксису та віртуального DOM, Vue.js може забезпечувати швидку відповідь користувачам. Проте, у порівнянні з Angular, Vue.js може мати меншу спільноту розробників та менше кількість готових рішень для психологічної підтримки.

Ember.js є повним та комплексним фреймворком, який пропонує структуровану розробку та вбудовані рішення для багатьох аспектів веб-застосунків. Він може бути корисним для великих та складних проектів психологічної підтримки, де потрібно дотримуватись певних конвенцій та стандартів розробки. Однак, у порівнянні з Angular, Ember.js може мати вищий поріг входження та потребувати більше часу для вивчення та оволодіння.

Вибір між Angular та іншими рішеннями залежить від потреб проекту, рівня досвіду розробників та вимог до продуктивності та масштабованості. Ураховуючи особливості кожного фреймворка та потреби проекту, можна зробити інформоване рішення щодо вибору технології для веб-застосунку психологічної підтримки під час війни.

1.3 Оцінка наявних варіантів рішень

Існує багато технологій, які використовуються в розробці вебзастосунків для психологічної підтримки під час війни. Нижче наведено опис та аналіз кількох таких технологій:

– чат-боти: чат-боти є популярними в інтерактивній комунікації з користувачами. Вони можуть надавати психологічну підтримку, відповідати на запитання та надавати поради. Чат-боти можуть бути програмовані з

використанням штучного інтелекту, що дозволяє їм адаптуватися до потреб користувача та забезпечувати персоналізовану підтримку;

– відеоконференції та онлайн-терапія: застосування відеоконференцій та онлайн-терапії дозволяє психологам взаємодіяти з клієнтами в режимі реального часу, навіть на віддаленій відстані. Це зручно та ефективно для осіб, які знаходяться в зоні конфлікту або не мають доступу до традиційних форм психологічної підтримки;

– мобільні застосунки: розробка мобільних застосунків для психологічної підтримки стала популярною. Ці застосунків можуть включати функції, такі як медитація, релаксація, трекінг настрою та журналування. Вони дозволяють користувачам отримувати доступ до психологічних інструментів та ресурсів прямо на своїх мобільних пристроях;

– вебплатформи спільноти: вебплатформи спільноти створюють простір для взаємодії та підтримки між людьми, які переживають війну.

Чат-боти - це програмні агенти, які можуть автоматично взаємодіяти з користувачами через текстові повідомлення [4]. Вони можуть бути реалізовані за допомогою різних технологій та інструментів.

Рисунок 1.1 – Приклад дизайну чат-бота

Відеоконференції та онлайн-терапія: застосування відеоконференцій та онлайн-терапії дозволяє психологам взаємодіяти з клієнтами в режимі реального часу, навіть на віддаленій відстані [5]. Це зручно та ефективно для осіб, які знаходяться в зоні бойових дій.

Мобільні застосунки: розробка мобільних застосунків для психологічної підтримки стала популярною [6]. Ці застосунки можуть включати функції, такі як медитація, релаксація, трекінг настрою та журналування. Вони дозволяють користувачам отримувати доступ до психологічних інструментів та ресурсів прямо на своїх мобільних пристроях.

Приклади мобільних застосунків:

1) **BetterHelp**

Через BetterHelp користувачі можуть отримати прямий доступ до професійної терапії з ліцензованим терапевтом [7].

Ця інноваційна платформа психічного здоров’я надає споживачам доступ до гнучких онлайн-сервісів психічного здоров’я, що робить це лікування більш гнучким і легкодоступним, ніж будь-коли раніше.

Користувачі можуть вибрати, чи отримувати доступ до онлайн-консультацій або терапевтичних послуг через відеочат, телефонні дзвінки або виключно текстове спілкування (залежно від їхніх особистих уподобань).

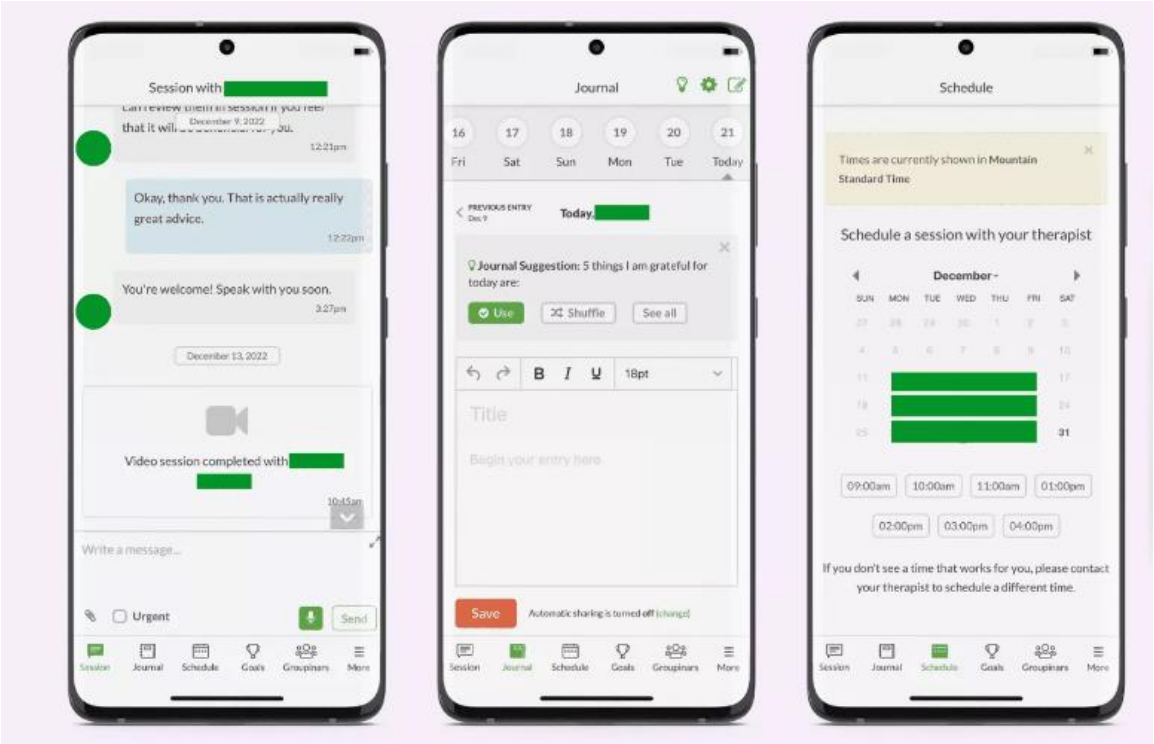

Рисунок 1.2 – Інтерфейс користувача Better Help

Наразі BetterHelp є найбільшою у світі терапевтичною службою, яка надає понад 290 мільйонів повідомлень, чатів, телефонних та відеосеансів, проведених командою з понад 30 000 ліцензованих терапевтів, які допомагають понад 3,8 мільйонам людей.

2) **Calm**

Calm є провідним застосунком для сну та медитації, лише в магазині застосунків Apple має понад 1,5 мільйона п’ятизіркових відгуків [8].

Його основна мета – допомогти користувачам покращити своє здоров’я та щастя. І це досягається за допомогою широкого спектру різних інструментів і послуг, у тому числі спеціалізованих рішень, спрямованих на покращення якості сну, зменшення стресу та тривоги та покращення концентрації уваги.

Рисунок 1.3 – Інтерфейс Calm

Користувачі програми можуть покращити свої навички медитації, отримати доступ до ексклюзивної музики для зосередження або релаксації, навчитися усвідомленим рухам і розтягуванням, а також стежити за аудіопрограмами, які викладають експерти з уважності – і все це через єдину комплексну платформу.

3) **Headspace**

Headspace – це програма для медитації, яка швидко зростає, глобальна база користувачів нараховує понад 70 мільйонів учасників у 190 країнах світу [9].

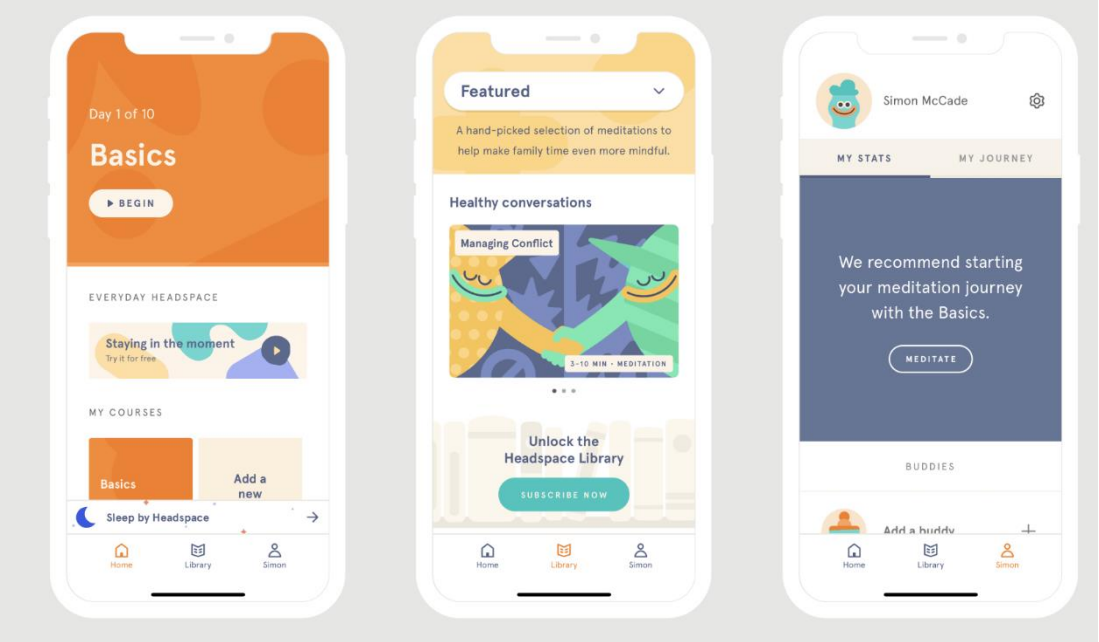

Рисунок 1.4 – Користувацький інтерфейс Headspace

Застосунок пропонує науково обґрунтовані інструменти для медитації та уважності, а також сотні варіантів керованої медитації, включаючи спеціальну музику, медитацію під час сну та курси, спрямовані на керування тривогою. Фактично доведено, що він зменшує стрес на 14% лише за 10 днів.

4) **WYSA**

Пропонуючи подібну послугу до IESO, WYSA є ще однією програмою, яка пропонує гнучкий віддалений інструмент підтримки психічного здоров'я [10].

Але він унікальний у використанні клінічно підтвердженого розмовного ШІ. Платформа спеціалізується на допомозі організаціям, підприємствам і роботодавцям покращити підтримку психічного здоров'я, яку отримують їхні команди.

Ця платформа охорони здоров'я зі штучним інтелектом провела понад півмільярда розмов із понад п'ятьма мільйонами людей про їх психічне здоров'я в 95 країнах.

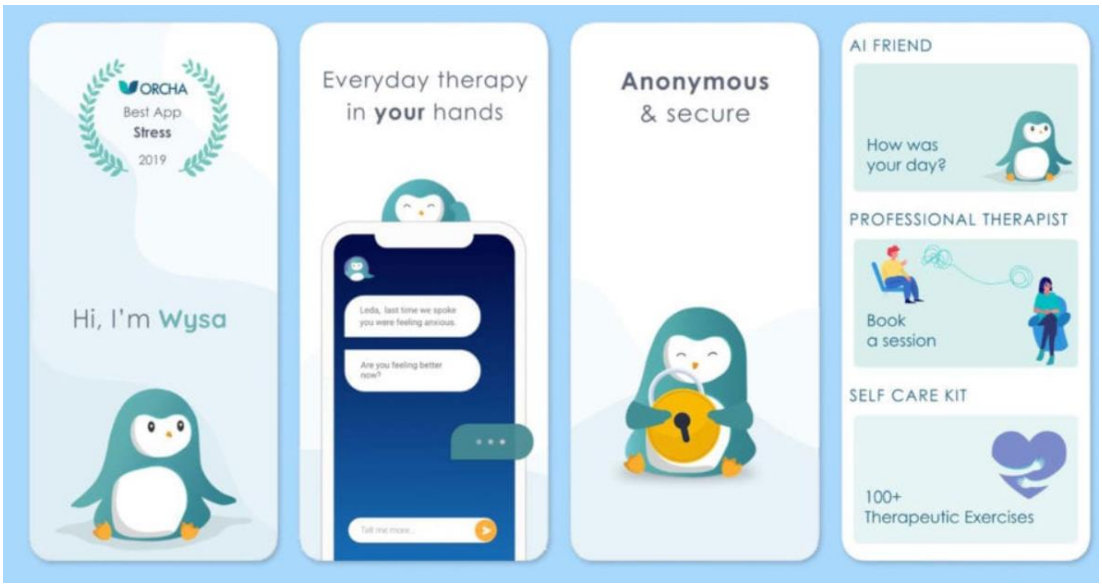

Рисунок 1.5 – Інтерфейс користувача Wysa

Його партнерами є BOSCH, Колумбійський університет, Гарвардська медична школа, L’Oreal і NHS. Крім того, він отримав численні галузеві нагороди, включно з 5 найкращими інноваціями у сфері психічного здоров’я за версією Forbes за 2020 рік, найкращим застосунком Google Play і визнанням 10 найкращих інноваторів Всесвітнього економічного форуму 2022 року.

Нижче наведено переваги та недоліки кожного з розглянутих застосунків для психологічної підтримки: Wysa, Headspace, BetterHelp та Calm.

Wysa, переваги:

– анонімність: Wysa дозволяє користувачам отримати психологічну підтримку, залишаючись анонімними. Це може бути корисним для тих, хто хоче зберегти конфіденційність своїх проблем і розмов;

– широкий спектр підтримки: Wysa надає доступ до різних інструментів та технік, таких як медитація, когнітивно-поведінкова терапія та позитивна психологія, які можуть допомогти управляти стресом, тривогами та іншими проблемами;

– цілодобовий доступ: Wysa доступний 24/7, що означає, що користувачі можуть отримати підтримку та поради у будь-який час, коли вони цього потребують.

Недоліки:

– відсутність особистого контакту: Wysa є чат-ботом, що означає, що взаємодія з ним буде відсутньою людського фактору, який може бути важливим для деяких користувачів.

Headspace, переваги:

– медитація та управління стресом: Headspace спеціалізується на навчанні медитації та технік управління стресом. Це може бути корисно для тих, хто шукає способи заспокоїтися та зосередитися;

– якість вмісту: Headspace має високоякісний вміст, включаючи голосові провідники, музику та аудіо-курси, які допомагають користувачам зосередитися, заспокоїтися та розслабитися;

– доступність на різних платформах: Headspace доступний на різних платформах, включаючи веб, мобільні пристрої та годинники, що дозволяє користувачам отримувати підтримку та практикувати медитацію у зручний для них спосіб.

Недоліки:

– обмежений фокус: Headspace сконцентрований переважно на медитації та управлінні стресом. Він може не надавати таку широку гаму психологічної підтримки, як би цього хотіли деякі користувачі;

– вартість: хоча Headspace має безкоштовний варіант, повний доступ до всіх його функцій вимагає підписки, яка може бути вартою для деяких користувачів.

BetterHelp, переваги:

– онлайн-терапія: BetterHelp надає можливість отримати онлайн-терапію з ліцензованими психологами та терапевтами. Це дозволяє зручно отримувати індивідуальну підтримку та поради;

– гнучкість розкладу: користувачі можуть планувати сесії з терапевтами у зручний для них час і місце, що робить BetterHelp досить гнучким варіантом для тих, хто має зайнятий розклад;

– мінімальні очікування: зазвичай, отримання традиційної терапії може зайняти тривалий час, але BetterHelp може надати користувачам швидкий доступ до психологічної підтримки.

Недоліки:

– віддалена взаємодія: взаємодія з терапевтом відбувається в основному через онлайн-повідомлення або відеодзвіноки, що може бути менш особистим та ефективним порівняно зі зустрічами тет-а-тет у традиційній практиці;

– вартість: BetterHelp вимагає платну підписку на отримання послуг терапевта. Це може бути фінансово вимогливим для деяких користувачів.

Calm, переваги:

– медитація та релаксація: Calm пропонує широкий вибір медитаційних вправ, релаксаційних звуків та сеансів музичної терапії, що допомагає користувачам заспокоїтися, зосередитися та зменшити стрес;

– сну та відновлення енергії: Calm надає спеціальні засоби для поліпшення сну, такі як голосові казки та заспокійливі звуки, що допомагають заснути та отримати якісний відпочинок;

– інтерактивні програми: Calm має різноманітні програми для зменшення тривоги, покращення настрою та здоров'я загалом. Вони включають голосові вправи та виклики, що стимулюють активну участь користувачів.

Недоліки:

– обмежені можливості психотерапії: Calm не надає особистої терапії або консультування з ліцензованими фахівцями. Він більше фокусується на медитації та релаксації.

1.4 Постановка завдання

Нижче наведена загальна постановка задачі або технічна пропозиція для розробки вебзастосунку для психологічної підтримки під час війни.

Огляд проєкту:

– розробка вебзастосунку, який надає психологічну підтримку користувачам, які перебувають у військових конфліктах або їх наслідках [11];

– забезпечення доступу до інструментів, ресурсів та інформації, які допоможуть користувачам знизити стрес, поліпшити психологічний стан та зміцнити здоров'я.

Функціональні вимоги:

– створення інтерактивного інтерфейсу, який дозволяє користувачам взаємодіяти з системою через текстові повідомлення, відеоконференції, медіа-елементи тощо;

– надання порад, інформації та стратегій психологічної підтримки, що відповідають індивідуальним потребам користувача;

– можливість відстежування настрою, зберігання журналу подій та взаємодії з системою.

Технічні вимоги:

– розробка серверної частини з використанням вебфреймворку, Nodejs, для обробки запитів користувачів та зберігання даних;

– використання мов програмування, таких як AngularJS, для розробки фронтенду та бізнес-логіки;

– використання алгоритмів машинного навчання та обробки природної мови для аналізу та розуміння текстової інформації.

Вимоги до управління проєктом:

– розробка плану проєкту з чіткими критеріями успіху, графіком робіт та ресурсними вимогами;

– управління командою розробників, тестувальників та інших зацікавлених сторін для забезпечення виконання проєкту вчасно та відповідно до вимог.

Тестування та якість:

– розробка тестових сценаріїв та проведення системного, модульного та інтеграційного тестування для перевірки функціональності та якості вебзастосунку;

– забезпечення стабільності, швидкодії та ефективності системи шляхом оптимізації коду, ресурсів та інших аспектів розробки.

Документація:

– розробка технічної документації, включаючи специфікації системи, діаграми бази даних, опис API та інструкції з розгортання та використання вебзастосунку [12];

– підтримка документації проєкту в актуальному стані та забезпечення доступу до неї для розробників та інших зацікавлених сторін.

Важливо враховувати, що конкретна постановка задачі, технічна пропозиція або специфікація проєкту ПЗ може значно варіюватись залежно від вимог та обмежень.

Висновки до розділу 1

На підставі аналізу предметної сфери в першому розділі були досліджені технології, які можуть бути використані для створення вебзастосунку, а також було розглянуто різні фреймворки для реалізації застосунку та існуючі аналоги.

Крім того, були проаналізовані аналогічні мобільні застосунки, проєкти з метою визначення їх особливостей та ідей, які можна використати. У результаті були визначені основні функції та характеристики.

Також був обґрунтований незвичний архітектурний підхід для проєктів, які базуються на платформі AngularJS.

2. МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Методи для вирішення задачі

Для досягнення поставленої мети було вибрано наступні фреймворки та технології. Для розробки вебзастосунка було обрано AngularJS, а для реалізації серверної частини використовуватиметься Node.js.

AngularJS — це платформа з відкритим кодом на основі JavaScript, розроблена Google для створення динамічних вебзастосунків. Він дотримується архітектурного шаблону Model-View-Controller (MVC), який допомагає розробникам структурувати свій код і розділяє проблеми між даними, презентацією та бізнес-логікою.

Іншим відмінним аспектом фреймворку є його можливість двостороннього зв'язування, що дозволяє динамічно змінювати дані в одній частині інтерфейсу при зміні даних моделі в іншій частині. Це означає, що AngularJS автоматично синхронізує модель і представлення.

Крім того, AngularJS має підтримку таких функцій, як Ajax-запити, керування DOM-структурою, можливості для створення анімацій, використання шаблонів, реалізацію маршрутизації та інші. Цей багатий функціонал разом з потужністю фреймворку призвели до того, що AngularJS широко застосовується у багатьох вебзастосунках і на даний момент вважається одним з найпопулярніших JavaScript-фреймворків.

Рисунок 2.1 – Головна сторінка вебсайту AngularJS

AngularJS надає набір потужних функцій, які полегшують розробку складних вебзастосунків.

Двостороннє прив'язування даних: AngularJS забезпечує автоматичну синхронізацію даних між моделлю (даними) і представленням (UI), усуваючи потребу в ручному маніпулюванні DOM.

Директиви: директиви дозволяють розробникам розширювати HTML за допомогою спеціальних атрибутів і елементів. Вони дозволяють створювати повторно використовувані компоненти та дозволяють декларативне програмування.

Ін'єкція залежностей: AngularJS має вбудовану систему ін'єкції залежностей, яка допомагає керувати залежностями між різними компонентами програми. Це робить код модульним, придатним для перевірки та зручнішим у обслуговуванні [13].

Шаблони: AngularJS використовує шаблони HTML для визначення структури та макета програми. Це дозволяє динамічно відтворювати дані шляхом інтеграції виразів і директив у шаблони.

Маршрутизація: AngularJS надає потужний механізм маршрутизації для створення односторінкових програм. Це дозволяє розробникам визначати маршрути, відображати їх у переглядах і керувати навігацією в програмі.

Тестування: AngularJS має чудову підтримку модульного тестування. Він надає такі інструменти та фреймворки, як Karma та Jasmine, які полегшують написання та виконання тестів для різних частин програми.

AngularJS завоював популярність серед розробників завдяки простоті використання, розширюваності та надійності. Однак важливо зазначити, що AngularJS є старішою версією фреймворку Angular.

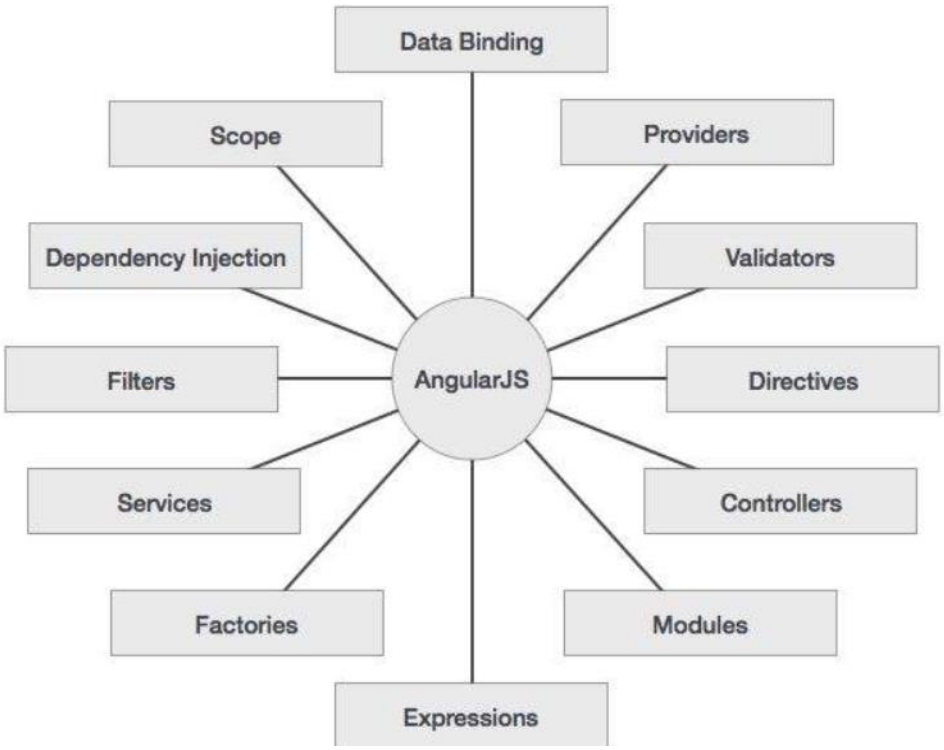

Рисунок 2.2 – Діаграма частин AngularJS

Angular є повністю переписаним AngularJS і пропонує кілька значних покращень. Покращена продуктивність Angular розроблено, щоб бути швидшим і ефективнішим, ніж AngularJS. Він представляє такі функції, як завчасна компіляція (AOT), відкладене завантаження та оптимізоване виявлення змін, що покращує продуктивність і покращує взаємодію з користувачем.

Model-View-Controller (MVC), як його зазвичай називають, — це шаблон проєктування програмного забезпечення для розробки вебзастосунків. Шаблон MVC [14] складається з наступних трьох компонентів:

– Model: це найнижчий рівень шаблону, який відповідає за зберігання та керування даними;

– View: представлення відповідає за відображення всіх або частини даних для користувача;

– Controller: контролер — це програмний код, який керує взаємодією між моделлю та представленням.

Простіше кажучи, шаблон MVC розділяє проблеми зберігання даних (модель), представлення даних (перегляд) і маніпуляції даними (контролер) у вебзастосунку. Це розділення допомагає підтримувати модульність коду, можливість повторного використання та загальну архітектуру програмного забезпечення.

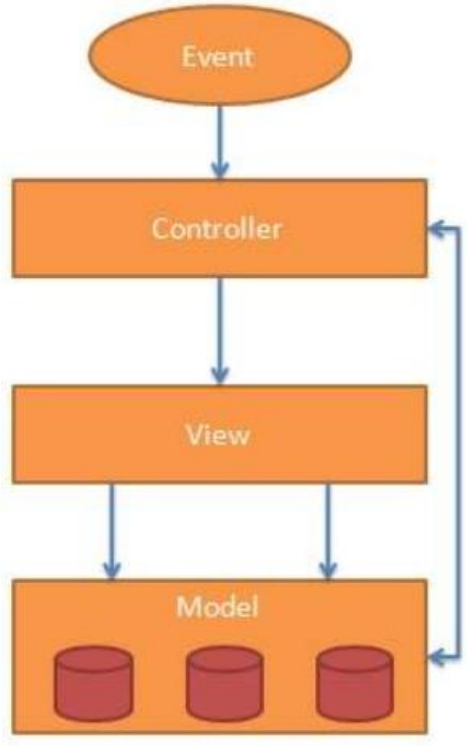

Рисунок 2.3 – Абстракція MVC

AngularJS має концепцію сервісів, які дозволяють створювати роздільні компоненти для обробки бізнес-логіки та спільного використання даних між компонентами застосунку. Сервіси допомагають підтримувати чистоту коду та спрощують керування залежностями між компонентами.

Node.js — це універсальне середовище виконання JavaScript, яке працює на різних платформах і доступне як рішення з відкритим кодом. Він набув популярності як інструмент, придатний для широкого кола проєктів.

Використовуючи механізм V8 JavaScript, який працює в Google Chrome, Node.js досягає вражаючих можливостей продуктивності.

Однією з відмінних особливостей Node.js є його здатність працювати в рамках одного процесу, усуваючи необхідність створювати нові потоки для кожного запиту. Стандартна бібліотека Node.js включає повний набір примітивів асинхронного введення-виведення, що гарантує, що код JavaScript не блокується. Більшість бібліотек у Node.js використовують неблокувальні парадигми, що робить поведінку блокування скоріше винятком, ніж нормою.

Коли Node.js стикається з операціями вводу-виводу, такими як мережевий зв’язок, доступ до бази даних або взаємодія з файловою системою, він уникає блокування потоку та зайвих витрат ЦП. Натомість Node.js відновлює ці операції після отримання відповідей.

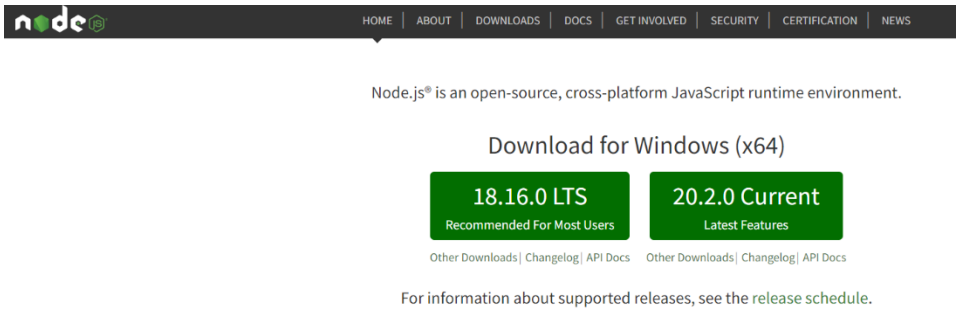

Рисунок 2.4 – Головна сторінка вебсайту Node.js

Цей унікальний підхід дозволяє Node.js обробляти безліч одночасних з'єднань за допомогою одного сервера без додаткової складності керування паралельністю потоків, яка часто призводить до помилок.

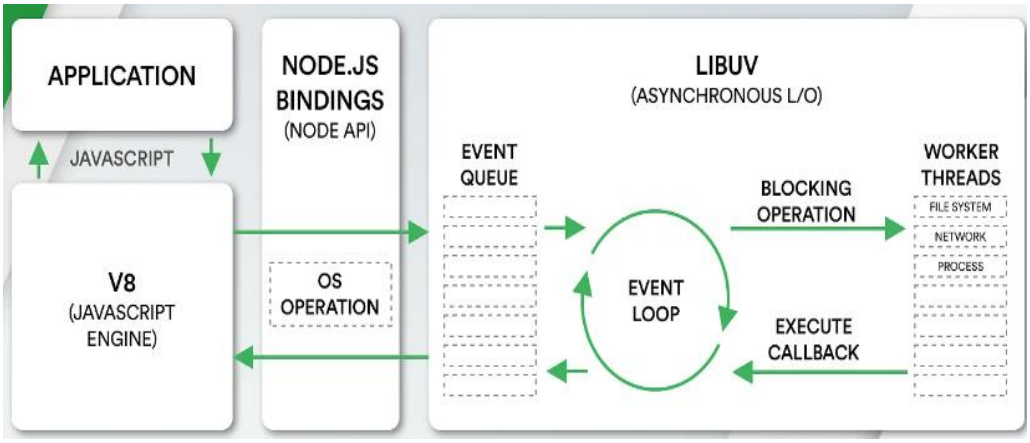

Рисунок 2.5 – Архітектура Node.js

Ще однією перевагою Node.js є його здатність подолати розрив між інтерфейсом і серверною розробкою. Фронтенд-розробники, які володіють JavaScript, можуть використати наявні навички для написання серверного коду, усуваючи потребу вивчати зовсім іншу мову програмування.

Node.js також забезпечує повну інтеграцію з останніми стандартами ECMAScript. Розробники можуть прийняти ці стандарти, не турбуючись про сумісність браузера, оскільки вони можуть вибрати версію Node.js, яка підтримує бажану версію ECMAScript. Крім того, можна ввімкнути експериментальні функції, використовуючи спеціальні позначки під час запуску Node.js.

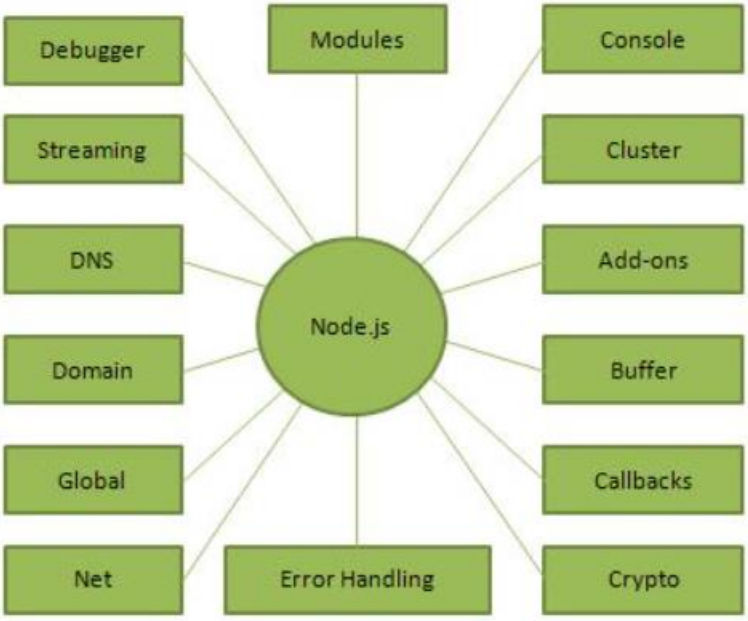

Рисунок 2.6 – Діаграма частин Node.js

Розширення за допомогою пакетного менеджера npm, Node.js має вбудований пакетний менеджер npm, який дозволяє легко встановлювати, оновлювати та управляти залежностями в проєкті. Багато модулів та бібліотек для Node.js доступні через npm, що допомагає розробникам ефективно використовувати готові рішення.

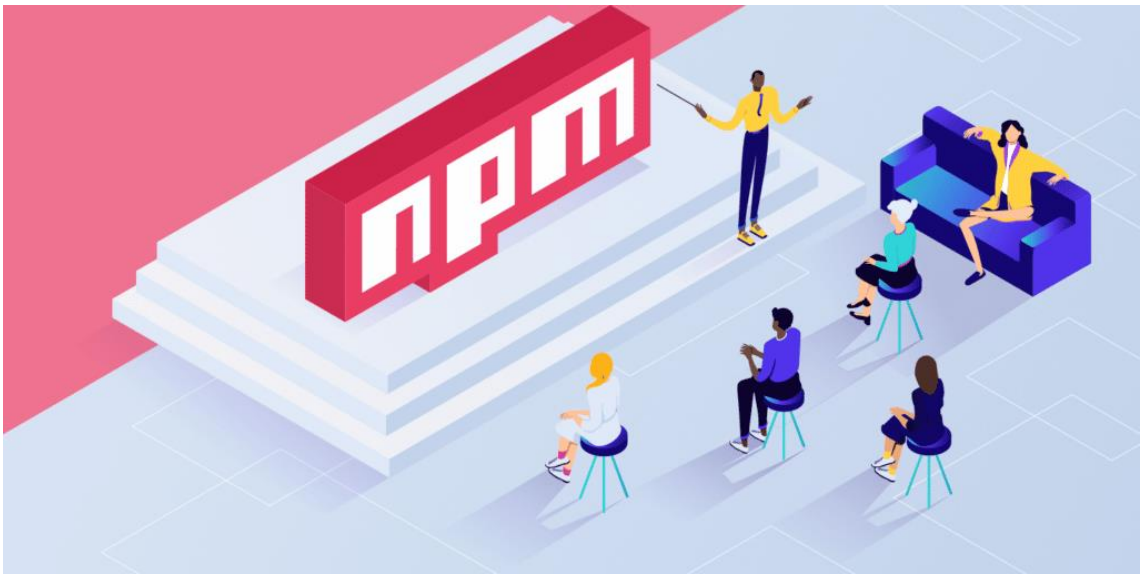

Рисунок 2.7 – Освоєння npm

NPM (Node Package Manager) — широко використовуваний менеджер пакетів для Node.js і JavaScript. Він служить центральним репозиторієм для спільного використання модулів і бібліотек багаторазового коду та керування ними, що полегшує розробникам включення попередньо створених функцій у свої проєкти [15].

Package Management, npm дозволяє розробникам легко встановлювати, оновлювати та видаляти пакети. Він керує залежностями, гарантуючи, що необхідні залежності для проєкту завантажено та встановлено правильно.

Command-Line Interface (CLI), npm надає інтерфейс командного рядка, який дозволяє розробникам ефективно взаємодіяти з менеджером пакетів. Розробники можуть виконувати різні команди, такі як встановлення пакетів, оновлення залежностей і публікація власних пакетів, безпосередньо з командного рядка [16].

Package Registry, npm містить велику колекцію пакетів у своєму реєстрі пакетів. Ці пакети містять широкий спектр функціональних можливостей, включаючи фреймворки, бібліотеки, утиліти та інструменти. Розробники можуть шукати пакети на основі своїх потреб і легко інтегрувати їх у свої проєкти.

Versioning: npm використовує семантичне керування версіями (semver) для керування версіями пакетів. Це гарантує, що розробники можуть вказати сумісні версії залежностей і контролювати, як оновлення застосовуються до їхніх проєктів. Це дозволяє розробникам використовувати переваги нових функцій і виправлення помилок, зберігаючи стабільність і сумісність.

Scripts: npm дозволяє розробникам визначати та запускати власні сценарії у своїх проєктах. Ця функція спрощує типові завдання розробки, надаючи ярлики для таких завдань, як виконання тестів, створення проєкту та запуск програми. Розробники можуть визначати сценарії у файлі package.json, пов’язаному з їхнім проєктом.

Collaboration npm сприяє співпраці та обміну кодом у спільноті розробників. Розробники можуть публікувати власні пакети в реєстрі npm, роблячи їх доступними для використання іншими. Вони також можуть брати участь в обговореннях, робити внесок у проєкти з відкритим кодом і співпрацювати з іншими розробниками через екосистему npm.

Загалом npm відіграє вирішальну роль в екосистемі Node.js і JavaScript, надаючи надійне рішення для керування пакетами, забезпечуючи ефективне повторне використання коду та сприяючи активній спільноті розробників.

Angular було обрано для розробки веб-додатку з кількох причин:

– надійний фреймворк: Angular — це потужний і комплексний фреймворк для створення веб-додатків. Він забезпечує міцну основу та чітко визначену структуру для розробки складних програм. Широкий набір функцій, таких як прив’язка даних, впровадження залежностей і компонентна архітектура, роблять його придатним для створення масштабованих і підтримуваних програм;

– двостороннє зв’язування даних: функція двостороннього зв’язування даних Angular спрощує синхронізацію даних між моделлю програми та представленням. Це означає, що будь-які зміни, внесені в інтерфейс користувача, автоматично відображаються в основній моделі даних, і навпаки. Це зменшує потребу в ручному маніпулюванні DOM, що призводить до більш ефективної та оптимізованої розробки;

– компонентна архітектура: Angular дотримується компонентної архітектури, де додаток розділено на повторно використовувані та незалежні компоненти. Цей підхід сприяє модульності, багаторазовому використанню та ремонтопридатності. Компоненти інкапсулюють власну логіку, стилі та шаблони, що полегшує розробку, тестування та зміну окремих частин програми, не впливаючи на всю систему;

– підтримка TypeScript: Angular створено за допомогою TypeScript, надмножини JavaScript зі статичною типізацією. TypeScript надає додаткові

переваги, такі як покращена якість коду, покращена підтримка інструментів і кращий досвід розробника. Його статична перевірка типів допомагає виявляти помилки під час розробки, роблячи код надійнішим і простішим у обслуговуванні;

– велика спільнота та екосистема: Angular має процвітаючу спільноту розробників, що означає доступ до великих ресурсів, навчальних посібників і бібліотек. Ця активна спільнота забезпечує постійну підтримку, регулярні оновлення та безліч інтеграцій сторонніх розробників, що полегшує вирішення проблем, пошук рішень і використання існуючих інструментів;

– зручна для мобільних пристроїв і кросплатформна розробка: Angular має вбудовану підтримку створення мобільних додатків за допомогою фреймворків, таких як Ionic і NativeScript. Це дозволяє створити єдину кодову базу, яку можна розгорнути на кількох платформах, включаючи веб, iOS та Android, заощаджуючи час і зусилля на розробку.

Враховуючи ці фактори, Angular було визнано придатним для розробки веб-додатку, пропонуючи надійну та ефективну структуру для створення масштабованого та багатофункціонального рішення для надання психологічної підтримки під час війни.

2.2 Технології розробки системи

Для збереження та керування даними було обрано MongoDB та API та взаємодія з іншими системами. Було реалізовано RESTful API для комунікації, вебсервісами.

MongoDB — це популярна документоорієнтована база даних NoSQL з відкритим кодом. Він надає гнучке та масштабоване рішення для керування та зберігання даних у спосіб, який відрізняється від традиційних реляційних баз даних.

Орієнтований на документ: MongoDB організовує дані у форматі документа під назвою BSON (двійковий JSON). Документ — це самодостатня

структура даних, яка може містити вкладені поля, масиви та пари ключ-значення. Цей документний підхід дозволяє розробникам зберігати та отримувати складні структури даних без необхідності складних об’єднань або попередньо визначених схем [17].

Масштабованість і продуктивність: MongoDB розроблено для горизонтального масштабування між кількома серверами, що робить його придатним для обробки великих обсягів даних і високих навантажень трафіку. Він розподіляє дані між кількома машинами та надає вбудовані механізми сегментування та реплікації для забезпечення доступності даних, від мовостійкості та продуктивності.

Гнучка схема: на відміну від традиційних реляційних баз даних, MongoDB не передбачає застосування жорсткої схеми. Кожен документ у колекції може мати власну унікальну структуру, що дозволяє створювати динамічні моделі даних, що розвиваються. Ця гнучкість особливо корисна в гнучких середовищах розробки, де вимоги часто змінюються.

Запити та індексування: MongoDB підтримує потужну мову запитів, яка дозволяє розробникам легко отримувати та маніпулювати даними. Він надає широкий спектр операторів запитів і підтримує розширені запити, включаючи фільтрування, сортування, агрегацію та геопросторові запити. Індекси можна створювати на полях для покращення продуктивності запитів.

Реплікація та висока доступність: MongoDB забезпечує автоматичну реплікацію та можливості відновлення після відмови. Він може підтримувати кілька копій даних на кількох серверах, гарантуючи, що якщо один сервер вийде з ладу, інший зможе безперешкодно взяти на себе роботу. Цей механізм реплікації забезпечує високу доступність і резервування даних.

Горизонтальне масштабування: функція шардингу MongoDB дозволяє розподіляти дані між декількома серверами або кластерами, забезпечуючи горизонтальне масштабування. Шардинг ділить дані на менші фрагменти та

розподіляє їх між шардами, забезпечуючи ефективний розподіл даних і робочого навантаження.

Інтеграція та екосистема: MongoDB добре інтегрується з популярними мовами програмування та фреймворками. Він надає офіційні драйвери для різних мов програмування, що полегшує взаємодію з базою даних на обраній мові. Крім того, MongoDB пропонує багату екосистему з численними сторонніми бібліотеками, інструментами та фреймворками, які розширюють її функціональність.

Перевірка документів: MongoDB дозволяє розробникам визначати правила перевірки документів, забезпечуючи цілісність і послідовність даних. Правила перевірки можуть застосовувати типи даних, обов’язкові поля, унікальні значення та спеціальні обмеження, допомагаючи підтримувати якість даних.

Агрегація: framework агрегації MongoDB дозволяє виконувати розширені операції аналізу та обробки даних. За допомогою конвеєрів агрегації та різних етапів, таких як $match, $group, $project, $sort тощо, можна застосовувати алгоритми для маніпулювання, перетворення та агрегування даних. Це дає змогу виконувати складну аналітику, створювати звіти та отримувати цінну інформацію з ваших даних.

Спільнота та підтримка: MongoDB має велику та активну спільноту розробників, які надають ресурси, навчальні посібники та форуми для обміну знаннями та вирішення проблем. MongoDB Inc., компанія, що стоїть за MongoDB, пропонує підтримку корпоративного рівня, консультаційні послуги та додаткові функції для комерційного розгортання.

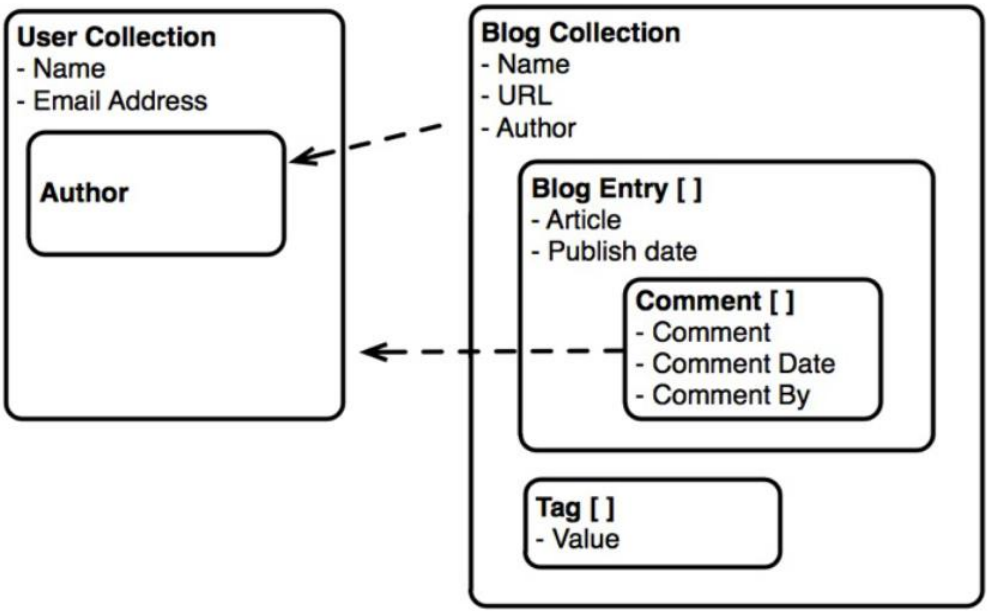

Рисунок 2.8 – Діаграма з усіма моделями

Поєднання гнучкості, масштабованості та простоти використання MongoDB зробило його популярним вибором для широкого спектру програм, включаючи веб- та мобільні програми, системи керування контентом, аналітику в реальному часі та платформи Інтернету речей. Його здатність обробляти різноманітні моделі даних і масштабувати по горизонталі робить його переконливим варіантом для сучасних потреб зберігання та керування даними.

Для комунікації з вебсервісами та забезпечення взаємодії з іншими системами було обрано RESTful API. REST (Representational State Transfer) є архітектурним стилем, що використовується для створення вебсервісів, які працюють на базі HTTP-протоколу [18].

Використання RESTful API дозволяє створювати легко доступні та розширювані вебсервіси. Цей підхід базується на принципах, таких як використання HTTP-методів (GET, POST, PUT, DELETE) для взаємодії з ресурсами, використання стандартних HTTP-статус кодів для повідомлення про статус операцій та передача даних у форматі JSON або XML.

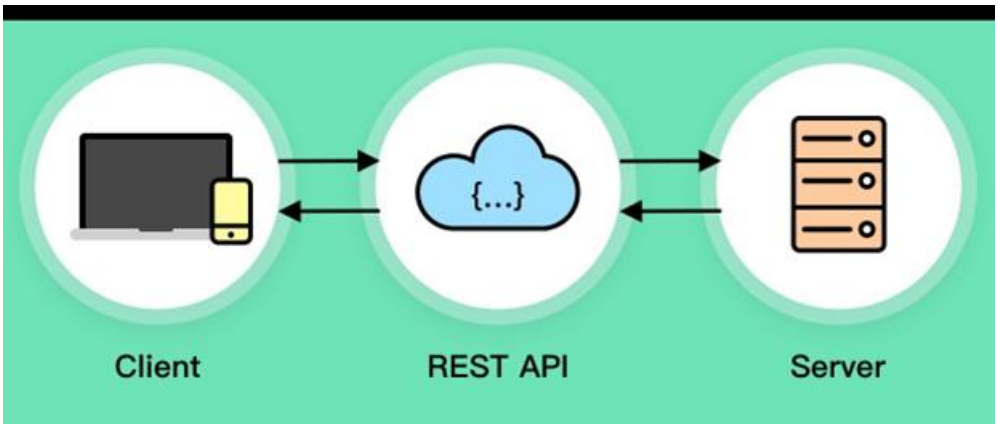

Рисунок 2.9 – Взаємодія сервіса та клієнта

Загальний стандарт API, який використовується більшістю мобільних і вебзастосунків для спілкування з серверами, називається REST. Це означає Representational State Transfer.

REST не є специфікацією. Це вільний набір правил, який був де-факто стандартом для створення веб-API з початку 2000-х років.

RESTful API (Representational State Transfer) — це архітектурний стиль для розробки мережевих програм, які спілкуються через HTTP. Він надає стандартний набір вказівок і принципів для створення масштабованих вебсервісів без збереження стану, які легко підтримувати.

Принципи REST: REST дотримується набору принципів, які керують дизайном і взаємодією його компонентів:

– не маючи конкретного стану: кожен запит від клієнта до сервера повинен містити всю необхідну інформацію, щоб сервер міг зрозуміти та обробити запит. Сервер не повинен зберігати стан клієнта між запитами;

– орієнтованість на ресурси: RESTful API побудовані навколо ресурсів, якими може бути будь-яка інформація чи об'єкт, відкритий сервером. Ресурси ідентифікуються унікальними URI (уніфікованими ідентифікаторами ресурсів);

– уніфікований інтерфейс: API RESTful використовують уніфікований набір методів HTTP (GET, POST, PUT, DELETE) для виконання

операцій над ресурсами. Ці методи використовуються послідовно на різних ресурсах;

– кінцеві точки та URI: RESTful API надають кінцеві точки, які представляють певні ресурси або колекції ресурсів. Кожна кінцева точка пов’язана з унікальним URI (уніфікованим ідентифікатором ресурсу), який клієнти можуть використовувати для доступу або маніпулювання відповідним ресурсом. Наприклад, /users може представляти колекцію користувачів, тоді як /users/{id} може представляти конкретного користувача.

Методи HTTP: API RESTful використовують методи HTTP для виконання операцій із ресурсами:

– GET: отримує представлення ресурсу або колекції ресурсів;
– POST: створює новий ресурс;
– PUT: оновлює або замінює існуючий ресурс;
– DELETE: видаляє ресурс.

Запит і відповідь: RESTful API обмінюються даними через HTTP-запити та відповіді. Клієнти надсилають запити на сервер, вказуючи бажаний метод HTTP, URI, заголовки та необов’язковий текст запиту. Сервер обробляє запит і повертає відповідну відповідь HTTP, яка зазвичай містить код стану, заголовки та тіло відповіді.

Представлення: RESTful API підтримують різні представлення ресурсів, наприклад JSON, XML або навіть HTML. Клієнти та сервери можуть узгодити відповідне представлення за допомогою заголовків Accept і Content-Type.

Гіпермедіа (HATEOAS): RESTful API можуть включати гіпермедійні посилання у відповіді, щоб забезпечити видимість і навігацію між пов’язаними ресурсами. Гіпермедійні посилання дозволяють клієнтам динамічно досліджувати та взаємодіяти з API, переходячи за посиланнями, вбудованими у відповіді.

Автентифікація та безпека: RESTful API часто використовують різні механізми автентифікації та безпеки для захисту ресурсів і забезпечення авторизованого доступу. Це може включати використання токенів, ключів API або OAuth для автентифікації, а також шифрування та безпечних протоколів (HTTPS) для передачі даних.

Обробка помилок: API RESTful використовують відповідні коди статусу HTTP, щоб вказати на успішність або невдачу запиту. Загальні коди стану включають 200 (OK), 201 (Створено), 400 (Неправильний запит), 401 (Неавторизовано), 404 (Не знайдено) і 500 (Внутрішня помилка сервера). Відповіді про помилки часто містять додаткові деталі в тілі відповіді.

Масштабованість і кешування: RESTful API можна розробити так, щоб вони були високомасштабованими за допомогою механізмів кешування. Кешування відповідей на стороні клієнта або використання заголовків кешування HTTP може зменшити навантаження на сервер і підвищити продуктивність.

Документація: добре задокументовані RESTful API надають чітку та вичерпну документацію, щоб допомогти розробникам зрозуміти доступні ресурси, їхні URI, підтримувані методи HTTP, формати запитів/відповідей і будь-які додаткові вказівки чи обмеження.

Дотримуючись принципів REST, розробники можуть створювати API, сумісні, масштабовані та легкі для використання різними клієнтами. RESTful API набули широкого поширення та зазвичай використовуються для створення вебсервісів, які забезпечують роботу сучасних програм і забезпечують бездоганну інтеграцію між різними системами.

Простими словами, в архітектурному стилі REST дані та функціональність вважаються ресурсами, і доступ до них здійснюється за допомогою уніфікованих ідентифікаторів ресурсів (URI).

Ресурси впливають за допомогою набору простих, чітко визначених операцій. Крім того, ресурси мають бути відокремлені від їх представлення,

щоб клієнти могли отримати доступ до вмісту в різних форматах, таких як HTML, XML, звичайний текст, PDF, JPEG, JSON та інші.

Клієнти та сервери обмінюються представленнями ресурсів за допомогою стандартизованого інтерфейсу та протоколу. Зазвичай протокол HTTP є найбільш використовуваним, але REST не вимагає його.

RESTful API було обрано з кількох причин:

– простота: RESTful API має простий та інтуїтивно зрозумілий архітектурний стиль на основі стандартних протоколів HTTP. Він використовує загальні методи HTTP, такі як GET, POST, PUT і DELETE для виконання операцій CRUD (Створення, читання, оновлення, видалення) над ресурсами. Ця простота полегшує проектування, розробку та підтримку API;

– сумісність: RESTful API сумісний з різними платформами, мовами та фреймворками. Його можна легко використовувати різними клієнтами, включаючи веб-браузери, мобільні пристрої та програми сторонніх розробників. Ця сумісність забезпечує кращу інтеграцію та взаємодію з існуючими системами;

– масштабованість: незалежний характер RESTful API і легкий зв’язок роблять його дуже масштабованим. Він може обробляти велику кількість одночасних запитів і розподіляти навантаження між кількома серверами або безсерверними середовищами. Ця масштабованість має важливе значення для додатків, яким необхідно пристосуватися до зростаючих баз користувачів і обробляти великі обсяги трафіку;

– гнучкість: RESTful API забезпечує гнучкість щодо форматів даних. Він підтримує різні представлення, такі як JSON, XML або навіть двійкові формати. Ця гнучкість дозволяє клієнтам вибирати формат, який найбільше підходить для їхніх потреб, і забезпечує легку інтеграцію з іншими системами чи службами;

– кешування та продуктивність: RESTful API використовує механізми кешування HTTP, дозволяючи кешувати та обслуговувати відповіді

з систем-посередників. Це покращує продуктивність, зменшує навантаження на сервер і покращує загальну взаємодію з користувачем. Кешування може бути особливо корисним для інтенсивних читання програм або API із часто доступними ресурсами;

– промисловий стандарт: RESTful API став галузевим стандартом для розробки веб-сервісів. Він широко прийнятий і добре задокументований, з великою спільнотою розробників і доступними ресурсами. Ця стандартизація забезпечує сумісність, взаємодію та довгострокову підтримку API.

Обираючи RESTful API, додаток отримує переваги від перевіреного та широко прийнятого архітектурного стилю, який сприяє простоті, сумісності, масштабованості, гнучкості та продуктивності. Він узгоджується з сучасними найкращими практиками та забезпечує бездоганну інтеграцію з іншими системами та майбутні вдосконалення.

Висновки до розділу 2

У другому розділі було проведено детальний аналіз різних методів та технологій для розробки вебзастосунку. Після уважного розгляду було вирішено використовувати AngularJS для розробки фронтенду, оскільки цей фреймворк надає потужні можливості розширення HTML та дозволяє створювати виразний та читабельний код.

Для бекенду було обрано node.js, оскільки це середовище виконання JavaScript, яке дозволяє швидко та ефективно обробляти запити на серверному боці. Використання node.js забезпечить швидку реакцію на запити клієнта та ефективне взаємодію з базою даних та іншими зовнішніми сервісами.

Для взаємодії з вебсервісами було вибрано restfulapi, який ґрунтується на стандартних вебпротоколах, таких як HTTP. Цей підхід забезпечує ефективний обмін даними між клієнтом та сервером та дозволяє створювати легкодоступний та масштабований інтерфейс для взаємодії з вебсервісами.

В результаті цього аналізу було визначено основні функції та характеристики вебзастосунку, які будуть реалізовані з використанням обраного набору технологій.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис вхідних даних та структури системи

В цьому розділі буде проведено ознайомлення зі сценаріями використання вебзастосунку. Для цього будуть створені діаграми варіантів використання та описані варіанти використання системи з використанням сценарної техніки опису взаємодії Use Case.

Для створення діаграми варіантів використання був обраний інструмент для розробки UML-діаграм під назвою draw.io, розроблений компанією JGraph Ltd [19].

На рис. 3.1 показано інтерфейс даного інструменту.

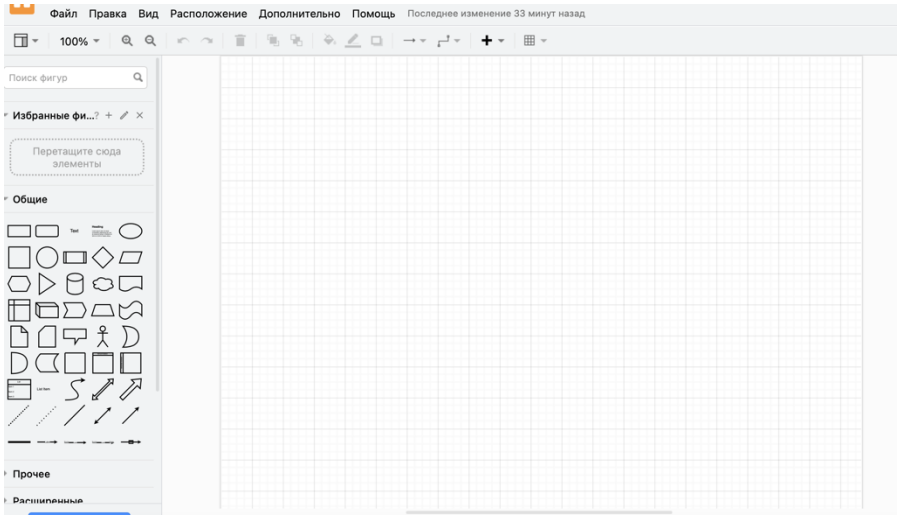

Рисунок 3.1 – Інтерфейс draw.io

У системі присутня лише одна активна особа - Користувач. На рис. 3.2 зображено діаграму варіантів використання, де відображена ця активна особа та функції вебзастосунку, до яких вона має доступ.

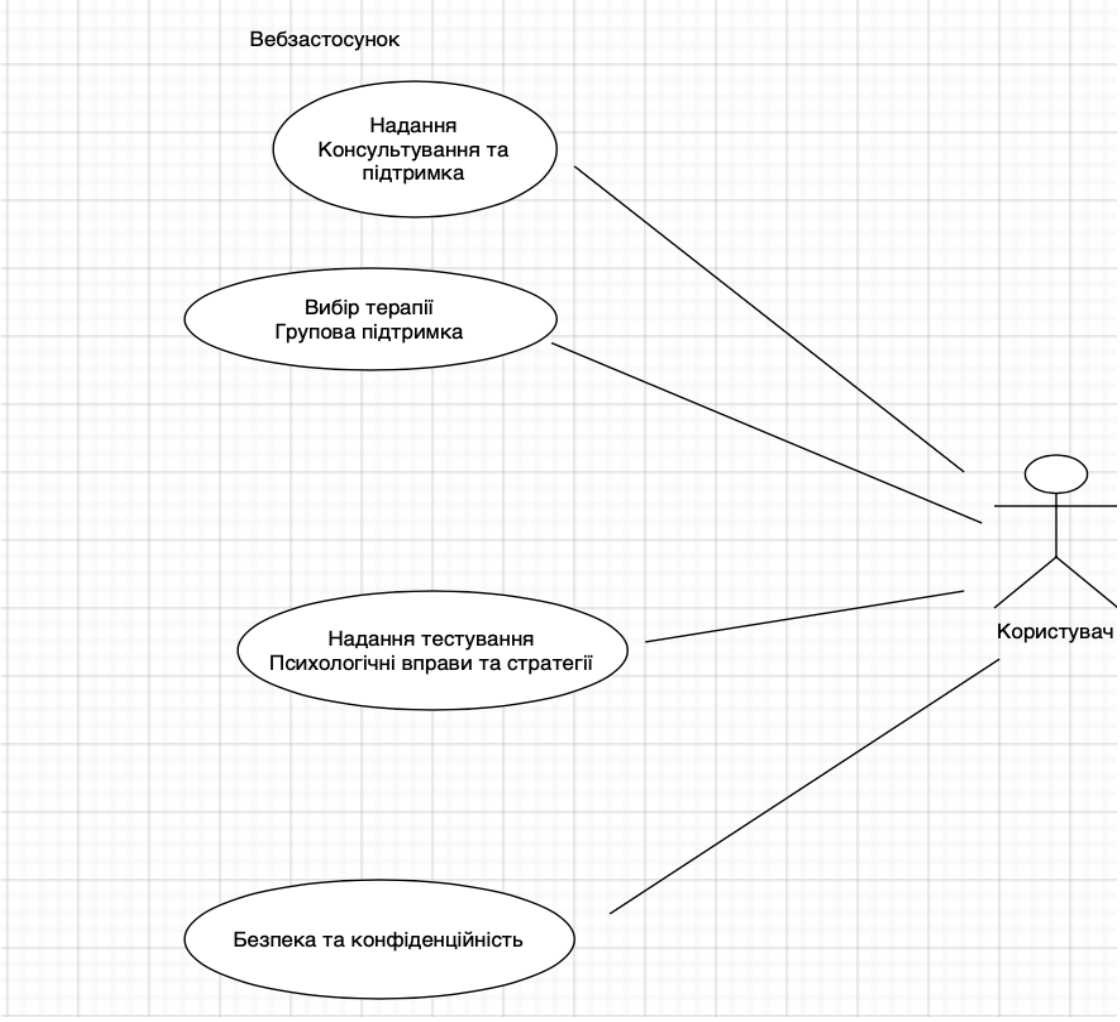

Рисунок 3.2 – Діаграма варіантів використання

Тепер ми перейдемо до створення прикладів сценаріїв використання системи.

Сценарій 1: вхід до системи:

– користувач відвідує сайт і бачить сторінку входу;

– користувач вводить свій логін та пароль;

– система перевіряє введені дані з базою даних;

– якщо введені дані співпадають з даними в базі, користувачу надається доступ до головної сторінки сайту;

– користувач потрапляє на головну сторінку сайту.

Діаграма для цього сценарію представлена на рис. 3.3.

Рисунок 3.3 – Вхід до системи

Сценарій 2: вхід до системи з альтернативними сценаріями.

Головний сценарій (Успішний):

– користувач відвідує сайт і бачить сторінку входу;

– користувач вводить свій логін та пароль;

– система перевіряє введені дані з базою даних;

– якщо введені дані співпадають з даними в базі, користувачу надається доступ до головної сторінки сайту;

– користувач потрапляє на головну сторінку.

Альтернативні сценарії:

Альтернативний сценарій - Невірний логін або пароль:

– після введення логіну та паролю, система виявляє, що вони не співпадають з даними в базі;

– система відображає повідомлення про невірний логін або пароль;

– користувач має можливість спробувати ввести логін та пароль знову або скористатися опцією відновлення паролю.

Альтернативний сценарій - Відсутність облікового запису:

– система не знаходить введений користувачем логін в базі даних;

– система відображає повідомлення про відсутність облікового запису;

– користувач має можливість створити новий обліковий запис або звернутися до адміністратора для отримання додаткової допомоги.

Діаграма для цього сценарію та альтернативних сценаріїв може бути представлена на рис. 3.4.

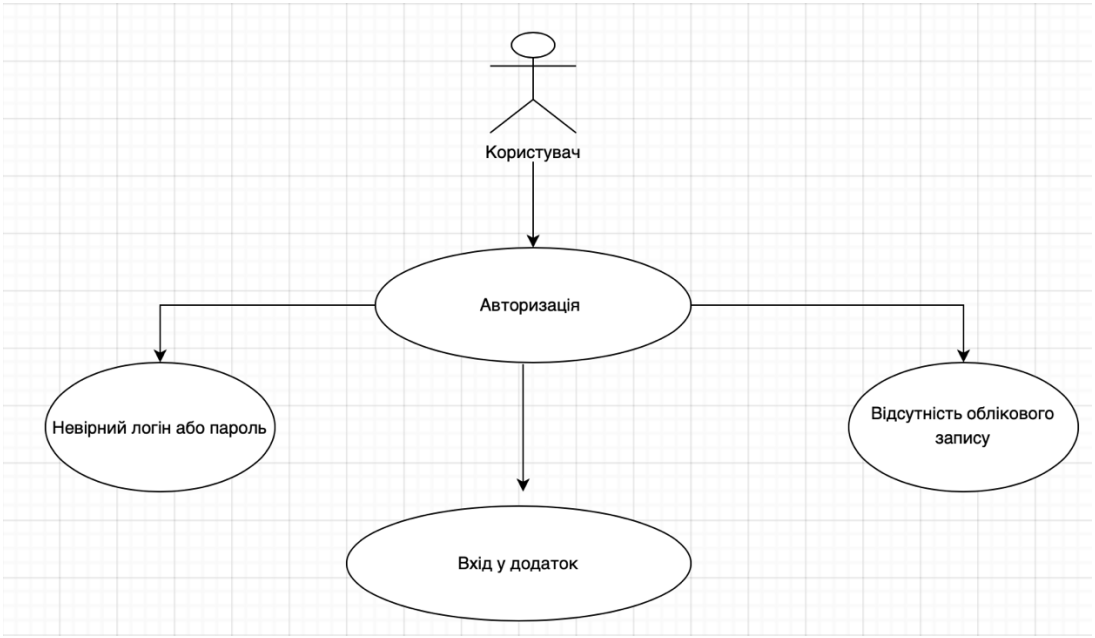

Рисунок 3.4 – Авторизація с альтернативними сценаріями

Ці альтернативні сценарії розширюють головний сценарій "Вхід до системи" і надають можливість користувачу взаємодіяти з системою в разі невірного логіну або паролю, або якщо відсутній обліковий запис.

Сценарій 3: психологічна підтримка під час війни:

– користувач перебуває в зоні конфлікту або під впливом війни;

– користувач звертається до системи для отримання психологічної підтримки під час цього складного періоду;

– система пропонує користувачеві різні форми психологічної підтримки, такі як телефонні лінії довіри, онлайн-консультації, групові терапії, та інші;

– користувач обирає зручний для нього спосіб отримання підтримки;

– у разі телефонної лінії довіри, користувач дзвонить на відповідний номер і спілкується з кваліфікованим психологом;

– психолог надає користувачеві емоційну підтримку, допомагає зняти стрес і надає поради щодо копінг-стратегій;

– у разі онлайн-консультацій, користувач здійснює відеодзвінок або чат з психологом, де вони обговорюють проблеми та шукають шляхи подолання труднощів;

– в групових терапіях, користувач бере участь у спільних сесіях з іншими людьми, які також переживають війну, де психолог керує груповим процесом та сприяє взаємопідтримці та обміну досвідом.

Цей сценарій описує процес надання психологічної підтримки користувачам, які перебувають під впливом війни. Через різноманітні форми підтримки, такі як телефонні лінії довіри, онлайн-консультації та групові терапії, система допомагає користувачам зняти стрес, знайти підтримку у спілкуванні з психологами та іншими людьми, а також розвивати копінг-стратегії для подолання труднощів, пов'язаних з війною. Крім того, система надає користувачам можливість доступу до ресурсів і матеріалів, спрямованих на підвищення свідомості щодо психологічного благополуччя та самодогляду під час воєнного конфлікту. Користувачі можуть отримати інформацію про стресові реакції, поширені симптоми травми та ефективні методи самодопомоги.

Таблиця 3.1 – Приклад психологічної підтримки під час війни

Use Case Name	Психологічна підтримка під час війни
Scope	Вебзастосунок для психологічної підтримки користувачів під час війни
Level	Забезпечити психологічну підтримку та ресурси для користувачів під час війни

Primary Actor	Користувач (особа, яка переживає воєнний конфлікт)
Stakeholders and interests	Користувач: Зацікавлений у отриманні психологічної підтримки, зниженні стресу та покращенні емоційного стану під час війни.
Precondition	Користувач відкриває вебзастосунок на смартфоні або комп'ютері та авторизується в системі.
Success guarantee	Користувач успішно отримує психологічну підтримку та ресурси для покращення свого психологічного стану під час війни.

Main Success Scenario	Користувач входить до системи. Користувач знаходить розділ, присвячений психологічній підтримці під час війни. Користувач має доступ до різних ресурсів, таких як статті, поради, відео, звукові записи, спеціалізовані форуми тощо, що надають психологічну підтримку та інформацію. Користувач вибирає потрібний ресурс або розділ для отримання психологічної підтримки. Користувач використовує доступні ресурси для отримання психологічної підтримки та розвитку копінг-стратегій під час війни.

Закінчення таблиці 3.1 - Приклад психологічної підтримки під час війни

Extensions	Користувач може звернутися до онлайн-консультанта або психолога для індивідуальної психологічної підтримки. Користувач може приєднатися до групових сесій або терапевтичних груп для спілкування та підтримки з іншими людьми, які переживають війну.
Special Requirements	Пристрій з доступом до Інтернету та веббраузером.
Frequency of Occurrence	Система доступна користувачам безперервно під час воєнного конфлікту.

У проєктуванні об'єктно-орієнтованої системи, діаграма класів займає важливе місце. Вона використовується на різних етапах проєктування та реалізації з різним ступенем деталізації.

Діаграма класів є одним з основних інструментів проектування об'єктно-орієнтованої системи. Вона дозволяє візуалізувати структуру системи, визначити класи, їх взаємозв'язки і основні властивості. Діаграма класів надає зрозумілий і зручний спосіб представлення системи, дозволяючи аналізувати її архітектуру, взаємодії між класами та організацію даних.

В цьому конкретному випадку, вебзастосунок, що розроблюється, складається з 4 класів. Діаграма класів для цього застосунку представлена на рис. 3.5.

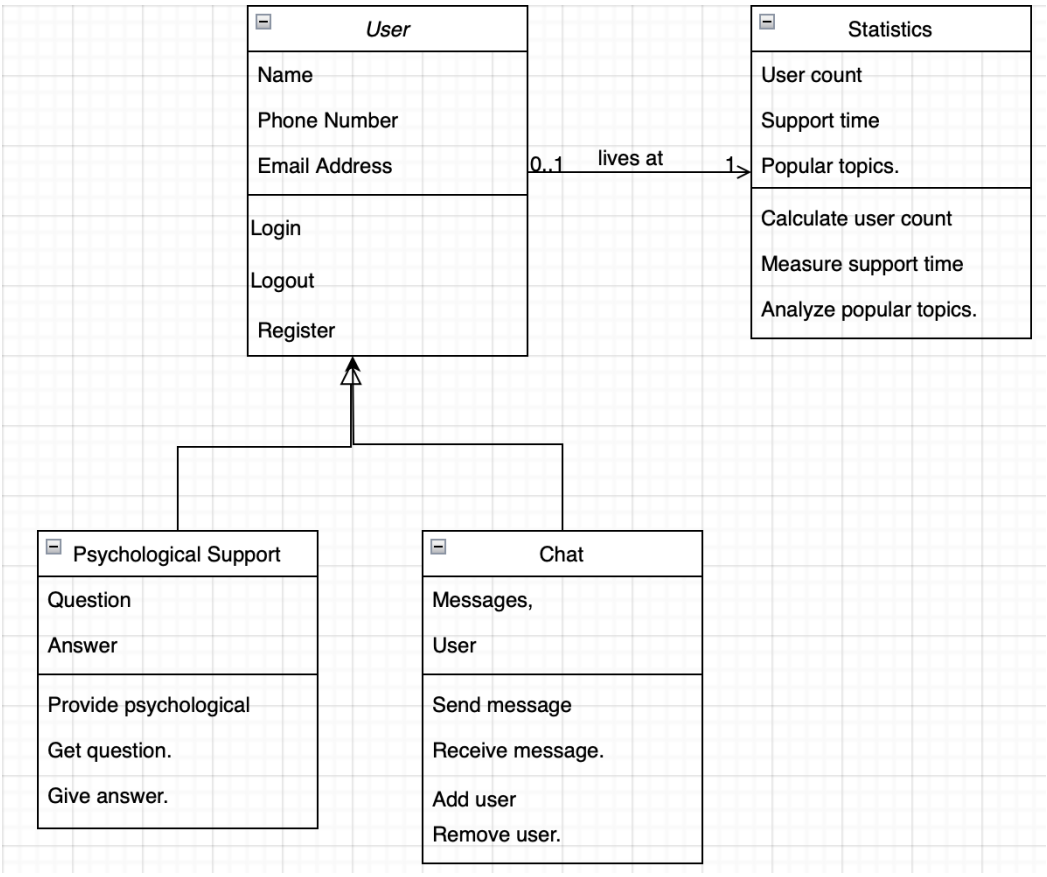

Рисунок 3.5 – Діаграма класів вебзастосунку

NoSQL бази даних використовують іншу модель даних, ніж традиційні реляційні бази даних, тому для них не використовується традиційна діаграма класів, яка базується на структурі таблиць і зв'язків між ними. Замість цього, NoSQL бази даних можуть використовувати різні моделі даних, такі як документи, колонки, ключ-значення або графи:

– діаграма документів, ця модель даних використовується, наприклад, в базі даних MongoDB. Діаграма може відображати колекції документів (аналог таблиць) і поля (положені всередині документів). Кожен документ може мати власну структуру і може містити різні типи полів, такі як рядки, числа, масиви або навіть вкладені документи;

– діаграма ключ-значення, ця модель даних використовується, наприклад, в базі даних Redis. Діаграма може показувати ключі (унікальні ідентифікатори) і значення, які пов'язані з цими ключами. Значення можуть бути простими рядками, числами або складнішими структурами даних, такими

як хеш-таблиці або списки. Діаграма графа: Ця модель даних використовується, наприклад, в базі даних Neo4j;

– діаграма може показувати вузли (сутності) і ребра (зв'язки) між цими вузлами. Кожен вузол може мати свої атрибути і може бути пов'язаний з іншими вузлами за допомогою ребер.

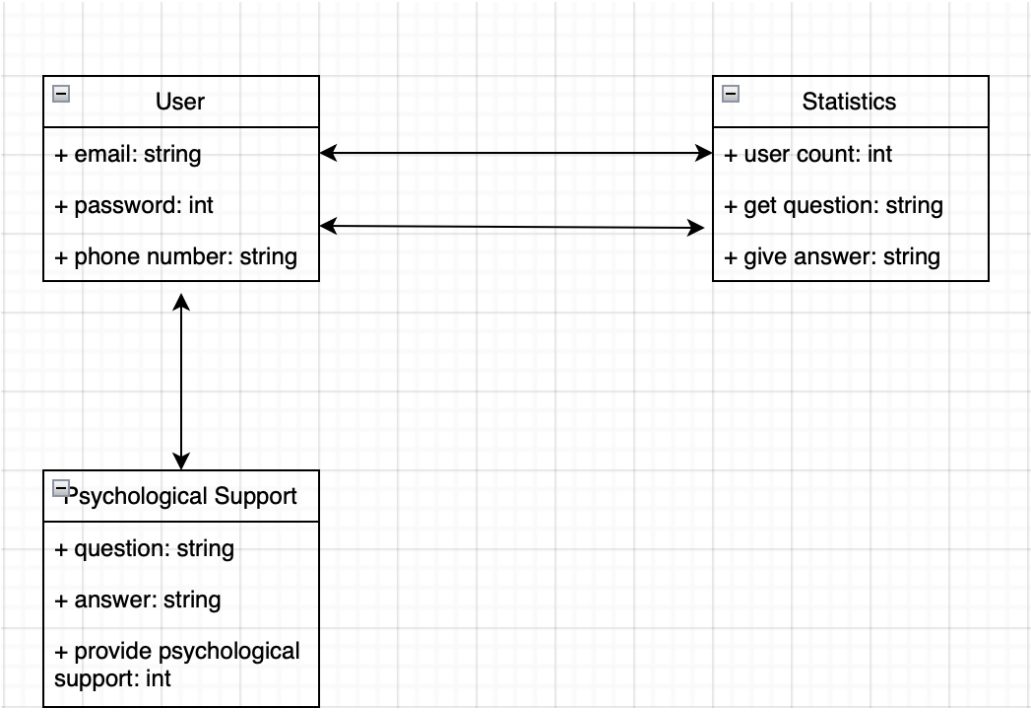

Рисунок 3.6 – Діаграма nosql бази даних

В діаграмі класів для NoSQL баз даних відображаються моделі даних та їх зв'язки, а також властивості та методи, які використовуються для роботи з цими моделями. Діаграма може також відображати індекси, спеціальні запити та інші деталі, які стосуються роботи з базою даних.

Враховуючи вищесказане, було б коректніше описувати структуру та взаємозв'язки конкретної NoSQL бази даних, замість загальної діаграми класів для всіх NoSQL баз даних.

3.2 Проєктування інформаційної системи

Visual Studio Code [20] - це високоефективний редактор коду, легкий і надійний. Його можна встановити на робочому столі та він сумісний з

операційними системами Windows, macOS і Linux. Він пропонує повні функції для розробки JavaScript, TypeScript і Node.js прямо з коробки. Крім того, він може похвалитися широким набором розширень, які підтримують різні мови програмування та середовища виконання, включаючи, але не обмежуючись C++, C#, Java, Python, PHP, Go та .NET.

Visual Studio Code (VS Code) — популярний редактор вихідного коду, розроблений Microsoft. Він розроблений як легкий, з можливістю налаштування та ефективний, що робить його кращим вибором для багатьох розробників:

– сумісність із різними платформами, VS Code доступний для Windows, macOS і Linux, що дозволяє розробникам без проблем працювати з різними операційними системами;

– розширюваність, VS Code має багату екосистему розширень, які надають додаткові функції та підтримку для різних мов програмування, фреймворків та інструментів. Ці розширення можна легко встановити та налаштувати для розширення можливостей редактора;

– функції, схожі на інтегроване середовище розробки (IDE): незважаючи на легкість, VS Code пропонує багато потужних функцій, доступних у повноцінних IDE. Він включає підтримку редагування коду, підсвічування синтаксису, IntelliSense (доповнення коду та пропозиції), налагодження, інтеграцію контролю версій (Git) і термінальний доступ;

– налаштування, VS Code дозволяє користувачам персоналізувати своє середовище розробки, налаштовуючи теми, комбінації клавіш і налаштування. Користувачі також можуть створювати та ділитися власними розширеннями, щоб адаптувати редактор до своїх конкретних потреб;

– інтегрований термінал, VS Code надає інтегрований термінал у редакторі, що дозволяє розробникам запускати команди, сценарії та взаємодіяти з інтерфейсом командного рядка, не виходячи з редактора;

– інтеграція Git, VS Code має вбудовану підтримку контролю версій Git, що дає змогу розробникам виконувати звичайні операції Git, такі як постановка, фіксація, розгалуження та злиття, безпосередньо з редактора;

– intelliCode, VS Code включає IntelliCode на основі штучного інтелекту, який використовує алгоритми машинного навчання для надання інтелектуальних пропозицій коду на основі загальних шаблонів кодування та практики;

– налагодження, VS Code пропонує надійні можливості налагодження, підтримуючи різні мови програмування та фреймворки. Це дозволяє розробникам встановлювати контрольні точки, перевіряти змінні, покроково виконувати код і ефективно налагоджувати свої програми.

Ви можете переглянути інтерфейс редактора на рис. 3.7.

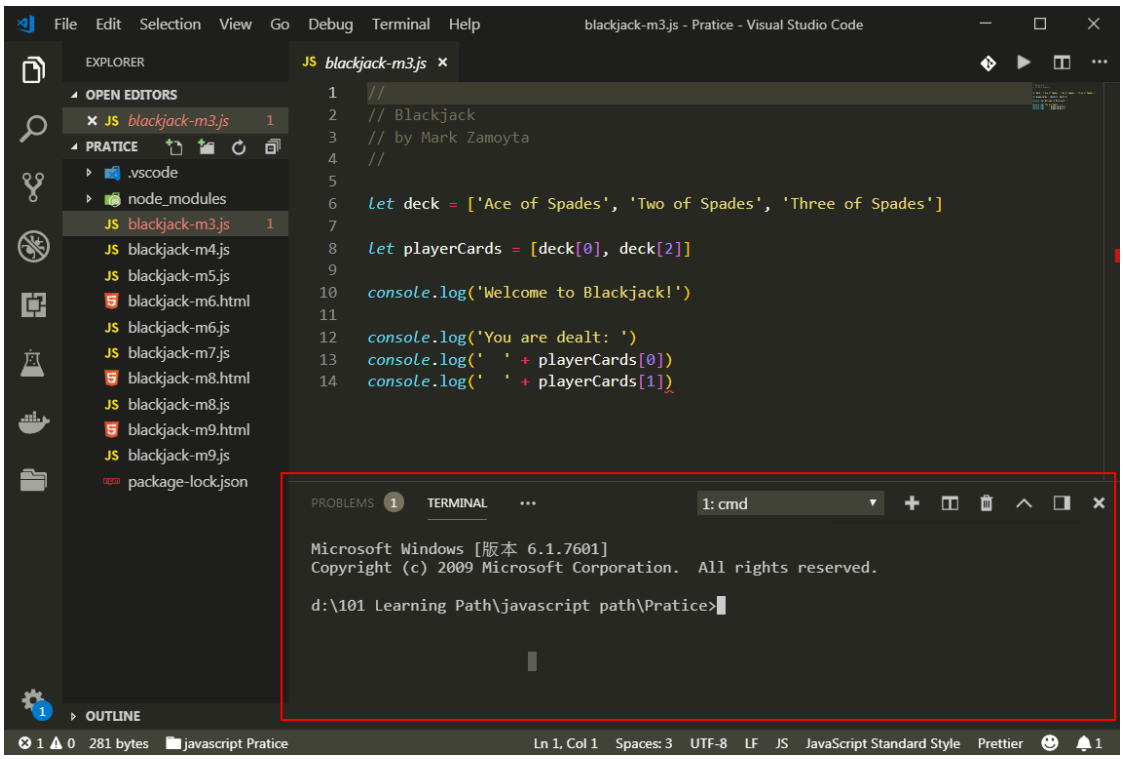

Рисунок 3.7 – Інтерфейс VS Code

Він виділяється як винятковий вибір завдяки широкому спектру функцій. Однією з помітних переваг є те, що він доступний як безкоштовна IDE, що робить його доступним для всіх розробників. Використання Visual

Studio Code значно прискорює процес розробки коду, роблячи його безцінним інструментом.

Для розробки бекенду використовується node.js разом з фреймворком nest.js для більшої ефективності.

Nest (NestJS) — це платформа для створення ефективних, масштабованих серверних програм Node.js. Він використовує прогресивний JavaScript, створений і повністю підтримує TypeScript (все ще дозволяє розробникам кодувати на чистому JavaScript) і поєднує елементи ООП (об’єктно-орієнтоване програмування), FP (функціональне програмування) і FRP (функціональне реактивне програмування).

Під капотом Nest використовує надійні фреймворки HTTP-сервера, такі як Express (за замовчуванням), і за бажанням також може бути налаштований на використання Fastify.

Nest забезпечує рівень абстракції, вищий за ці звичайні фреймворки Node.js (Express/Fastify), але також надає їхні API безпосередньо розробнику. Це дає розробникам свободу використовувати незліченну кількість сторонніх модулів, які доступні для основної платформи.

В останні роки, завдяки Node.js, JavaScript став «лінгва франка» Інтернету для зовнішніх і внутрішніх програм. Це призвело до появи чудових проєктів, таких як Angular, React і Vue, які підвищують продуктивність розробників і дозволяють створювати швидкі, тестовані та розширювані зовнішні програми. Однак, незважаючи на те, що існує безліч чудових бібліотек, допоміжних засобів та інструментів для Node (і серверного JavaScript), жоден із них не вирішує ефективно головну проблему – архітектури.

Nest надає готову архітектуру застосунків, яка дозволяє розробникам і командам створювати високоякісні, масштабовані, слабко пов’язані та легко підтримувані застосунки. Архітектура значною мірою натхненна Angular.

Ви можете переглянути головну сторінку редактора на рис. 3.8.

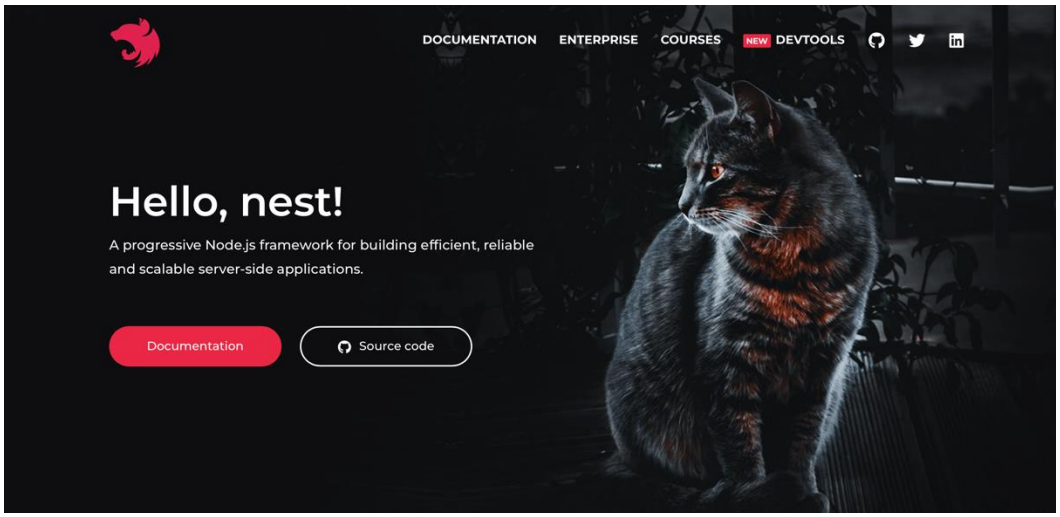

Рисунок 3.8 – Головна сторінка nest.js

NestJS [21] — потужний і популярний фреймворк для створення масштабованих серверних програм за допомогою Node.js. Він поєднує в собі найкращі функції сучасного JavaScript із TypeScript [22], забезпечуючи надійну основу для створення ефективних програм, які легко підтримувати.

Nest.js використовує TypeScript для перевірки типів і надає готову архітектуру програмного забезпечення для створення та розгортання придатних для тестування, масштабованих, слабко зв’язаних і легких в обслуговуванні програм:

– архітектура, NestJS дотримується модульного архітектурного стилю на основі компонентів. Він знаходиться під сильним впливом Angular, що означає, що він пропонує схожу структуру та знайомі концепції, такі як модулі, контролери, сервіси та декоратори. Ця модульна архітектура сприяє організації коду, повторному використанню та тестуванню;

– підтримка TypeScript, NestJS створено за допомогою TypeScript, надмножини JavaScript зі статичною типізацією. TypeScript надає такі переваги, як статична перевірка типів, покращені інструменти та краща читабельність коду. NestJS використовує TypeScript, щоб забезпечити більш структурований і надійний досвід розробки;

– ін’єкція залежностей, NestJS підтримує ін’єкцію залежностей, що дозволяє вам керувати залежностями та впроваджувати їх у свої класи. Це сприяє слабкому зв’язку та полегшує тестування та повторне використання компонентів;

– декоратори та метадані, NestJS широко використовує декоратори та метадані для визначення маршрутів, проміжного програмного забезпечення, захисників та інших аспектів вашої програми. Декоратори надають декларативний спосіб покращити ваш код і спростити складні конфігурації;

– експресивна маршрутизація, NestJS забезпечує надійну та експресивну систему маршрутизації. Ви можете визначати маршрути за допомогою декораторів, що полегшує обробку HTTP-запитів і відповідей. Фреймворк також підтримує різні методи HTTP, зв’язування параметрів, захист маршрутів і проміжне програмне забезпечення.

Оскільки фреймворк використовує TypeScript, Nest.js особливо популярний серед команд, які прагнуть використовувати можливості перевірки типу TypeScript. Більше того, його легко вивчити та освоїти, завдяки потужному інтерфейсу командної команди для підвищення продуктивності та простоти розробки. Завдяки цьому потужному інтерфейсу команди можна легко розпочати будь-який серверний проєкт і довести його до кінця.

Крім того, Nest.js підтримує детальну документацію, а його спільнота розробників і учасників дуже активна та готова реагувати на проблеми миттєво.

Ці причини дозволяють легко зрозуміти, чому так багато компаній змінюють фреймворки на користь Nest.js. Нижче наведено список кількох популярних брендів, які використовують каркас у виробництві:

– потужний, але зручний для користувача, фреймворк зручний для розробників, достатньо простий у використанні навіть найскладнішими та потужними функціями. Команда Nest.js розробила фреймворк, щоб

розробники могли швидко розпочати роботу та зосередитися виключно на написанні бізнес-логіки, тоді як фреймворк піклується про інші важливі аспекти розробки, як-от безпеку;

– синтаксис у стилі Angular (backend), Angular є дуже популярним фреймворком інтерфейсу, який зосереджується на архітектурі та структуруванні. Nest.js діє як Angular для серверної частини, оскільки він використовує стиль і синтаксис Angular, щоб допомогти вам структурувати ваш корпоративний проєкт;

– TypeScript, Nest.js підтримує TypeScript безпосередньо з коробки, і це покращує продуктивність і швидко створює програми, що підходять для обслуговування, надаючи помилки компіляції та попередження. TypeScript робить його добре інтегрованим у VSCode для доступного середовища розробки;

– ретельна документація, Nest.js містить найкращу документацію для будь-якої структури, яку дуже легко зрозуміти. Це економить час налагодження, щоб швидко переглянути документацію та отримати рішення вашої проблеми;

– хороша архітектура та швидка розробка, Nest.js економить ваш час під час створення вашої програми, незалежно від того, створюєте ви свій перший MVP чи фактичну програму, оскільки він дає вам надійну структуру та архітектуру для роботи з воріт, тим самим покращуючи ваші процеси розвитку.

Висновки до розділу 3

Загальний висновок до цього розділу полягає в тому, що було проведено ознайомлення зі сценаріями використання вебзастосунку і розроблено діаграми варіантів використання. Також було представлено діаграму класів для бази даних і надано огляд основних функцій і можливостей системи. Діаграми і описи сценаріїв допомагають зрозуміти, як користувачі будуть взаємодіяти з системою і які функції вони можуть використовувати. Результатом цього розділу є більш чітке уявлення про можливості і потенціал вебзастосунку, що дозволяє краще планувати його розвиток і вдосконалення.

Крім того, в цьому розділі було висвітлено важливі аспекти інтерфейсу користувача, зокрема, представлено рисунок, який демонструє зовнішній вигляд редактора. Інтерфейс користувача відіграє ключову роль у взаємодії з системою, і зрозуміння його структури та функціональних можливостей є важливим для зручного використання вебзастосунку.

Додатково, було розглянуто основні переваги Visual Studio Code як потужного та безкоштовного редактора коду, з підтримкою багатьох мов програмування та функціональних можливостей. Visual Studio Code забезпечує швидку розробку програмного забезпечення та можливості розширення за допомогою розширень, що робить його привабливим вибором для розробників з різними потребами.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РОЗРОБКА ДОКУМЕНТАЦІЇ

4.1 Опис програмної реалізації

Спочатку необхідно налаштувати середовище для програмування перед розробкою. Як було згадано раніше, для розробки ми використовуємо VSCode. Налаштування у цьому інструменті досить просте. Все починається з першого запуску, коли відкривається головне вікно редактора з підказками до гарячих клавіш (див. рис. 4.1). Далі середовище пропонує завантажити додаткові плагіни для більш зручного програмування, які надають підказки та автозаповнення тексту.

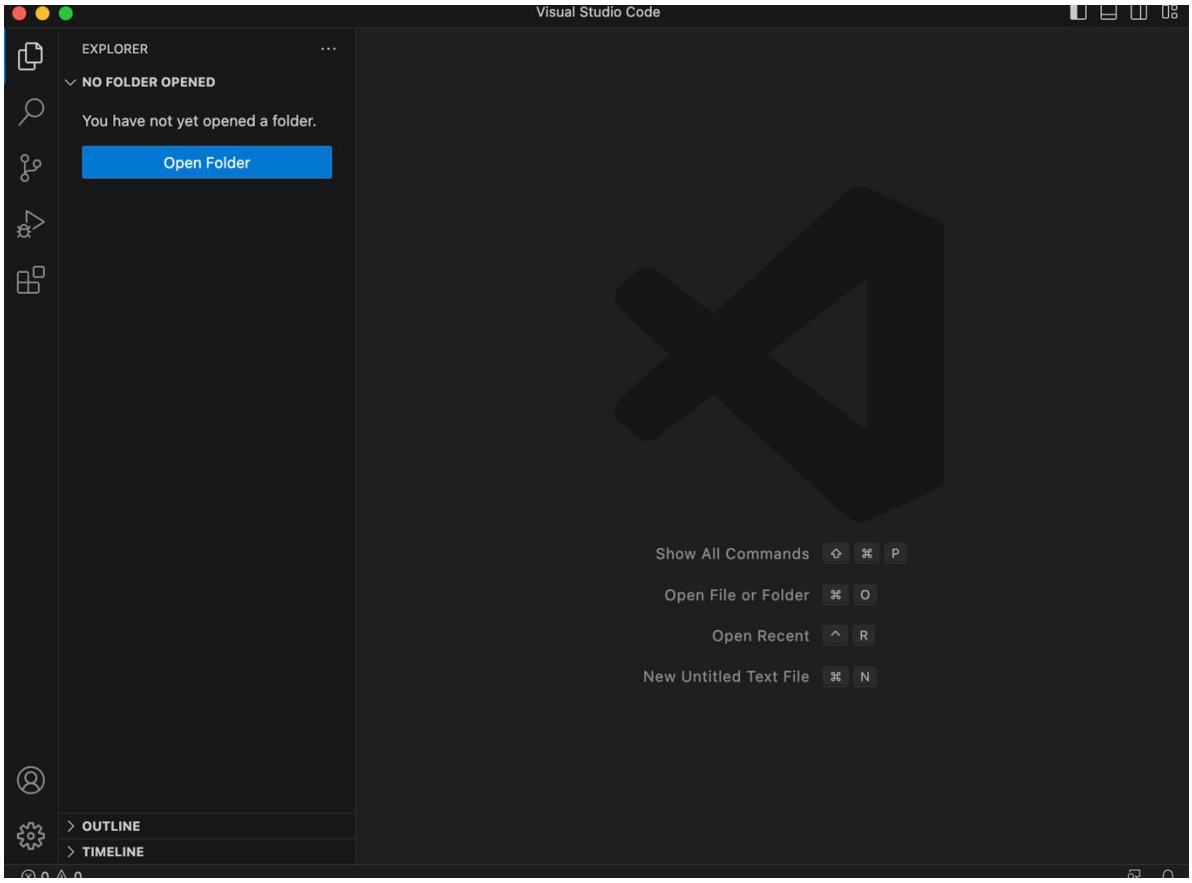

Рисунок 4.1 – Головна сторінка редактору

Наступним кроком налаштування є створення проєкту в редакторі. Завдяки плагінам, середовище негайно пропонує вибрати вебфреймворк. Ми обираємо необхідний нам фреймворк, наприклад, AngularJS, і натискаємо

кнопку "Створити". Після створення нового проєкту і вибору відповідного фреймворку, останній починає завантажуватись та встановлюватись до проєкту.

Вибираємо розташування, де проєкт буде створюватися, і натискаємо кнопку.

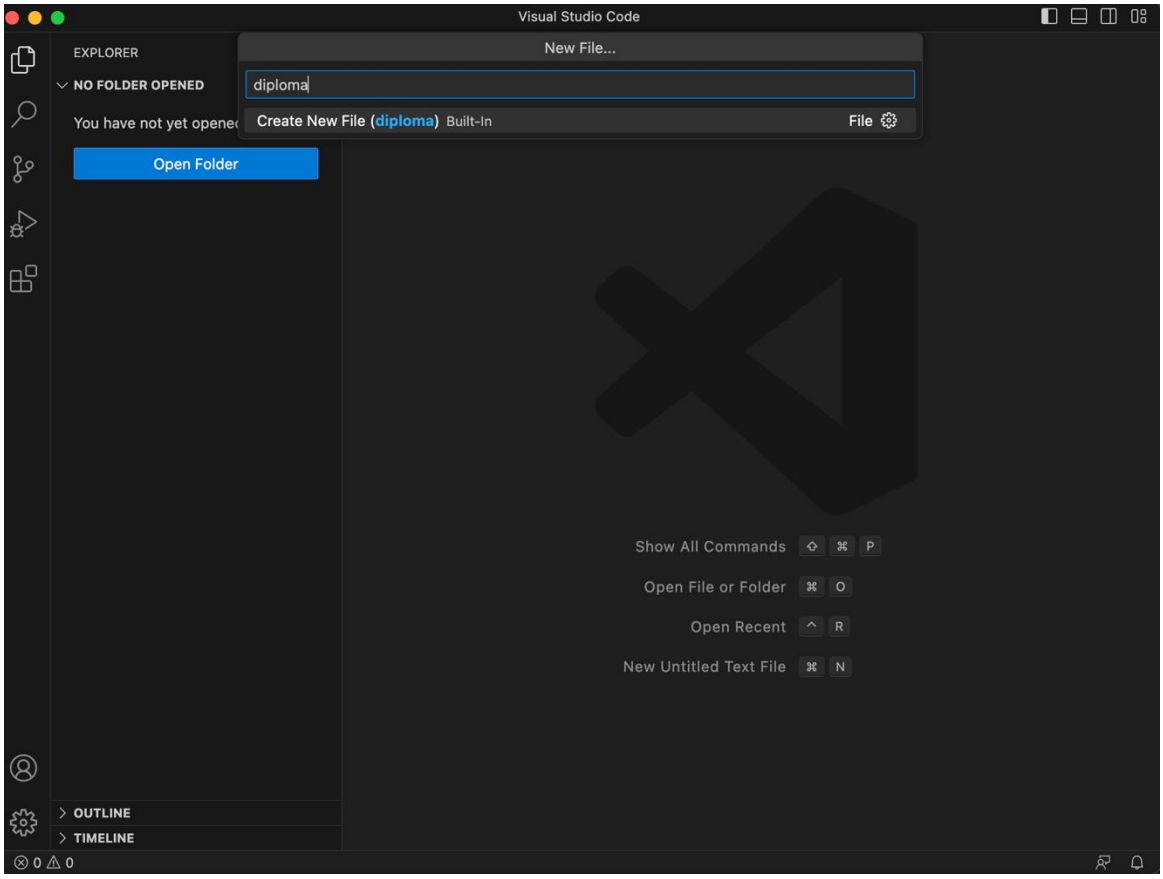

Рисунок 4.2 – Створення нового проєкту

Перед розробкою вебзастосунку також потрібно налаштувати середовище і встановити всі необхідні залежності, бібліотеки та пакети. Це можна зробити за допомогою командного рядка, який має багато альтернатив.

Як командний рядок можна використовувати звичайну консоль, MongoDB, термінал Node.js або консоль OpenServer. Таким чином, першим кроком у розробці клієнтської частини є завантаження та встановлення фреймворку. Для цього використовується пакетний менеджер npm, який однією командою завантажує всі необхідні залежності та встановлює їх.

NPM - це менеджер, який використовується багатьма фреймворками JavaScript для завантаження додаткових модулів та компонентів.

У відкритій консолі редактора вводимо необхідну команду.

```
==> Downloading https://downloads.sf.net/project/machomebrew/Bottles/mongodb-2.6
######################################################################## 100.0%
==> Pouring mongodb-2.6.4.mavericks.bottle.tar.gz
==> Caveats
To have launchd start mongodb at login:
    ln -sfv /usr/local/opt/mongodb/*.plist ~/Library/LaunchAgents
Then to load mongodb now:
    launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mongodb.plist
Or, if you don't want/need launchctl, you can just run:
    mongod --config /usr/local/etc/mongod.conf
==> Summary
🍺  /usr/local/Cellar/mongodb/2.6.4: 17 files, 319M
```

Рисунок 4.3 – Встановлення додаткових бібліотек

Зараз, після успішного встановлення фреймворку та всіх необхідних залежностей, проєкт готовий для написання коду.

Створення директорії для даних, після встановлення MongoDB, створюється директорія, де будуть зберігатися дані бази. Називаємо цю директорію за вибором, наприклад, "data". Далі запускаємо MongoDB сервер у командному рядку, вказавши шлях до виконавчого файлу сервера MongoDB. За замовчуванням, MongoDB працює на порту 27017. Також, налаштовуємо інші параметри, такі як шлях до директорії з даними, логування тощо. Перевірка підключення: запускаємо MongoDB оболонку командної строки, набравши mongo у командному рядку. Використовуючи оболонку MongoDB, створюємо нову базу даних командою use (див. рис. 4.4).

```
const MongoClient = require('mongodb');
const client = new MongoClient('mongodb://localhost:27017');
client.connect((err) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Connected to MongoDB!');
});
```

Рисунок 4.4 – Ініціалізація у проєкті бази даних

Підключення до бази даних встановлене і функціонує.

На початку розробки ми звернемося до файлу app/app.module.ts у нашому проєкті. Цей файл є конфігураційним документом проєкту, в якому міститься основна інформація про проєкт. Він містить назву проєкту, версію, команди для виконання різних функцій, список залежностей, а також додаткові конфігураційні налаштування.

app/app.module.ts є важливим файлом, який визначає основні модулі, компоненти та сервіси, які використовуються у проєкті. Цей файл виконує роль точки входу у програму, де визначаються всі компоненти, які будуть завантажені та використані в проєкті.

Крім того, в app/app.module.ts можна знайти імпорти і налаштування додаткових модулів, необхідних для функціонування проєкту. Це можуть бути модулі для роботи з HTTP-запитами, маршрутизації, формами, аутентифікації та багато інших.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  imports: [
    BrowserModule,
    // import HttpClientModule after BrowserModule.
    HttpClientModule,
  ],
  declarations: [
    AppComponent,
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

Рисунок 4.5 – Відображає вміст розділу файлу конфігурації для клієнтської частини

Клієнтська частина, як було обговорено раніше, базується на фреймворку Angular та використовує мову програмування JavaScript. Для прискорення процесу розробки використовується бібліотека готових елементів NestJS.

При потраплянні на головну сторінку сайту, користувач зустрічає варіант увійти до системи.

Увійшовши до системи, користувач може отримати доступ до особистого профілю, функцій аутентифікації та авторизації, зміни налаштувань акаунту або взаємодії з даними, що вимагають авторизації.

Цей функціонал реалізований за допомогою механізмів аутентифікації, таких як введення облікових даних (логін та пароль) (див. рис. 4.6).

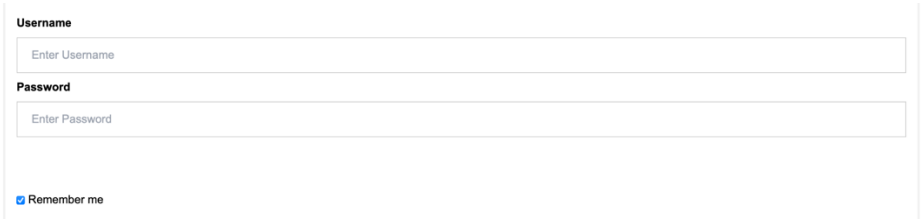

Рисунок 4.6 – Вікно авторизації користувача

Також, з метою зручності, була розроблена кнопка реєстрації, яка надає користувачам можливість створення нового облікового запису. Ця кнопка може бути представлена окремою сторінкою або модальним вікном.

При натисканні на кнопку реєстрації, користувачеві будуть доступні основні поля для заповнення, такі як ім'я, прізвище, електронна пошта, пароль та підтвердження пароля. Додатково можуть бути включені інші поля, які відповідають вимогам проєкту, наприклад, дата народження, номер телефону або інші персональні дані.

Після заповнення обов'язкових полів і натискання на кнопку "Зареєструватися", система може виконати перевірку введених даних, таку як перевірку унікальності електронної пошти або силу пароля. У разі успішної

реєстрації, користувача може бути перенаправлено до вітальної сторінки або до сторінки підтвердження електронної пошти, якщо це необхідно.

Ця функціональність реєстрації дозволяє користувачам створювати облікові записи і використовувати їх для отримання додаткових функцій або доступу до обмеженого контенту на сайті або застосунку.

Після натискання кнопки реєстрації, користувач буде перенаправлений на форму реєстрації, де він зможе введення своїх основних даних для створення облікового запису (див. рис. 4.7).

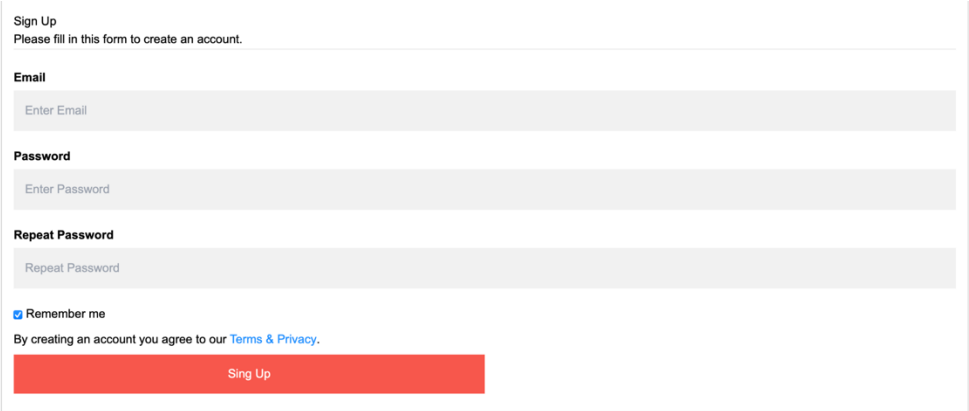

Рисунок 4.7 – Вікно з формою для реєстрації користувача

У коді встановлюються правила валідації для перевірки оголошених полів на відповідність певним правилам. Валідація поля здійснюється згідно цих правил, які можуть включати перевірку наявності значення, його формату, мінімальної або максимальної довжини тощо.

Крім валідації полів, також в коді включені методи для авторизації користувачів. Ці методи забезпечують процес перевірки облікових даних користувача та надання доступу до певних функцій або ресурсів системи. Авторизація може включати перевірку логіну та паролю, перевірку ролей користувача, встановлення сесій або токенів для ідентифікації авторизованого користувача.

Ці функції валідації та авторизації виконують важливу роль у забезпеченні безпеки, відповідності введених даних встановленим вимогам і

контролю над доступом до системи чи її ресурсів. Вони сприяють забезпеченню цілісності та надійності даних, а також допомагають зменшити ризик вразливості системи перед несанкціонованим доступом (див. рис. 4.8).

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const UserSchema = new Schema({
  name: {
    type: String,
    required: true,
    minLength: 3,
    maxLength: 255,
  },
  email: {
    type: String,
    required: true,
    unique: true,
    minLength: 5,
    maxLength: 255,
  },
  password: {
    type: String,
    required: true,
    minLength: 6,
    maxLength: 255,
  },
});
UserSchema.pre('save', function (next) {
  const user = this;
  // Validate the name
```

Рисунок 4.8 – Обробка помилок валідації

Далі була реалізована серверна частина проєкту за допомогою платформи Node.js та фреймворку NestJS. Node.js є середовищем виконання JavaScript на серверному боці, що дозволяє розробникам створювати швидкі та масштабовані серверні застосунки. NestJS, з свого боку, є елегантним та потужним фреймворком, побудованим на основі Node.js, який надає розробникам уніфіковану структуру, механізми роутингу, внедрення

залежностей та багато інших функцій для зручної розробки серверних застосунків.

Використання Node.js та NestJS дозволило розробникам зосередитись на бізнес-логіці та функціональності серверної частини, надаючи потужні інструменти та готові рішення для реалізації різних серверних операцій. Фреймворк NestJS дозволяє розбити серверний код на модулі, контролери та провайдери, що спрощує структуру проєкту та підвищує його читабельність та розширюваність.

Завдяки Node.js та NestJS, серверна частина проєкту стала потужною та ефективною, забезпечуючи обробку запитів, взаємодію з базою даних, автентифікацію, авторизацію та багато іншого. Це дозволило створити надійний та швидкий сервер, який може взаємодіяти з клієнтською частиною та забезпечувати потрібні функції та сервіси для користувачів проєкту (див. рис. 4.9).

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
async function main() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
}
main();
```

Рисунок 4.9 – Забезпечення зв'язку між node.js та nestjs

App.module.ts використовується для конфігурації модулів, контролерів та провайдерів у нашому застосунку. В даному прикладі ми маємо один контролер (AppController) та один провайдер (AppService), які включені в модуль AppModule. App.controller.ts містить код контролера, який визначає HTTP-маршрути та обробляє запити. У даному прикладі ми маємо один

маршрут /, який обробляє GET-запити та повертає повідомлення за допомогою сервісу AppService.

Після встановлення та ініціалізації сервісів у проєкт, потрібно провести налаштування роутінгу для обробки різних запитів. Це можна зробити, додавши в контролер методи, що відповідають на різні HTTP-запити до визначених маршрутів (див. рис. 4.10).

```
import { Controller, Get, Post, Body } from '@nestjs/common';
import { AppService } from './app.service';

@Controller('users') // Базовий маршрут для всіх методів контролера
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getUsers(): string {
    return this.appService.getUsers();
  }

  @Get(':id')
  getUser(@Param('id') id: string): string {
    return this.appService.getUser(id);
  }

  @Post()
  createUser(@Body() userData: any): string {
    return this.appService.createUser(userData);
  }
}
```

Рисунок 4.10 – Http запити маршрутів

getUsers()// Обробляє GET-запит до шляху /users і повертає список користувачів.

getUser(id: string)// Обробляє GET-запит до шляху /users/:id, де :id є параметром, і повертає користувача з вказаним ідентифікатором.

createUser(userData: any)// Обробляє POST-запит до шляху /users і створює нового користувача з переданими даними.

4.2 Керівництво користувача

Наступною реалізованою сторінкою є головна сторінка нашого застосунку.

Як видно, у верхній частині вікна застосунку розміщене меню, яке дозволяє переходити на інші сторінки. Основну частину сторінки займає інформація про психологічну підтримку, яку надає наш застосунок. У верхній частині сторінки розташована панель, де можна виконати вхід до системи або зареєструватись у застосунку (див. рис. 4.11).

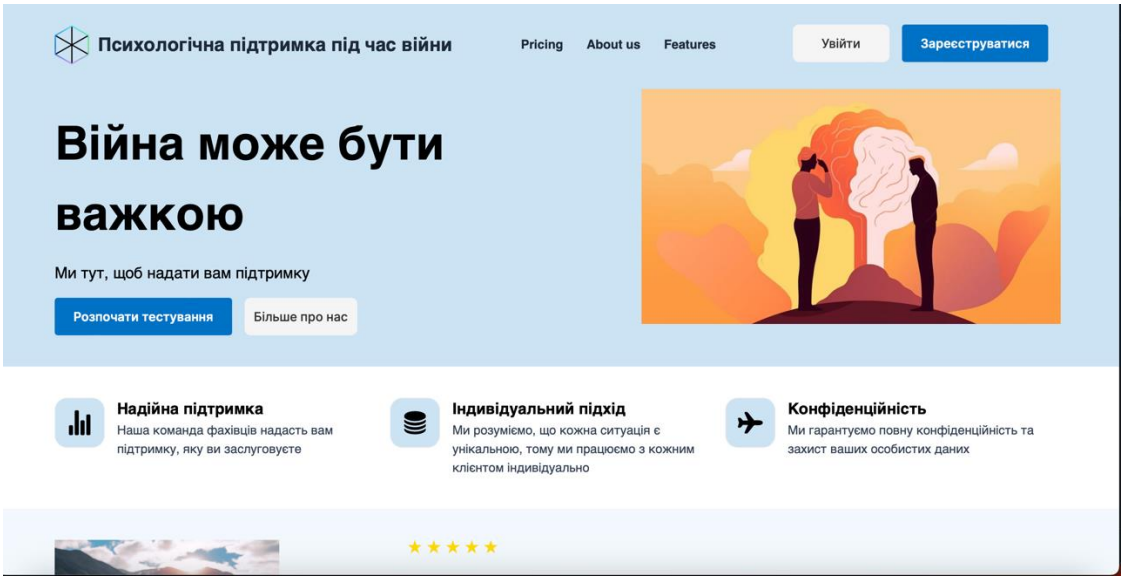

Рисунок 4.11 – Головна сторінка застосунку

Додатково на головній сторінці також доступний тест на психологічне здоров'я. Користувач має можливість пройти цей тест, що допоможе оцінити його психологічний стан та зрозуміти його потреби. Результати тесту допоможуть надати корисні поради та рекомендації для поліпшення психологічного благополуччя.

Після натискання кнопки "Розпочати тестування" користувач буде автоматично перенаправлений на сторінку з тестом. На цій сторінці він зможе проходити тест та відповідати на питання, які стосуються його психологічного стану. Цей тест допоможе зібрати необхідну інформацію для подальшої оцінки та аналізу його психологічного здоров'я (див. рис. 4.12).

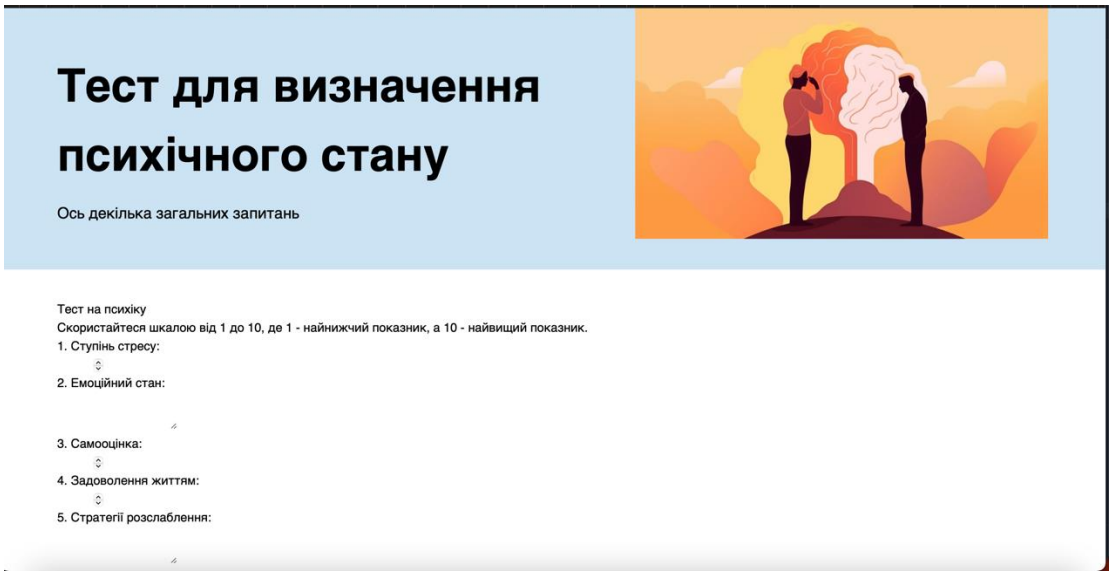

Рисунок 4.12 – Сторінка з тестом для визначення психічного стану

Опускаючись нижче на головній сторінці, користувач також має можливість пройти ще один тест на емоційну стабільність. Цей тест дозволяє оцінити рівень емоційної стабільності користувача. Після завершення тесту, користувачу буде представлена шкала, яка відображатиме його емоційну стабільність на основі відповідей, наданих під час тестування. Ця шкала допоможе користувачу оцінити свій рівень емоційної стійкості та зрозуміти свої емоційні потреби та сильні сторони (див. рис. 4.13).

Рисунок 4.13 – Тест на визначення емоційного стану

Приклад коду який генерує тести (див. рис. 4.14).

```
</head>
<body>
  <h1>Тест на емоційний стан</h1>

  <p>Виберіть відповідь, яка найкраще описує ваш поточний емоційний стан.</p>

  <h4>1. Мені часто сняться нічні кошмари:</h4>
  <input type="radio" name="emotion1" value="2"> Так (+2 бали)
  <input type="radio" name="emotion1" value="1"> Ні (+1 бал)


  <h4>2. Мене дратує велика кількість людей:</h4>
  <input type="radio" name="emotion2" value="3"> Так (+3 бали)
  <input type="radio" name="emotion2" value="0"> Ні (+0 балів)
```

Рисунок 4.14 – Приклад генерації тестів

Далі на головній сторінці розташований розділ, в якому надається онлайн-консультація людям, які потребують допомоги. Цей розділ надає можливість отримати терапевтичну, соціальну та психологічну підтримку через інтернет. Користувачі мають змогу звернутися до кваліфікованих фахівців, які нададуть їм необхідну допомогу та поради. Цей сервіс дозволяє людям знайти підтримку та отримати професійну допомогу у зручний для них спосіб (див. рис. 4.15).

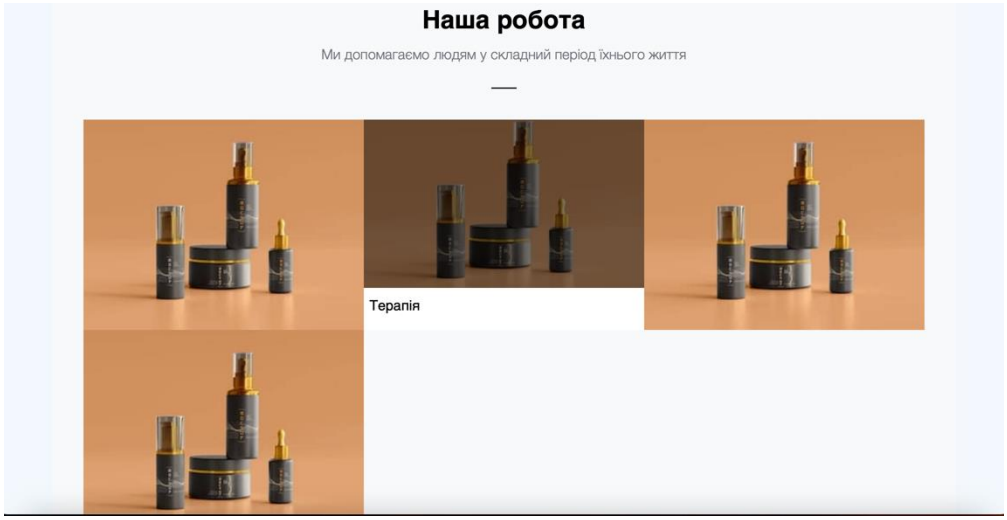

Рисунок 4.15 – Сторінка надання необхідної допомоги

Далі на головній сторінці представлений розділ "Продукти та послуги", в якому користувачам пропонується переглянути різноманітні продукти та

послуги, пов'язані з психологічною підтримкою. Крім того, в цьому розділі надано детальний опис кваліфікації спеціалістів, що працюють на платформі. Користувачі можуть ознайомитись з професійними навичками та досвідом кожного спеціаліста, щоб знайти найбільш підходящого для своїх потреб.

Після цього, на сторінці представлені два типи психологічної підтримки: індивідуальна консультація та групові сесії. Індивідуальна консультація надає можливість користувачам зустрітися зі спеціалістом один на один та отримати особисту підтримку та поради. Групові сесії, з свого боку, створюють сприятливу атмосферу для спільного обговорення проблем та отримання підтримки від інших учасників групи. Обидва типи психологічної підтримки допомагають користувачам знайти підтримку та розв'язати свої проблеми з впевненістю та підтримкою фахівців (див. рис. 4.16).

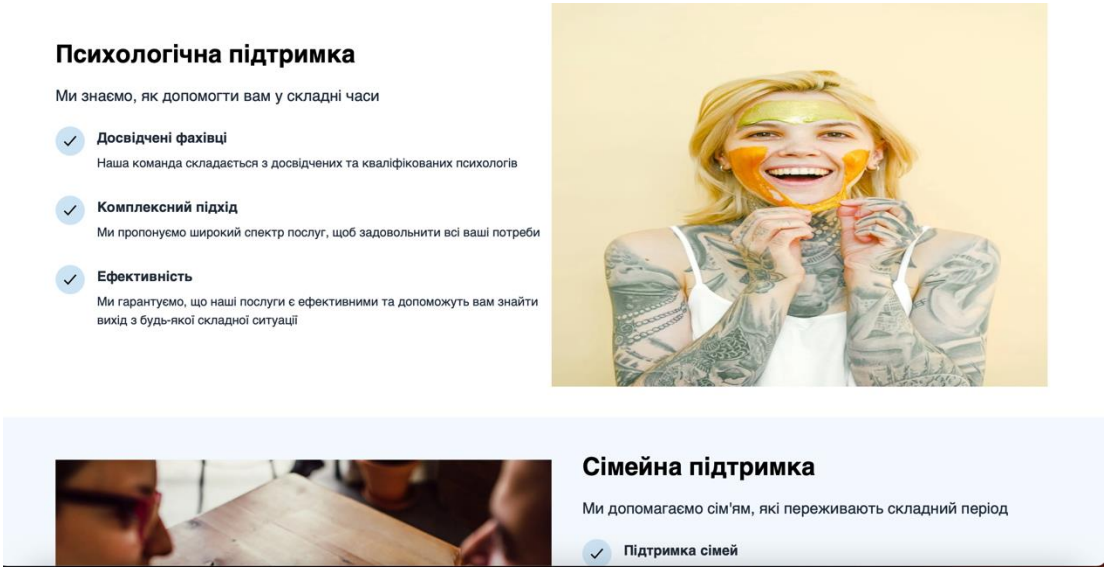

Рисунок 4.16 – Запит надання допомоги

У кінці головної сторінки розміщена контактна інформація, яка дозволяє користувачам зв'язатися з адміністрацією платформи. Тут наведені різні способи зв'язку, такі як електронна пошта, телефонний номер або онлайн-форма зворотного зв'язку. Користувачі можуть відправити своє повідомлення, запитання або звернутися з проханням про допомогу за допомогою зручного для них способу. Адміністрація платформи старатиметься надати швидку та

якісну відповідь на всі запити користувачів. Контактна інформація є важливим елементом для забезпечення зв'язку та подальшої співпраці між платформою та користувачами.

4.3 Інформаційні діаграми

Користувач реєструється в застосунку, надаючи необхідну інформацію, наприклад, особисті дані та контактні дані. Після реєстрації користувач авторизується, вводячи свої облікові дані (логін та пароль) для отримання доступу до функціоналу застосунку.

Застосунок може попросити користувача відповісти на певні запитання або проходити тест для оцінки його потреб у психологічній підтримці під час війни. Це допомагає зрозуміти, які аспекти підтримки можуть бути найбільш корисними для конкретного користувача.

Застосунок надає доступ до різних форм підтримки, таких як онлайн-консультації, психологічні сесії, групові зустрічі, форуми або чати, де користувачі можуть обмінюватися досвідом та підтримувати один одного.

Застосунок надає користувачам доступ до різноманітних ресурсів, які можуть бути корисними під час війни, наприклад, статті, практичні поради, посилання на важливі документи та ресурси з психологічної підтримки.

Застосунок може мати механізми для надання екстреної підтримки в разі кризових ситуацій або випадків, коли користувачі потребують негайної допомоги. Це може включати контактні дані служб невідкладної допомоги або кнопку "Термінова допомога", яка сприяє швидкому зв'язку з фахівцем.

Користувачам може бути надана можливість залишити відгук про якість підтримки, яку вони отримали, що дозволяє вдосконалювати роботу застосунку та забезпечувати якісну підтримку.

Алгоритм роботи застосунку зображений на рис. 4.17.

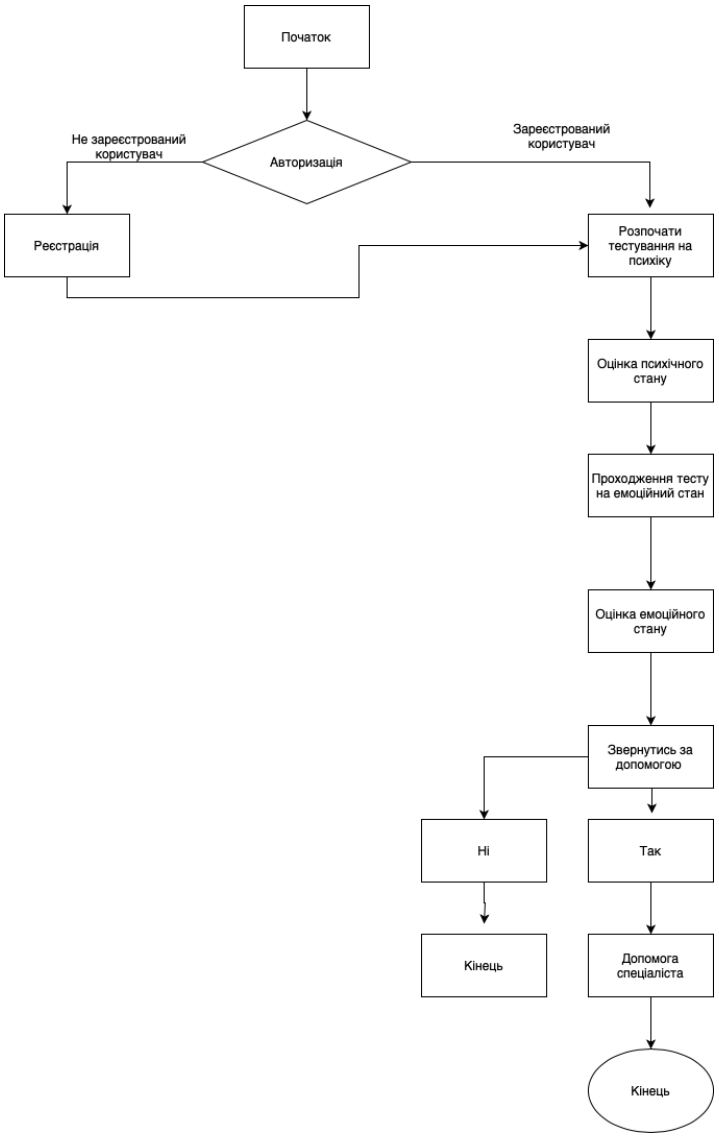

Рисунок 4.17 – Алгоритм роботи застосунку

Інформаційні діаграми можуть бути використані для представлення архітектури системи, процесів бізнесу, взаємодії між компонентами, аналізу даних, структури даних, відображення трендів, визначення пріоритетів та багато іншого. Вони є потужним інструментом для зрозуміння та візуалізації інформації у зручній та доступній формі.

Висновки до розділу 4

У цьому розділі було описано різні аспекти розробки та функціональності психологічної підтримки вебзастосунку. Були розглянуті основні етапи налаштування середовища розробки, ініціалізація бази даних MongoDB та налаштування серверної частини за допомогою Node.js та фреймворку NestJS.

Також було описано головну сторінку застосунку, де користувачі мають можливість пройти тести на психологічне здоров'я та емоційну стабільність. Після цього їм надається доступ до онлайн-консультацій, терапії та соціальної підтримки. Користувачі можуть скористатися послугами кваліфікованих спеціалістів та вибрати підходящий тип психологічної підтримки.

У розділі була також зазначена важливість контактної інформації, що дозволяє користувачам зв'язатися з адміністрацією застосунку. Це забезпечує зручний спосіб комунікації та підтримки користувачів.

В цілому, розроблений вебзастосунок надає комплексну психологічну підтримку, починаючи від тестування та аналізу емоційного стану користувача до надання консультацій та терапевтичної допомоги. Це дозволяє створити зручне та доступне середовище для тих, хто шукає психологічну підтримку та допомогу.

ВИСНОВКИ

Підчас виконання кваліфікаційної роботи було опрацьовано багато матеріалів в області розробки вебзастосунків. Були розглянуті аналоги застосунків які надають допомогу людям під час війни. Також були розглянуті мобільні застосунки, які мають схожий функціонал, схожу ідею та реалізацію. В процесі виконання проєкту, із них було взято найкраще, та усунено недоліки.

Також були виконанні завдання:

– дослідити об’єктну та предметну галузь;

– проаналізувати аналоги, що вже існують;

– розробити вебзастосунок для психологічної підтримки під час війни;

– створити користувацький інтерфейс;

– розробка структури бази даних;

– проєктування програмних інтерфейсів для взаємодії з серверною частиною.

Проєкт вебзастосунку для психологічної підтримки під час війни реалізований успішно, принесе користь своїм користувачам та сприятиме поліпшенню їх психічному стану.

Використання кешу для зберігання та повторного використання популярних або часто запитуваних даних значно зменшило навантаження на базу даних та знизила час відповіді.

Розширення апаратних ресурсів, таких як сервери, бази даних, мережеві компоненти, допомогли розподілити навантаження та забезпечити більшу швидкість та доступність системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Steve Krug, "Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability", vol 12, no 9, pp. 65-159.

2. Matthieu Napoli, " AngularJS Web Application Development Cookbook,", vol. 118, no. 24, pp. 291-299, 2018.

3. Pedro Teixeira, "Professional Node.js: Building Javascript Based Scalable Software" IOP Conference Series: Materials Science and Engineering, vol. 379, no. 1, p. 012099, 2018

4. Szymon Rozga, ""Practical Bot Development: Designing and Building Bots with Node.js and Microsoft Bot Framework", vol 44 , no 2, pp 151-221

5. Bunn, Geoff. "A Short History of The British Psychological Society", vol 22, no 10, pp. 87-121, 2021

6. D. Che, D. Chell and T. Erasmusll , "The Mobile Application Hacker's Handbook", vol 5 , no 7, pp 220-300 (дата звернення: 11.05.2023)

7. Better Help, Платформа, яка надає консультаційні та терапевтичні послуги. URL: https://www.betterhelp.com/about/ (дата звернення: 11.05.2023)

8. Calm, Популярний мобільний застосунок і вебсайт, який надає різноманітні ресурси. URL: https://www.calm.com/ (дата звернення: 12.05.2023)

9. Headspace, Мобільний застосунок і онлайн-платформа, яка пропонує керовану медитацію. URL: https://www.headspace.com/ (дата звернення: 12.05.2023)

10. Wysa, Програма для психічного здоров’я на основі ШІ. URL: https://www.wysa.com/ (дата звернення:12.05.2023)

11. L.Shklar and R.Rosen " Web Application Architecture: Principles, Protocols, and Practices". vol 104, no 2, p 201-210

12. Jeff Patton, "User Story Mapping: Discover the Whole Story, Build the Right Product", vol 5, no 1, p 23-89

13. Dhanji R. Prasanna, "Dependency Injection Principles, Practices, and Patterns", vol 9, no 4, p 55-68

14. E. Gamma, R. Helm, R. Johnson, J. Vlissides "Design Patterns: Elements of Reusable Object-Oriented Software", vol 2, no 5 , p 121-158

15. NPM is the world's largest software registry, URL: https://docs.npmjs.com/ (дата звернення:25.05.2023)

16. Command-Line Interface (CLI), URL: https://nodejs.org/api/cli.html (дата звернення: 25.05.2023)

17. K. Chodorow, E. Horowitz "MongoDB: The Definitive Guide", vol 1, no 7, p 129-190

18. L. Richardson, M. Amundsen, S. Ruby "RESTful Web APIs" , vol 11 no 2, p 24-56

19. Draw.io an online diagramming and flowcharting tool, URL: http://draw.io (дата звернення: 12.06.2023)

20. VSCode is a popular source code editor developed by Microsoft, URL: https://code.visualstudio.com (дата звернення 13.06.2023)

21. NestJS is a popular framework for building scalable and efficient server-side applications , URL: https://nestjs.com (дата звернення 13.06.2023)

22. Boris Cherny, "Programming TypeScript: Making Your JavaScript Applications Scale", vol 11, no 2 , p 77-192

23. Пат. на корисну модель US. Pat. № AU2016298101A1 Software application architecture /, заявл. 23.07.2016, опубл. 04.05.2019

ДОДАТОК А

Програмний код з’єднання сервісів

```
// main.ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule); // Створює екземпляр
додатку за допомогою AppModule
  await app.listen(3000); // Слухає на порту 3000
}
bootstrap();

// app.module.ts
import { Module } from '@nestjs/common';
import { MongooseModule } from '@nestjs/mongoose';
import { AppController } from './app.controller';
import { AppService } from './app.service';
@Module({
  imports: [

MongooseModule.forRoot('mongodb://localhost/nestjs_mongodb_example'),
// Встановлює з'єднання з MongoDB за допомогою MongooseModule
  ],
  controllers: [AppController], // Додає контролер AppController
  providers: [AppService], // Додає провайдер AppService
})
export class AppModule {}

import { Module } from '@nestjs/common';
import { MongooseModule } from '@nestjs/mongoose';;
```

```
@Module({
  @Get() // Декоратор, що вказує, що цей метод обробляє GET-запит  }

  @Post() // Декоратор, що вказує, що цей метод обробляє POST-запит

}

  async findAll(): Promise<Cat[]> {
  }
// app.controller.ts
import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';

@Controller() // Контролер для головного шляху
export class AppController {
  constructor(private readonly appService: AppService) {} // Внедрення
AppService

  @Get() // Декоратор, що вказує, що цей метод обробляє GET-запит
  getQuiz(): string {
AppService
  }
}

// app.service.ts
import { Injectable } from '@nestjs/common';

@Injectable()
export class AppService {
}
```

ДОДАТОК Б

Програмний код Angular UI

```
// Імпортуємо необхідні модулі та класи з Angular
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

// Імпортуємо компоненти, які будуть використовуватись в маршрутизації
import { HomeComponent } from './home.component';
import { AboutComponent } from './about.component';
import { ContactComponent } from './contact.component';

// Визначаємо маршрути за допомогою об'єктів Routes
const routes: Routes = [
  { path: '', component: HomeComponent }, // Для шляху '/' використовується компонент HomeComponent
  { path: 'about', component: AboutComponent }, // Для шляху '/about' використовується компонент AboutComponent
  { path: 'contact', component: ContactComponent }, // Для шляху '/contact' використовується компонент ContactComponent
];

// Використовуємо NgModule для налаштування маршрутизації
@NgModule({
  imports: [RouterModule.forRoot(routes)], // Використовуємо RouterModule для конфігурації маршрутизації та передаємо маршрути
  exports: [RouterModule] // Експортуємо RouterModule для використання його в інших модулях
})
export class AppRoutingModule { }

import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { Document } from 'mongoose';
```

```
// Визначення типу документа для user, який розширює стандартний тип документа Mongoose
export type document = Document;

@Schema()
export class user {
  // Властивість name  з вимогою її обов'язкового наявності
  @Prop({ required: true })
  name: string;

  // Властивість age кота з вимогою її обов'язкового наявності
  @Prop({ required: true })
  age: number;

  // Властивість  з вимогою її обов'язкового наявності
  @Prop({ required: true })
  genderselection: string;
}

// Створення схеми для моделі user за допомогою фабрики схем
export const Schema = SchemaFactory.createForClass;
@Injectable()
export class UsersService {
  // Конструктор сервісу UsersService залежить від внедрення залежності userModel, який представляє модель user
  constructor(@InjectModel(user.name)  private  readonly  UserModel: Model<UserDocument>) {}

  // Асинхронний метод, який повертає масив всіх юзерів
  async findAll(): Promise<User[]> {
    // Використовуємо модель user userModel для пошуку всіх документів із колекції
```

```
    return this.userModel.find().exec();
  }

  // Асинхронний метод для створення нового user
  async create(createUserDto: User): Promise<User> {
    // Створюємо новий екземпляр моделі user на основі отриманого об'єкту
createUserDto
    const createdUser = new this.userModel(createUserDto);
    // Зберігаємо створеного user у базі даних
    return createdUser.save();
  }
}
@Module({
  imports: [
    MongooseModule.forFeature([{ name: user.name, schema: UserSchema
}]),
  ],
  controllers: [UsersController],
  providers: [UsersService],
})
```