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

_____Юрій КОНДРАТЕНКО

«____»_____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ДЛЯ ПСИХОЛОГІЧНОГО ТА НЕЙРОКОРЕКЦІЙНОГО ПРОСТОРУ «АКУЛА»

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Руслан ЛИСИЦЯ

«___»_____ 2025 р.

Керівник канд. фіз-мат. наук, доцент _____ Оксана БРАГІНЕЦЬ

«___»_____ 2025 р.

Миколаїв – 2025

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Лисиці Руслана Сергійовича
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Вебзастосунок для психологічного та нейрокорекційного простору «Акула»».
Керівник роботи: Брагінець Оксана Вікторівна, канд. фіз-мат. наук, доцент.
Затверджена наказом ЧНУ ім. Петра Могили від «18» листопада 2024 р. № 321.
2. Строк представлення кваліфікаційної роботи «___» _____ 2025 р.
3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: Очікуваним результатом роботи є повнофункціональний вебзастосунок для психологічного та нейрокорекційного простору «Акула», який включатиме розділи самодіагностики, інформаційні сторінки про методики, інтерактивну галерею, а також зворотній зв'язок із надсиланням повідомлень на електронну пошту.
Початковими даними є вимоги замовника (нейропсихолога), перелік методик і напрямів роботи простору «Акула», візуальні матеріали (зображення, кольорова палітра, фірмовий стиль), а також інформаційні тексти, надані для наповнення сторінок.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану застосування вебтехнологій у сфері психологічної та нейрокорекційної допомоги; огляд існуючих вебфреймворків і вибір оптимального середовища для розробки MVP вебзастосунку; визначення основного функціоналу системи на основі потреб психологічного простору «Акула» та його цільової аудиторії; побудова структури та інтерфейсу вебзастосунку із врахуванням принципів зручності, візуальної привабливості та доступності; реалізація модулів самодіагностики, галереї та зворотного зв'язку на основі обраної архітектури застосунку; аналіз результатів роботи створеного вебзастосунку, тестування працездатності та зручності використання.

5. Перелік графічних матеріалів: презентація.

Керівник роботи _____ Оксана БРАГІНЕЦЬ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Руслан ЛИСИЦЯ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «20» грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Вебзастосунок для психологічного та нейрокорекційного простору «Акула»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	18.12.2024	20.12.2024	Виконано
2	Аналіз предметної області та постановка задачі	21.12.2024	30.01.2025	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогічних застосунків для психологічної та нейрокорекційної допомоги	31.01.2025	01.03.2025	Виконано
4	Огляд існуючих методів психологічної та нейрокорекційної допомоги дітям	02.03.2025	01.04.2025	Виконано
5	Реалізація вебзастосунку для психологічного та нейрокорекційного простору	02.04.2025	15.05.2025	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	16.05.2025	16.05.2025	Виконано
7	Корегування роботи за результатами попереднього захисту	29.05.2025	29.05.2025	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	30.05.2025	30.05.2025	Виконано
9	Доробка та остаточне оформлення КР	31.05.2025	10.06.2025	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	11.06.2025	11.06.2025	Виконано

Керівник роботи _____ Оксана БРАГІНЕЦЬ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Руслан ЛИСИЦЯ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«30» січня 2025 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 402 ЧНУ ім. Петра Могили

Лисиці Руслана Сергійовича

на тему: **“ ВЕБЗАСТОСУНОК ДЛЯ ПСИХОЛОГІЧНОГО ТА НЕЙРОКОРЕКЦІЙНОГО ПРОСТОРУ «АКУЛА»”**

Актуальність даного дослідження зумовлена стрімким зростанням потреби у доступних цифрових інструментах для раннього виявлення та корекції психоемоційних порушень у дітей дошкільного віку. Вебзастосунок «Акула» покликаний заповнити прогалину між браком фахівців-нейропсихологів та потребою сімей у швидких рекомендаціях, пропонуючи інтерактивну самодіагностику, щоденник настрою та ігрові сенсомоторні вправи, розроблені за методикою сенсорної інтеграції.

Об’єктом роботи є процес розробки інформаційної системи для підтримки психологічної та нейрокорекційної діяльності.

Предметом роботи є архітектура, функціонал та інтерфейс вебзастосунку для підтримки психологічної та нейрокорекційної діяльності.

Метою роботи є розробка вебзастосунку для психологічного та нейрокорекційного простору «Акула», який реалізує базовий функціонал діагностики, демонстрації методик, інтерактивної галереї та зворотного зв’язку без використання бази даних у межах MVP-версії.

В результаті виконання роботи було досліджено rule-based підхід до інтерпретації результатів і доведено його достатність на стартовому етапі, спроєктовано клієнт-серверну архітектуру на Flask із шаблонами Jinja2 і єдиним CSS-файлом, реалізовано шість основних сторінок (Головна, Про нас, Методики, Галерея, Контакти, Тест) та форму зворотного зв’язку з надсиланням електронною поштою та виконано ручне тестування семи критичних сценаріїв, яке підтвердило стабільність,

коректність валідацій і відсутність 404/500-помилок.

Дана робота складається з чотирьох розділів. У першому розділі проведено аналіз предметної сфери, огляд літератури та аналогів, сформовано постановку задачі. Другий розділ присвячений вибору моделей, методів та інформаційних технологій. У третьому розділі наведено результати проєктування й реалізація системи, аналіз отриманих результатів. В четвертому – програмна реалізація, керівництво користувача й тестування. Загальний обсяг роботи – 89 сторінок. Кваліфікаційна робота містить 3 додаток, 7 рисунків і 32 джерел посилання.

Ключові слова: психологічна самодіагностика, сенсомоторна нейрокорекція, вебзастосунок, Flask, rule-based алгоритм, дитячий UX.

ABSTRACT

to the qualification work by the student of the group 402 of Petro Mohyla Black Sea National University

Lysytsia Ruslan

“ WEB APPLICATION FOR THE PSYCHOLOGICAL AND NEUROCORRECTIONAL SPACE «AKULA»”

The relevance of this study is driven by the rapidly growing need for accessible digital tools for the early detection and correction of psycho-emotional disorders in preschool children. The web application "Akula" aims to bridge the gap between the shortage of neuropsychology specialists and the need of families for quick recommendations. It offers interactive self-diagnostics, a mood diary, and gamified sensorimotor exercises developed based on sensory integration methodology.

The object of the study is the process of developing an information system to support psychological and neurocorrectional activities.

The subject of the study is the architecture, functionality, and interface of a web application designed to support psychological and neurocorrectional activities.

The aim of the study is to develop a web application for the psychological and neurocorrectional space "Akula," which implements core functionality for diagnostics, methodology demonstration, an interactive gallery, and feedback—all without the use of a database, within the MVP version.

As a result of the work, a rule-based approach to interpreting results was studied and proven sufficient at the initial stage. A client-server architecture based on Flask with Jinja2 templates and a single CSS file was designed. Six main pages were implemented (Home, About Us, Methods, Gallery, Contacts, Test), along with a feedback form with email

delivery. Manual testing of seven critical scenarios was conducted, confirming stability, correct validation, and the absence of 404/500 errors.

This thesis consists of four chapters. The first chapter provides an analysis of the subject area, a literature review, and an overview of existing solutions, followed by problem statement formulation. The second chapter focuses on the selection of models, methods, and information technologies. The third chapter presents the results of system design and implementation, along with an analysis of the outcomes. The fourth chapter includes the software implementation, user guide, and testing. The total length of the work is 89 pages. The qualification thesis includes 3 appendices, 7 figures, and 32 references.

Keywords: psychological self-diagnostics, sensorimotor neurocorrection, web application, Flask, rule-based algorithm, child UX.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

HTML – HyperText Markup Language – мова розмітки гіпертексту, що використовується для створення веб-сторінок.

CSS – Cascading Style Sheets – каскадні таблиці стилів, що використовуються для оформлення веб-сторінок.

HTTP – Hypertext Transfer Protocol – протокол передачі гіпертексту, що використовується для взаємодії між клієнтом і сервером.

URL – Uniform Resource Locator – уніфікований локатор ресурсу (веб-адреса).

SMTP – Simple Mail Transfer Protocol – простий протокол передачі пошти, що використовується для надсилання електронних листів.

UI – User Interface – інтерфейс користувача.

UX – User Experience – користувацький досвід.

IDE – Integrated Development Environment – інтегроване середовище розробки.

MVC – Model–View–Controller – архітектурний шаблон для побудови програм.

ORM – Object-Relational Mapping – об'єктно-реляційне відображення, що використовується для роботи з базами даних.

SMTP – поштовий сервер, через який передаються електронні листи.

Flask – мікрофреймворк на мові Python для створення веб-застосунків.

Blueprint – механізм у Flask для структуризації коду шляхом розбиття на модулі.

MVP – Minimum Viable Product – мінімально життєздатний продукт.

Jinja2 – шаблонізатор, що використовується у Flask для генерації HTML із динамічними даними.

Bootstrap – CSS-фреймворк, який дозволяє швидко створювати адаптивні веб-інтерфейси.

ВСТУП

У сучасному світі спостерігається зростаюча потреба у підтримці психічного та емоційного здоров’я дітей, особливо у віці до семи років, коли відбувається активне формування когнітивних і сенсорних функцій. Психологічні та нейрокорекційні методики, що базуються на грі, сенсорній інтеграції, розвитку емоційного інтелекту, є потужним інструментом у роботі з дітьми. Але, для їх ефективної реалізації необхідна доступна та інтуїтивна цифрова підтримка, яка дозволить як фахівцям, так і батькам швидко отримувати рекомендації.

Об’єктом роботи є процес розробки інформаційної системи для підтримки психологічної та нейрокорекційної діяльності.

Предметом роботи є архітектура, функціонал та інтерфейс вебзастосунку для підтримки психологічної та нейрокорекційної діяльності.

Метою роботи є розробка вебзастосунку для психологічного та нейрокорекційного простору «Акула», який реалізує базовий функціонал діагностики, демонстрації методик, інтерактивної галереї та зворотного зв’язку без використання бази даних у межах MVP-версії.

Розроблений застосунок орієнтовано на цільову аудиторію – батьків дітей дошкільного віку та фахівців у сфері дитячої психології. Основні завдання роботи включали: аналіз предметної області, проєктування архітектури системи, реалізацію функціоналу без використання бази даних (у межах MVP), налаштування механізму обробки форм, тестування результатів та підготовку системи до подальшого розширення.

Актуальність обраної теми зумовлена зростанням попиту на спеціалізовані онлайн-сервіси у сфері психології та реабілітації, коли цифрові інструменти стають невіддільною частиною допомоги й комунікації між фахівцем і клієнтом. Розробка вебзастосунку надає практичну цінність та слугує реальним прикладом застосування ІТ для підтримки емоційного розвитку дитини.

1 АНАЛІЗ ПОДІБНИХ ВЕБЗАСТОСУНКІВ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної сфери

Психічне здоров’я та розвиток дітей дошкільного віку (до 7 років) є надзвичайно важливою сферою, оскільки саме в ранньому дитинстві закладаються основи подальшого життя. Відомо, що до п’яти років мозок дитини розвивається на ~90% свого обсягу, і ранній досвід формує здатність дитини навчатися, мислити та процвітати. Позитивні чи негативні події раннього дитинства суттєво впливають на формування ключових навичок – від емпатії до самоконтролю. Відсутність належної підтримки або стимуляції в цей період може призвести до того, що необхідні нейронні зв’язки розвинуться не повністю, і надалі виправити це буде значно складніше [1]. Отже, надання психологічної підтримки дітям дошкільного віку є актуальним завданням, яке має довгостроковий вплив на їхній емоційний стан і когнітивний розвиток.

Психологічна підтримка дітей раннього віку передбачає створення безпечного простору та інструментів для розвитку емоційної стійкості, навичок саморегуляції та соціальної адаптації. Діти до 7 років ще не завжди вміють усвідомлювати і виражати свої емоції; тому важливо допомагати їм навчитися розпізнавати власні почуття та реагувати на них конструктивно. Такі навички не є вродженими – їх потрібно тренувати з підтримкою дорослих. Як зазначається в дослідженнях, навчити дитину розпізнавати емоції та приймати рішення у стресових ситуаціях – це навички, які потребують керованої практики так само, як і будь-які інші [2]. Отже, спеціалізовані засоби (наприклад, ігри або вправи) для розвитку емоційного інтелекту дітей є невід’ємною складовою психологічної підтримки. Окрім роботи з дитиною, важливо підтримати і батьків, адже батьки мають розуміти емоційний стан малюка та сприяти його психічному благополуччю.

Нейрокорекція – це напрям психологічної допомоги, спрямований на розвиток

та корекцію вищих психічних функцій дитини шляхом спеціальних вправ і методик. Зокрема, сенсомоторна нейрокорекція (вона ж методика сенсорної інтеграції) базується на принципі покращення обробки мозком сенсорної інформації через рухові й чуттєві стимули. Спеціально підібрані сенсомоторні вправи здатні зміцнювати нейронні зв'язки у мозку дитини та сприяти формуванню нових; вони стимулюють роботу кори головного мозку, глибинних структур та обох півкуль і покращують їх взаємодію. Результатом таких впливів є прогрес як у моторній (руховій), так і у пізнавальній сферах розвитку дитини [3]. Метод сенсорної інтеграції був розроблений А. Дж. Айрес ще у 1970-х роках для допомоги дітям із порушеннями сенсорної обробки (зокрема, дітям з аутизмом або гіперактивністю) і передбачає ігрові заняття, що поєднують різні відчуття – рівновагу, дотик, рух тощо. Наприклад, у терапії використовуються гойдалки, балансувальне обладнання, тактильні стимули (щітки, масаж, "важкі" жилети), які можуть заспокоювати тривожних дітей та підвищувати їхній поріг толерантності до сенсорних навантажень [4]. Отже, нейрокорекційні методики сенсорної інтеграції дозволяють скоригувати особливості розвитку нервової системи дитини через гру і фізичну активність, допомагаючи дитині краще адаптуватися до навколишнього середовища.

В сучасних умовах стрімкого розвитку технологій, виникає можливість перенести елементи психологічної підтримки та нейрокорекції у веб-простір. Інтерактивні вебзастосунки можуть стати зручним доповненням до традиційної роботи психолога чи нейропсихолога, надаючи батькам і дітям інструменти для самостійної діяльності вдома. Зокрема, вебзастосунок "Акула" задуманий як онлайн-платформа психологічного та нейрокорекційного простору, орієнтованого на дітей до 7 років і їхніх батьків. Назва "Акула" асоціюється з цікавою дитині тематикою (морський світ, тварини) і покликана привернути увагу малечі в дружній формі.

Платформа передбачає кілька ключових функцій для підтримки розвитку дитини:

– самодіагностика модуль, що дозволяє швидко оцінити поточний стан або ризики у розвитку дитини. За допомогою простих опитувальників чи тестів, які заповнюють батьки (або спільно з дитиною, якщо це можливо), можна виявити певні сигнали: наприклад, рівень тривожності, наявність сенсорних труднощів, емоційний настрій тощо. Самодіагностика не ставить клінічного діагнозу, але дає миттєвий результат – попередній висновок чи індикатор, чи варто приділити увагу тій чи іншій сфері. Важливо, що вебзастосунок не зберігає історію таких проходжень (кожна сесія оцінювання є конфіденційною і одноразовою), що захищає приватність сім'ї та спрощує використання без реєстрації;

– тести та опитувальники – розширений набір психологічних ігор або завдань, спрямованих на різні аспекти розвитку. Наприклад, це можуть бути прості вікові опитувальники розвитку (визначення рівня мовного розвитку, уваги, пам'яті), тести на емоційний інтелект для дітей (через картинки із ситуаціями і варіанти реакцій) тощо. Вбудовані алгоритми аналізують відповіді і одразу надають рекомендації батькам: наприклад, якщо виявлено труднощі з концентрацією уваги, система може порадити ігри на розвиток уваги; якщо в тесті настрою виявлено часті погані настрої – порадити консультацію спеціаліста або заспокійливі вправи.

Концепція вебзастосунку передбачає опору на сенсомоторну нейрокорекцію у наданні рекомендацій. Тобто, платформа "Акула" буде пропонувати батькам та дітям прості сенсорні ігри чи вправи як реакцію на результати самодіагностики чи щоденника. Наприклад, якщо дитина виявила ознаки тривожності, система може запропонувати вправу з глибоким диханням і одночасним обійманням "важкої" подушки (елемент глибокого тиску для сенсорної стабілізації) або гру, де треба тиснути на іграшковий пластилін (тактильна стимуляція для заспокоєння). Якщо в тестах видно проблеми з дрібною моторикою – порекомендує гру для пальців (малювання пальчиками, сортування дрібних предметів під наглядом батьків). Таким чином, методика сенсорної інтеграції пронизує контент платформи: вебзастосунок не

просто оцінює стан, але й надає практичні поради у вигляді ігор і вправ, заснованих на науково обґрунтованих підходах нейрокорекції. Слід зазначити, що вебзастосунок не замінює професійного втручання повністю. В ідеалі, його використання відбувається під кураторством нейропсихолога або консультанта: фахівець допомагає розробити зміст тестів, інтерпретує нестандартні результати та може консультувати батьків за підсумками використання платформи. Однак безпосередньо автором і розробником платформи є фахівці в галузі ІТ, тоді як психолог виступає експертом-консультантом. Такий підхід забезпечує надійність методик (контроль якості з боку психолога) і водночас зручність та масштабованість ІТ-рішення. Окремої уваги потребує інтерфейс і дизайн вебзастосунку, зважаючи на дитячу аудиторію. Діти дошкільного віку сприймають інформацію переважно образно, їм потрібні яскраві візуальні елементи та простота у взаємодії. Дизайн платформи “Акула” має бути візуально дружнім: використовувати тематику, цікаву дітям (морські тварини, казкові персонажі), і м’яку кольорову палітру. Рекомендовано застосувати теплі, приємні кольори – ніжно-рожевий, жовтий, світло-зелений як основні акценти. Надто яскраві чи контрастні поєднання слід дозувати, щоби не перевантажувати сенсорну систему дитини. Дослідження у сфері дизайну для малят зазначають, що найменші діти (2–3 роки) віддають перевагу яскравим простим кольорам і чітким формам, але для ширшого віку дошкільнят доречно поєднувати яскравість із м’якими відтінками, створюючи заспокійливу атмосферу. Інтерактивні елементи (кнопки, іконки) повинні бути великі і зрозумілі, бажано з графічними підказками (малюнками емоцій, символами вправ). Таким чином, інтерфейс “Акули” стане інтуїтивним і привабливим як для дітей, так і для їхніх батьків, що спонукає до регулярного використання платформи. Отже, предметна сфера охоплює з одного боку психологічні та нейропсихологічні аспекти розвитку дитини (емоційна підтримка, когнітивна корекція, сенсорна інтеграція), а з іншого – засоби їх реалізації у вигляді вебзастосунку. Рішення на стику цих галузей повинно бути науково обґрунтованим,

безпечним для дитини, зручним для батьків та технічно доступним (через Інтернет, на типових пристроях). Вебзастосунок “Акула” прагне об’єднати ці компоненти, надаючи сім’ям з малими дітьми інноваційний інструмент для покращення психічного благополуччя та розвитку малечі.

1.2 Огляд та аналіз наявних аналогів і публікацій

Розробці платформи “Акула” передував аналіз існуючих рішень у сфері психологічної допомоги та нейрокорекції дітей, а також вивчення наукових публікацій за цією тематикою. Були розглянуті як іноземні, так і українські аналоги – від спеціалізованих мобільних застосунків до онлайн-платформ та дослідницьких проєктів. Наведемо короткий огляд найбільш релевантних прикладів і висновків з літератури.

Існуючі платформи для психологічної підтримки дітей:

– Regul-A – інноваційний мобільний застосунок, створений в Португалії для допомоги сім’ям, які виховують дітей з розладом аутистичного спектра (віком 3–6 років). Метою застосунку Regul-A є навчити батьків застосовувати сенсорні стратегії для регулювання стану дітей у повсякденних домашніх рутинах. Іншими словами, цей інструмент пропонує батькам ідеї, як через сенсорні стимули (дотик, рух, звук) допомогти дитині заспокоїтися або сконцентруватися під час типових ситуацій вдома (прийом їжі, підготовка до сну, ігри). Застосунок було розроблено за участю команди ерготерапевтів та IT-спеціалістів, що забезпечило високу якість контенту і зручність дизайну [5]. Досвід Regul-A показує, що вузько спеціалізовані інструменти (в даному випадку – для сенсорної підтримки дітей з аутизмом) можуть ефективно доповнювати традиційну терапію, залучаючи батьків до активної ролі. Для проекту “Акула” цей приклад цінний застосуванням науково обґрунтованої методики (сенсорної інтеграції) у форматі зрозумілому для неспеціалістів (батьків);

– WonderTree – міжнародна платформа, що надає серію ігор з використанням доповненої реальності (AR) для дітей з особливими потребами. WonderTree позиціює себе як “гейміфіковані адаптації перевірених вправ та навчальних матеріалів” і стверджує, що їхні ігри сприяють розвитку різноманітних навичок у дітей з розладами уваги, аутизмом, ДЦП тощо . Наприклад, гра через веб-камеру може спонукати дитину виконувати рухи (підстрибувати, махати руками), які відповідають терапевтичним вправам для покращення рівноваги чи координації. Платформа співпрацює з експертами та має підтримку авторитетних організацій (зокрема UNICEF)[6], підкреслюючи клінічну валідність свого підходу. Для “Акули” WonderTree є прикладом успішної інтеграції технологій у сферу дитячого розвитку: воно демонструє, що ігрова форма подачі терапевтичних завдань значно підвищує мотивацію дітей і регулярність виконання вправ. Хоча “Акула” не використовуватиме складну AR, принципи гейміфікації та дружнього інтерфейсу подібні;

– додатки для відстеження настрою дітей: існує низка мобільних застосунків, орієнтованих на розвиток емоційної сфери у дітей. Для наймолодших (3–5 років) прикладом є “Emotions and Feelings Chart” – простий щоденник настрою на iPad/iPhone, де кожен член сім’ї має профіль і дитина щодня обирає значок, що відповідає її настрою. Великі яскраві піктограми емоцій дозволяють навіть дошкільнятам самостійно фіксувати свій стан, що дає їм відчуття контролю та “володіння” своїми почуттями. Для дітей старшого дошкільного та молодшого шкільного віку (5–10 років) популярним є застосунок “Stop, Breathe & Think Kids”, який поєднує облік настрою з короткими анімованими медитаціями та вправами уважності. Кожного разу дитина обирає, як вона себе почуває, а програма підбирає відповідну “пригоду” – інтерактивну історію чи гру, що навчає заспокійливим технікам. Діти отримують нагороди (наклейки) за пройдені практики, а батьки можуть переглядати, які емоції дитина відзначала і які вправи виконувала. Такий формат не лише відстежує настрій, але й навчає навичкам саморегуляції через гру. Ще один

приклад – Zones of Regulation (заснований на відомій програмі “Зони регуляції”), що допомагає дітям розуміти, до якої “зони” (стану) належить їхня емоція і як перейти до “зеленої” (спокійної) зони. Ці аналоги підтверджують, що цифрові щоденники та інтерактивні вправи можуть успішно використовуватися для розвитку емоційного інтелекту. Водночас, більшість з них – мобільні додатки англійською мовою, і не всі адаптовані для українських реалій, що підкріплює потребу у створенні локалізованого рішення на кшталт “Акули”;

– платформи психологічної допомоги онлайн: Окрім ігрових застосунків, існують веб-платформи, які надають доступ до психологічних послуг для дітей та батьків. В українському контексті варто згадати проект Ucare.Kids, запущений після початку війни 2022 року. Це платформа, місія якої – відновити психічне здоров’я дітей та підлітків, що постраждали від воєнних дій, шляхом надання безкоштовних професійних консультацій. На Ucare.Kids родина може отримати до 12 онлайн-сесій з кваліфікованим психологом або психотерапевтом, які допомагають подолати травматичний досвід. Проект акцентує на тому, що ~75% українських дітей зазнали різного роду психологічних травм від війни, а 20% (понад 1,5 млн дітей) мають ризики розвитку депресії, тривожності, посттравматичного стресу та інших розладів. Водночас близько 40% дітей не змогли вчасно отримати необхідну допомогу через брак спеціалістів чи недоступність послуг [7]. Ці дані свідчать про колосальну потребу в нових формах надання психологічної допомоги, зокрема дистанційних. Хоча Ucare.Kids орієнтована на пряме спілкування з терапевтом (онлайн або офлайн), вона підкреслює важливість також роботи з батьками та родиною для підвищення стійкості дітей. Вебзастосунок “Акула” має інше позиціонування – не як сервіс консультацій, а як інструмент самодопомоги та просвіти. Проте обидва підходи взаємодоповнювані: “Акула” може слугувати першим кроком, превентивним засобом, який допоможе виявити проблеми та надати базові рекомендації, тоді як у складніших випадках родина звернеться до фахівців (наприклад, через ті ж платформи на кшталт

Ucare.Kids). Також помітимо, що “Акула” охоплює дещо молодшу аудиторію (діти до 7 років), тоді як більшість існуючих сервісів акцентують або на підлітках, або загалом на дітях шкільного віку.

Наукові публікації та дослідження: Окрім огляду готових продуктів, було проаналізовано актуальні дослідження, які підтверджують дієвість або необхідність компонентів, закладених у концепцію “Акули”. Зокрема:

– за даними Всесвітньої організації охорони здоров’я та профільних досліджень, до 10–20% дітей у світі страждають на психічні, емоційні або поведінкові розлади [8]. Це означає, що в середньому кожна п’ята дитина може потребувати допомоги у сфері психічного здоров’я – чи то професійної, чи то у вигляді підтримуючих заходів. Отже, розробка програм раннього виявлення та м’якої корекції (як-от через ігрові вебзастосунки) є виправданою глобальною тенденцією;

– у дослідженнях, присвячених онлайн-інструментам для дітей з особливими потребами, відзначається ефективність поєднання технологій і залучення батьків. Наприклад, використання мобільних застосунків для сенсорної стимуляції дітей з аутизмом показало позитивні результати у плані залученості сім’ї та покращення адаптивної поведінки дітей. Інший напрям – біологічний зворотний зв’язок у іграх: комерційні проекти на кшталт Mightier (США) використовують ігри з контролем серцевого ритму дитини, навчаючи її заспокоюватися, щоб знизити пульс під час гри. Клінічні оцінки цього підходу показали зменшення проявів гніву і тривожності у дітей 6–14 років після кількох місяців тренувань з грою (за даними розробників, 84% батьків відзначили покращення емоційного стану дітей) [9]. Цей приклад хоча і стосується дещо старших дітей, демонструє принцип: інтерактивність + науковий підхід дають відчутний терапевтичний ефект;

– важливим аспектом, висвітленим у літературі, є питання конфіденційності та етичності при роботі з даними дітей. Фахівці наголошують, що цифрові продукти для дітей мають убезпечувати персональну інформацію та уникати зберігання чутливих

даних без необхідності. Це узгоджується з рішенням у “Акулі” не зберігати історію проходжень: усі результати видаються користувачу разово, після чого видаляються. Такий підхід мінімізує ризики витоку інформації і спрощує питання згоди батьків на використання (немає довготривалого збору даних про здоров’я дитини);

– також дослідження після пандемії COVID-19 та в умовах воєнного часу (як у Україні) показують зростання ролі дистанційних рішень: телемедицина і телепсихологія для дітей зросли у використанні в десятки разів. Це підкреслює готовність суспільства приймати онлайн-формат допомоги. Проте в онлайн-середовищі особливо важливо підтримувати залученість: просте інформування батьків про проблеми дитини не дає ефекту, якщо не забезпечити інструменти для дії. Тому в “Акулі” акцент робиться на інтерактивності (щоденник, тести) та негайному зворотному зв’язку (поради, ігри), щоб користувачі активно взаємодіяли з платформою, а не пасивно споживали інформацію.

Загалом, аналіз аналогів підтвердив, що на ринку є фрагментарні рішення, які реалізують окремі функції, схожі на задумані в “Акулі”: є програми для настрою, є ігри для сенсорної інтеграції, є платформи консультацій. Але комплексного веб-застосунку, орієнтованого саме на українських дошкільнят та інтегруючого самодіагностику, щоденник та нейрокорекційні ігри – наразі не виявлено. Це створює нішу, яку покликана заповнити “Акула”. Отримані з огляду ідеї та зауваження (щодо дизайну, методик, безпеки) будуть враховані при постановці задачі та безпосередній розробці платформи.

1.3 Постановка задачі

Актуальність. Потреба в забезпеченні психологічного благополуччя дітей дошкільного віку є надзвичайно актуальною як у світі, так і в Україні. Ранні втручання у випадку емоційних чи розвиткових проблем значно підвищують шанси успішної корекції та адаптації дитини в подальшому житті. Натомість ігнорування перших

тривожних сигналів може призвести до закріплення проблемної поведінки або загострення розладів у шкільному віці. На жаль, далеко не всі сім'ї мають швидкий доступ до кваліфікованих дитячих психологів чи нейропсихологів – особливо це стосується віддалених районів чи періодів криз (як от пандемія чи воєнні дії). Зокрема, в Україні на тлі війни спостерігається різке збільшення кількості дітей, які потребують психологічної допомоги: за оцінками UNICEF, близько 1,5 мільйона українських дітей перебувають у зоні ризику розвитку депресії, тривожних розладів, ПТСР та інших проблем психіки [10]. В той же час традиційна система допомоги не встигає охопити всіх постраждалих – значна частина дітей не отримує вчасної підтримки через нестачу спеціалістів. За таких умов, впровадження технологічних рішень у сферу дитячої психології є своєчасним кроком. Вебзастосунок "Акула" актуальний тим, що він пропонує доступний, інтерактивний та науково обґрунтований інструмент, яким можуть скористатися будь-які батьки з доступом до Інтернету. Поєднання самодіагностики, навчальних ігор та порад за методиками нейрокорекції дозволить заповнити прогалину між відсутністю допомоги і професійною терапією – надавши родині хоча б базовий рівень підтримки тут і зараз. "Акула" відповідає стратегії розвитку e-health і m-health (електронного здоров'я) у психічній сфері, адаптуючи їх до потреб найменших користувачів. Актуальність підсилюється й тим фактом, що в інформаційному просторі України мало локалізованих продуктів для розвитку дошкільнят із фокусом на психічне здоров'я – більшість наявних додатків іноземною мовою або ж загальноосвітнього характеру. Тому розробка вітчизняного вебзастосунку з урахуванням культурних особливостей, мови та реалій є важливим і суспільно значущим завданням.

Мета роботи. Метою даної роботи є розробка вебзастосунку "Акула" для психологічного та нейрокорекційного простору, орієнтованого на дітей дошкільного віку та їхніх батьків. Застосунок повинен забезпечувати користувачам можливість проведення самодіагностики розвитку та емоційного стану дитини, проходження

інтерактивних тестів і вправ, ведення щоденника настрою, а також отримання миттєвих результатів і рекомендацій на основі сенсомоторних нейрокорекційних методик. Реалізація мети передбачає створення дружнього до дітей інтерфейсу, застосування сучасних веб-технологій (HTML, CSS для фронтенду та Python для бекенду) та залучення експертних знань з дитячої психології при розробці контенту.

Об’єкт роботи. Об’єктом даної роботи є процес надання психологічної підтримки та нейропсихологічної корекції дітям дошкільного віку з використанням засобів інформаційних технологій. Іншими словами, об’єкт охоплює сам феномен взаємодії дитини і батьків з цифровим інструментом, призначеним для покращення емоційного та когнітивного стану дитини.

Предмет роботи. Предметом роботи є методи та програмні засоби реалізації вебзастосунку для психологічної підтримки і нейрокорекції дітей. Сюди входять: архітектура та функціональні компоненти вебзастосунку “Акула”, підходи до розробки користувацького інтерфейсу для дітей, алгоритми обробки результатів тестування та формування рекомендацій, а також практичні прийоми інтеграції нейрокорекційних методик (сенсорної інтеграції) у форматі онлайн-взаємодії.

Завдання. Відповідно до поставленої мети, у межах роботи необхідно вирішити такі завдання.

1. Провести аналіз предметної області – вивчити психологічні особливості розвитку дітей до 7 років, принципи нейропсихологічної корекції (зокрема сенсомоторної) та існуючий досвід застосування цифрових технологій у цій сфері. Зібрати вимоги користувачів (батьків та фахівців) до функціоналу та інтерфейсу подібного вебзастосунку.

2. Здійснити огляд аналогів і літератури (що виконано у розділі 1.2) та на основі нього сформувати перелік необхідних функцій платформи “Акула” і критерії успішності її реалізації. Визначити, які саме тести, опитувальники та ігрові вправи будуть інтегровані, та як забезпечити наукову коректність їх використання.

3. Розробити концептуальну модель та архітектуру вебзастосунку. Спроєктувати структуру системи, включно з модулем фронтенду (інтерфейс користувача, сторінки для самодіагностики, щоденника тощо) та бекенду (логіка обробки даних, видача рекомендацій, збереження мінімально необхідної інформації). Обрати технічні засоби реалізації: мову програмування та фреймворк для сервера (Python, можливе використання Flask/Django), базові веб-технології для клієнтської частини (HTML5, CSS3) і забезпечення їх взаємодії.

4. Реалізувати прототип вебзастосунку “Акула” згідно з проєктом. Створити веб-сторінки з адаптивним дитячим дизайном (використовуючи принципи UX/UI для дітей), імплементувати основний функціонал: опитувальники і тести з обробкою відповідей, інтерфейс щоденника настрою, генерацію результатів і рекомендацій. Забезпечити відсутність збереження персональних даних дитини (окрім тимчасових даних сесії) та реалізувати опцію друку або збереження результатів для користувача, якщо це потрібно.

5. Залучити експерта (дитячого нейропсихолога) для перевірки коректності змістовної частини прототипу. Отримати відгуки щодо відповідності тестів і рекомендацій заявленим методикам нейрокорекції, а також щодо зручності інтерфейсу для цільової аудиторії.

6. Провести тестування і оцінку прототипу. Перевірити працездатність усіх модулів, протестувати сценарії використання (імовірні шляхи взаємодії батьків і дітей із застосунком). Виміряти, чи зрозумілі результати для користувачів, чи достатньо дружнім є дизайн. За можливості, залучити декілька сімей з дітьми дошкільного віку для апробації прототипу та збору зворотного зв’язку.

7. На основі результатів тестування та відгуків підготувати рекомендації щодо подальшого вдосконалення вебзастосунку “Акула”. Визначити перспективи розвитку функціоналу (наприклад, додавання нових тестів, підтримка різних мов, мобільна

адаптація), окреслити, які компоненти потребують доопрацювання для промислового впровадження платформи.

Виконання перелічених завдань дозволить досягти поставленої мети – створити і перевірити працездатність вебзастосунку для психологічної підтримки та нейрокорекції дітей раннього віку. У результаті роботи буде отримано прототип платформи “Акула” та методичні рекомендації щодо її застосування у практиці (батьками самостійно чи у співпраці з фахівцями). Це стане внеском у розвиток доступних цифрових рішень для забезпечення психічного здоров’я дітей в Україні.

Висновки до розділу 1

Проведений аналіз предметної області підтвердив актуальність розробки вебзастосунку “Акула” та окреслив науково-практичні основи для його створення. Огляд аналогів і публікацій показав, що існують окремі рішення, але немає їхнього повного аналога, що підкреслює новизну проекту. У підрозділах 1.1–1.2 було сформовано розуміння ключових вимог до функціоналу, дизайну та методичного наповнення платформи. Постановка задачі (підрозділ 1.3) визначила чіткий план реалізації мети, включаючи етапи проектування, розробки та оцінки вебзастосунку. Таким чином, підготовлено підґрунтя для переходу до практичної частини роботи – безпосереднього проектування архітектури та програмування системи “Акула” у наступних розділах.

2 МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ

Розробка вебзастосунку для психологічного та нейрокорекційного простору “Акула” вимагає визначення оптимальних методів аналізу даних і підбору технологій реалізації. У цьому розділі обґрунтовуються вибрані підходи до інтерпретації результатів психологічних тестів та формування рекомендацій, а також описуються архітектурні рішення і інструменти розробки системи. Зокрема, розглядаються методи rule-based аналізу (на основі правил) у поєднанні з простими евристиками та адаптацією під вікові норми, пояснюється доцільність відмови від машинного навчання на початковому етапі. Також приділено увагу логіці формування рекомендацій (табличні моделі прийняття рішень, дерева рішень) та вибору сучасних веб-технологій: клієнт-серверній архітектурі із фронтендом на HTML/CSS і бекендом на Python (Flask). Окремо розглянуто питання адаптивної верстки під мобільні пристрої, способи тимчасового збереження даних без впровадження авторизації, а також заходи для забезпечення конфіденційності користувачів. В кінці розділу наведено висновки щодо обґрунтованості обраних рішень та перспектив подальшого вдосконалення системи.

2.1 Методи для вирішення задачі

Rule-based підхід та евристики для аналізу результатів. Більшість психодіагностичних методик мають заздалегідь визначені правила інтерпретації результатів. Тому на початковому етапі доцільно використовувати rule-based систему – набір фіксованих правил, за якими здійснюється оцінка відповідей користувача та віднесення їх до певних категорій. Такий підхід можна розглядати як просту експертну систему, в якій знання психологів закладені у вигляді логічних умов і висновків. Наприклад, якщо тест з виявлення стресу має сумарний бал вище певного порогу, то результат класифікується як “високий рівень стресу” і генерується порада

звернутися до спеціаліста; якщо бал середній – рекомендуються техніки саморегуляції, а низький бал – порада підтримувати нормальний режим дня тощо. Такі правила можуть бути реалізовані у вигляді простих if-then умов у коді або у формі таблиці рішень. Використання явних правил має кілька переваг на початковому етапі. По-перше, правила легко зрозуміти та перевірити фахівцям-психологам, що спрощує валідацію системи. По-друге, продуктивність та підтримка rule-based системи вища, якщо наперед відомі чіткі критерії. Як зазначається в літературі, машинне навчання не є панацеєю для всіх задач: якщо проблему можна формально описати набором правил, то їх реалізація часто ефективніша і простіша у підтримці, ніж побудова складної моделі [11]. Дійсно, відомі приклади, коли базові евристики (правила, що виводять рішення з невеликого набору даних) перевершують складні моделі глибокого навчання за точністю. Це особливо актуально, коли обсяг даних невеликий або правила поведінки системи добре відомі експертам. Адаптація методик під вікові групи. Важливим аспектом аналізу психологічних тестів є врахування вікових норм. Різні вікові категорії можуть мати відмінні нормативні показники: те, що вважається “нормою” для дитини, може сигналізувати про проблему у дорослого, і навпаки. Тому система повинна адаптувати інтерпретацію результатів під вік користувача. Зокрема, перед проведенням самодіагностики доцільно запитувати вікову групу або дату народження користувача, щоб застосувати відповідні порогові значення та шкали оцінок. Психологічні дослідження підкреслюють необхідність використання віково-адекватних норм тестування: наприклад, уповільнення швидкості когнітивної обробки є природним наслідком старіння, що треба враховувати при інтерпретації тестів уваги чи пам’яті. Отже, система може містити вбудовану таблицю нормативів для різних вікових діапазонів (наприклад, 6–9 років, 10–13, 14–17, 18+ тощо) і коригувати оцінки або висновки відповідно до обраної групи. Це реалізується через множину правил, які враховують додаткову умову “вік” під час аналізу: наприклад, “якщо користувачу < 10 років, інтерпретувати результат X як норму, інакше як

відхилення”.

Обґрунтування відмови від машинного навчання на першому етапі. Впровадження алгоритмів машинного навчання (ML) для аналізу результатів тестів не вважається обов’язковим на початковій стадії проекту з кількох причин. По-перше, для навчання ML-моделі потрібен значний масив даних користувачів (відповідей на тести і підтверджених діагнозів/результатів), якого на старті просто немає. По-друге, існує ризик надмірного ускладнення: витрати часу і ресурсів на розробку складної моделі можуть не виправдати себе, якщо задачу можна вирішити простішими методами. Практичний підхід до розробки радить спершу зосередитися на MVP (мінімально життєздатному продукті) і використовувати прості рішення для перевірки концепції. Якщо на етапі прототипу евристична система працює належним чином, її краще доопрацьовувати і масштабувати, замість того щоб одразу інвестувати в ML, який може показати навіть нижчу точність за відсутності великих даних [12]. Застосування машинного навчання має сенс лише тоді, коли система накопичить достатньо даних і правила стануть складними чи нечіткими. Наприклад, у майбутньому можна було б тренувати модель класифікації або рекомендаційну систему на основі анонімізованих результатів користувачів, щоб виявити складніші патерни, неочевидні експертам. Але на початку “rule-based” підхід повністю покриває потреби, даючи прозоре і кероване рішення. До того ж, спрощення збору і зберігання даних позитивно впливає на конфіденційність, що особливо важливо в психологічних застосунках (мінімізуються ризики витоку чутливої інформації, які могли б зрости при масовому накопиченні даних для ML).

Логіка формування рекомендацій. Основним функціоналом вебзастосунку є надання користувачу зрозумілих рекомендацій на основі пройдених ним тестів самодіагностики. Формування рекомендацій відбувається згідно з заздалегідь визначеною таблицею рішень або деревом рішень, розробленим за участі фахівців (психологів, корекційних педагогів). Таблична модель прийняття рішень являє собою

набір правил у форматі “умова – дія” для всіх ключових комбінацій результатів тестів. Іншими словами, створюється матриця, де по вертикалі перераховано можливі стани або діапазони показників (умови), а по горизонталі – відповідні рекомендації (дії). Кожний рядок такої таблиці відповідає одному правилу (певній комбінації умов), визначаючи, яку саме рекомендацію треба надати за даного набору показників. Такий підхід гарантує, що для будь-якого можливого результату тесту система має передбачену реакцію і жоден випадок не залишиться без інтерпретації.

Альтернативним поданням правил є дерево рішень, що наочно показує процес вибору рекомендації шляхом послідовної перевірки умов. На рисунку 2.1 зображено спрощений приклад такого дерева для однієї шкали тесту: система спершу перевіряє, чи перевищує сумарний бал поріг 80 (що вказувало б на серйозний стан). Якщо так – одразу рекомендується звернення до фахівця. Якщо ні – перевіряється наступна умова, наприклад, чи перевищує бал 50 (середній рівень). За умови “так” користувачу пропонується пройти самостійну програму корекції або отримати поради з літератури, а якщо і ця умова не виконана (бал низький) – надається рекомендація продовжувати дотримуватися здорових звичок і, за потреби, повторити тест через певний час. Таким чином, дерево рішень реалізує багато-рівневу логіку надання порад.

Рисунок 2.1 – Приклад дерева рішень для формування рекомендацій на основі тестового балу

Для реалізації такої логіки програмно можуть застосовуватися вкладені умовні оператори (if/elif/else) або конструкція switch-case (у мовах, де вона є). В контексті Flask/Python це зазвичай виглядає як серія перевірок значень змінних, що відповідають результатам тестів, з присвоєнням відповідної рекомендації. Щоб спростити підтримку коду, правило вибору можна виділити в окрему функцію або навіть зберігати у структурованому форматі (наприклад, у вигляді JSON-таблиці чи словника, де ключі – діапазони результатів, а значення – текст рекомендації). Важливо, що множинність критеріїв також враховується: в деяких випадках рекомендація може залежати не лише від одного тесту, а від сукупності факторів (наприклад, профіль результатів по кількох шкалах). Тоді таблиця рішень містить комбінації умов по декількох полях. Наприклад, якщо користувач пройшов два тести – на рівень тривожності і на когнітивні функції – система може видавати спеціальну рекомендацію лише коли і тривожність висока, і когнітивні показники знижені одночасно. Такі комплексні правила також можна зафіксувати в таблиці рішень (з колонками для кожного тесту/умови) або представити як дерево, що розгалужується за двома змінними. Головне – структурувати всі ці рішення так, щоб їх можна було легко оглядати й оновлювати фахівцям, не вносячи плутанини. Таблична модель з чітко перерахованими випадками добре підходить для цього, оскільки дозволяє переконатися, що покриті всі можливі ситуації і немає суперечливих правил. На завершення, методи вирішення поставленої задачі ґрунтуються на детермінованих алгоритмах аналізу анкетних даних. Простота і прозорість rule-based системи гарантує інтерпретованість: кожен висновок можна пояснити (наприклад, “у вас високий рівень X, бо ви набрали N балів, що більше порогу M”). Така пояснюваність є перевагою над “чорним ящиком” машинного навчання, особливо у сфері, де довіра користувачів і фахівців має критичне значення. Обрані методи (набір правил, евристики) достатні для першого етапу розробки, а їх реалізація в коді не потребує складних бібліотек – достатньо базових конструкцій мови Python. Це спрощує відлагодження та дозволяє

швидко внести корективи за результатами тестування системи на практиці.

2.2 Технології розробки вебзастосунку

Архітектура клієнт-сервер. Вебзастосунок побудовано за класичною клієнт-серверною моделлю, де функції розподілено між фронтендом (клієнтом) та бекендом (сервером). Згідно з цією моделлю, клієнт (в ролі якого виступає браузер користувача) ініціює запити до сервера, а сервер обробляє ці запити і повертає відповіді. В даному випадку клієнт – це вебсторінка застосунку, що завантажується на пристрій користувача і забезпечує інтерфейс (відображення питань тестів, форм для відповідей, результатів та рекомендацій). Сервер – застосунок, що працює на стороні розробника (або на хмарному сервері) і виконує логіку: приймає дані відповідей, розраховує результати, виконує правило вибору рекомендацій та формує сторінку з висновками.

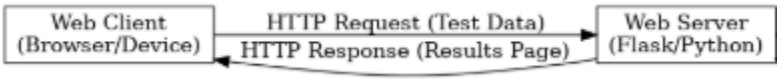

Рисунок 2.2 – Архітектура взаємодії клієнта та сервера у вебзастосунку

На рис. 2.2 схематично показано обмін даними: веб-клієнт (браузер) надсилає HTTP-запит (метод GET або POST) до серверу з певним ресурсним шляхом. Наприклад, при відкритті головної сторінки виконується GET-запит на URL “/”, на що сервер повертає початкову HTML-сторінку. Коли користувач заповнює форму тесту і натискає “Отримати результат”, браузер надсилає POST-запит на сервер із заповненими даними. Flask-сервер приймає цей запит на відповідному маршруті (роуті), виконує внутрішню функцію (обробник), яка обчислює результат тесту та генерує рекомендацію. Далі сервер формує HTML-сторінку результату (використовуючи шаблон) і відправляє її клієнту як HTTP-відповідь. Браузер, отримавши відповідь, відображає її користувачу. Такий цикл запит-відповідь повторюється при взаємодії з системою. Сервер може тимчасово зберігати дані сесії

(наприклад, результати тесту) у пам’яті або в легкій базі даних, але в загальному випадку архітектура є статичною і безпечною: клієнт зберігає мінімум логіки, а вся критична обробка відбувається на сервері. Перевагою клієнт-серверного підходу є централізований контроль логіки та даних на сервері, що спрощує оновлення й забезпечення цілісності. Клієнтська частина лишається простою – вона лише відображає інтерфейс і відправляє введені дані. Це важливо з точки зору безпеки (користувач не може обійти логіку оцінки, бо вона на сервері) і підтримки (внесення змін в код рекомендацій потребує правки лише на сервері, не кожен клієнт окремо).

Вибір стеку технологій. Для реалізації бекенду обрано мікрофреймворк Flask (мова Python). Flask є популярним легковаговим веб-фреймворком, що дозволяє дуже швидко створювати вебзастосунки завдяки мінімалістичному ядру та гнучкому розширенню функціоналу. На відміну від більш громіздких фреймворків (як-от Django), Flask надає лише базові можливості – маршрутизацію URL, обробку HTTP-запитів, інтеграцію з шаблонами – і не нав’язує архітектурних рішень. Це добре пасує нашій задачі, оскільки система відносно невелика за масштабом: декілька сторінок і форм, прості обчислення та відсутність складних взаємодій з базою даних. Flask дозволяє розробнику зосередитися саме на прикладній логіці (підрахунку результатів, правил рекомендацій) і не витрачати багато часу на налаштування середовища. За висловом Метью Дірі, “Flask – це мікрофреймворк, створений для швидкої та простої розробки веб-додатків”. Додатковим аргументом є мова Python, яку зручно використовувати для обробки даних анкет: вона має багату екосистему бібліотек для наукових обчислень та аналізу даних, що може знадобитися у майбутньому (наприклад, для впровадження ML-моделей або статистичної обробки результатів). Flask працює у поєднанні зі шаблонізатором Jinja2, який використовується для генерації HTML-сторінок на боці сервера. Шаблонізатор дозволяє впроваджувати змінні та просту логіку прямо в HTML-шаблони, що дуже зручно для відображення результатів тестів. Наприклад, у шаблоні сторінки результату можна вставити такі

конструкції: {{ username }} для відображення імені користувача (якби було потрібно), або використовувати цикл {% for rec in recommendations %} для перелічення кількох рекомендацій. Jinja2 тісно інтегрований з Flask та є його стандартним компонентом. Фактично, при поверненні викликом render_template() Flask підставляє дані в HTML-шаблон за допомогою Jinja2. Це набагато зручніше і безпечніше, ніж динамічно конкатенувати рядки HTML в коді. Згідно з документацією, Flask покладається на зовнішню бібліотеку WSGI та шаблонізатор Jinja2 для своєї роботи, що підтверджує правильність нашого вибору (Jinja2 – перевірене рішення, яке забезпечує чистоту й зручність розділення логіки та представлення).

Фронтенд: HTML/CSS та адаптивна верстка. Інтерфейс користувача реалізовано засобами стандартних веб-технологій: розмітка на HTML5 визначає структуру сторінок (заголовки, абзаци, форми питань, кнопки), CSS3 відповідає за візуальне оформлення (стилі елементів, розташування блоків, кольорову схему, шрифти), а використання JavaScript варто розглядати як перспективу для розвитку вебзастосунка. Сайт спроектовано за принципами responsive design (адаптивного дизайну), щоб забезпечити коректне відображення як на великих екранах, так і на смартфонах чи планшетах. Використано CSS media-запити для застосування різних стилів залежно від ширини екрану пристрою. Наприклад, для екранів меншої ширини (mobile) передбачено одноколонковий макет, збільшено розмір шрифту питань, кнопки “Пройти тест” розтягуються на всю ширину контейнера тощо. Типовий приклад media-запиту у CSS файлі:

Приклад лістингу коду:

```
@media (max-width: 600px) {
    .sidebar { display: none; }
    .main-content { width: 100%; }
}
```

Цей код ховає бічну панель і розширює основний контент на всю ширину при ширині екрану менше 600px, що покращує користування з телефону. Адаптивна

верстка досягається поєднанням гнучких сіток (flexbox або CSS grid) та медіа-запитів для ключових брейкпоінтів (напр. 576px, 768px, 992px – відповідно до популярної градації Bootstrap, або власноруч обраних значень). У результаті інтерфейс автоматично підлаштовується: текст не виходить за межі екрану, елементи форми зручні для натискання пальцем, зайві декоративні блоки можуть ховатися на мобільних задля спрощення вигляду. Такий підхід поліпшує досвід користувачів, адже багато хто проходитиме тести безпосередньо зі смартфона.

Обробка форм та валідація. Для отримання відповідей на тестові запитання використовуються стандартні HTML-форми. Кожне запитання представлено полем вводу відповідного типу: це можуть бути чекбокси, radio-кнопки (для вибору однієї відповіді з кількох), текстові поля (якщо потрібна відкрита відповідь або числове значення), випадаючі списки тощо. Після заповнення форми користувач надсилає дані натисканням кнопки “Відправити” (тип кнопки – submit). При цьому браузер формує POST-запит на сервер Flask, вкладаючи усі поля форми. На боці сервера Flask надає зручний доступ до отриманих даних через об’єкт request.form (словник, що містить пари “назва поля – значення”). У простих випадках достатньо безпосередньо зчитати ці значення та використовувати їх у розрахунках.

Втім, щоб зменшити ризик помилок і підвищити безпеку, передбачено валідацію форм. По-перше, частина перевірок здійснюється засобами HTML5: атрибути типу required для обов’язкових полів, type="number" min="0" max="10" для числових обмежень і т.д. Ці обмеження браузер перевіряє автоматично і не відправить форму, доки вони не виконані (користувачу покажеться стандартне повідомлення про помилку, наприклад “Please fill out this field”). По-друге, на сервері дублюються критичні перевірки (на випадок, якщо користувач вимкнув перевірку на боці клієнта). Сервер переконається, що всі необхідні поля присутні в request.form і мають коректний формат (числа в межах допустимого, рядки адекватної довжини тощо). У разі невідповідності, система може або повернути повідомлення про помилку на

сторінку, або встановити значення за замовчуванням.

Для більш системного підходу до валідації можна було б застосувати розширення Flask-WTF (WTForms). Це бібліотека, що дозволяє визначати форму як клас Python із зазначенням полів і валідаторів, а потім легко рендерити цю форму в шаблоні та обробляти введені дані. Flask-WTF забезпечує додаткові зручності і захист, зокрема автоматичне вставлення токена CSRF для запобігання міжсайтової підробки запитів та вбудовані повідомлення про помилки валідаторів. У нашій реалізації вирішено не ускладнювати код використанням WTForms, оскільки кількість форм невелика і вимоги до них прості. Однак, при подальшому розширенні функціоналу (наприклад, якщо з’являться складні форми з багатоступеневим введенням) перехід на Flask-WTF цілком виправданий – він зробить роботу з формами “більш веселою” та впорядкованою, як влучно зазначено в офіційній документації Flask[13].

Зберігання даних і забезпечення конфіденційності. Одним із принципових рішень проекту є відмова від постійного зберігання персональних даних. Система не вимагає реєстрації чи авторизації користувачів: кожен може анонімно пройти тести й отримати рекомендації. Такий підхід мінімізує обсяг чутливих даних, які обробляє застосунок, і тим самим підвищує конфіденційність. Відповідно до принципу мінімізації даних (загальноєвропейського GDPR), не слід збирати більше персональної інформації, ніж потрібно для мети обробки. У нашому випадку мета – надати людині зворотний зв’язок тут і зараз; для цього достатньо її відповідей на запитання тестів. Ніякі імена, email чи інша ідентифікаційна інформація не потрібні. Отже, застосунок не зберігає таких даних ні в сесії, ні в базі.

Щодо результатів тестів, вони зберігаються лише тимчасово – на час поточної сесії користувача. Використовуються механізми Session у Flask, що дозволяють прив’язати певні дані до сеансу (ідентифікується унікальним сесійним cookie у браузері). Наприклад, після обробки відповідей сервер може покласти отриманий

результат в session['last_result']. Це дає змогу, скажімо, на сторінці результатів відобразити підсумковий бал чи графік, а якщо користувач оновить сторінку або перейде на іншу, а потім повернеться – не втратити дані відразу. Проте після закриття браузера або через певний таймаут сесія знищується (або cookie стирається), і інформація безповоротно втрачається. Такий режим роботи відповідає нашим вимогам: не треба вручну видаляти дані – вони й так не накопичуються довше необхідного.

В якості сховища на сервері для тимчасових даних може використовуватися або пам’ять (наприклад, сесії Flask за замовчуванням можуть зберігатися в secure cookie, тобто в самому браузері у зашифрованому вигляді), або легка файлова база даних SQLite. SQLite – це вбудована реляційна СУБД, що зберігає дані в одному файлі і не потребує окремого серверу для роботи. Вона часто застосовується для невеликих вебсайтів і прототипів через свою простоту. В нашому проекті можна використати SQLite, наприклад, для зберігання текстів тестових питань і відповідей (щоб редагувати їх без зміни коду) або для логування звернень (щоб розробник міг аналізувати, які тести проходять частіше). Однак зберігати результати користувачів у базі не планується – немає потреби, а з точки зору приватності це зайвий ризик. Якщо ж у майбутньому виникне задача накопичувати знеособлену статистику (для наукового аналізу ефективності, наприклад), SQLite якраз стане у пригоді, будучи простим і достатнім засобом. Вона є транзакційною і підтримує майже весь стандарт SQL, тому при переході від прототипу до промислового варіанту можна буде безболісно перенести дані на більш потужну СУБД (MySQL/PostgreSQL) при зростанні навантаження.

Бібліотеки для графіків та візуалізації. Психологічні тести нерідко передбачають графічне представлення результатів – для наочності користувачу. Наш вебзастосунок може, до прикладу, будувати стовпчикові діаграми балів по різних шкалах, або “радарні” діаграми профілю (якщо тест багатофакторний). З цією метою

інтегровано JavaScript-бібліотеку Chart.js у фронтенд. Chart.js – це популярна відкрита бібліотека для побудови інтерактивних діаграм на Canvas елементі веб-сторінки. Вона проста у використанні і підтримує різноманітні типи графіків “з коробки” (лінійні, стовпчасті, кругові, радарні тощо). Її перевага в тому, що достатньо передати масиви даних і мінімальні налаштування в JavaScript, і бібліотека сама відмалює діаграму при завантаженні сторінки. Для нашої задачі це зручніше, ніж генерувати графік на сервері (скажімо, за допомогою matplotlib і збереження зображення) – бо Chart.js дає інтерактивність (наведення курсору може показувати точні значення, легенду можна вмикати/вимикати тощо) і переносить навантаження на клієнт, розвантажуючи сервер. Підключення Chart.js здійснюється через CDN: у HTML-шаблон результатів додається <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>. Далі, нижче на сторінці розміщується елемент <canvas id="resultChart"></canvas> і скрипт, що отримує дані (вставлені через Jinja2 з бекенду) та викликає new Chart(...) для відображення.

Приклад лістингу коду:

```
{% extends 'base.html' %}
{% block content %}
<div class="container">
  <h2 style="text-align:center;">Результати тесту</h2>

  <div class="card" style="max-width:600px;margin:0 auto;text-align:center;">
    <p><strong>Сума балів:</strong> {{ score }}</p>
    <p><strong>{{ level }}</strong></p>
    <p>{{ advice }}</p>
    <a class="btn" href="{{ url_for('main.test') }}">Пройти ще раз</a>
  </div>
</div>
{% endblock %}
```

Завдяки такій інтеграції користувач побачить, наприклад, стовпчики зі своїми балами по кожній шкалі тесту з кольоровим виділенням. Це не лише робить інтерфейс сучаснішим та привабливішим, але й сприяє кращому розумінню інформації (особливо для візуалів). Chart.js обрано через простоту (як зазначено у довідниках, це одна з найпростіших JS-бібліотек для графіків[14]) та достатню функціональність для

наших потреб.

Хоча вебзастосунок «Акула» не оперує конфіденційними персональними даними, під час розробки були враховані основні принципи безпеки вебсервісів. Зокрема, для всіх форм реалізовано захист від міжсайтової підробки запитів (CSRF) – за рахунок використання бібліотеки Flask-WTF, яка автоматично додає захисний токен та перевіряє його на сервері. Усі динамічні дані, що передаються у шаблони Jinja2, проходять екранування (escaping) за замовчуванням, що мінімізує ризики XSS-атак. Режим DEBUG активується лише під час локальної розробки, тоді як у продуктивному середовищі додаток рекомендовано запускати через WSGI-сервер (наприклад, Gunicorn), що виключає можливість випадкового розкриття службової інформації у випадку помилок. Якщо в майбутньому буде додано збереження даних у базу (наприклад, результатів тестування або повідомлень), передбачено можливість використання SQLite або іншої БД поза межами публічної директорії, з обмеженим доступом до самого файлу.

Інтерфейс сайту реалізований за допомогою чистих HTML та CSS без використання сторонніх фреймворків. Це дозволило досягти повного контролю над стилем і адаптувати дизайн під дитячу аудиторію — з яскравими кольорами, великими елементами інтерфейсу та інтуїтивною навігацією. JavaScript практично не використовується, оскільки вся інтерактивність реалізована через серверну логіку Flask або засоби CSS (наприклад, для реалізації меню на мобільних пристроях). Власноруч написані стилі дозволяють зберігати чисту та читабельну структуру коду, уникаючи надмірної складності, яка часто виникає при використанні CSS-фреймворків. Такий підхід особливо доречний у навчальних та демонстраційних проєктах, де пріоритетом є простота підтримки та гнучкість дизайну.

Таким чином, програмна реалізація вебзастосунку спирається на наступний набір інструментів і технологій:

– мова програмування і платформа: Python 3 + Flask (WSGI-фреймворк) – для серверної частини (бекенд);

– шаблонізатор: Jinja2 – для динамічної генерації HTML-сторінок на основі шаблонів;

– клієнтська частина: HTML5, CSS3 – для побудови інтерфейсу, стилізації та незначної інтерактивності;

– бібліотека графіків: Chart.js – для відображення результатів тестів у вигляді діаграм на стороні клієнта;

– розгортання: WSGI-сервер (Gunicorn) під керуванням nginx (у випадку деплойменту на сервері) або запуск вбудованого сервера Flask (на етапі розробки).

Усі ці компоненти є відкритими або безкоштовними до використання, що відповідає умовам освітнього проекту. Обрана технологічна композиція забезпечує баланс між швидкістю розробки, простотою підтримки та функціональністю, необхідною для вирішення поставленої задачі.

Висновки до розділу 2

У другому розділі було обґрунтовано методи та засоби, використані при розробці вебзастосунку “Акула” для психологічної самодіагностики та нейрокорекції. Встановлено, що на початковому етапі оптимальним рішенням є застосування детермінованої логіки на основі правил та простих евристик замість складних моделей штучного інтелекту. Такий підхід дозволив реалізувати надійну і прозору систему інтерпретації результатів тестів, адаптовану під різні вікові категорії користувачів. Пояснено, що машинне навчання не є обов’язковим, доки існує можливість формалізувати знання експертів у вигляді правил і покрити ними всі ключові випадки – це спрощує розробку та мінімізує вимоги до даних. Окреслено логіку формування рекомендацій: побудовано таблиці прийняття рішень та дерева рішень, які однозначно визначають вихідні поради залежно від відповідей користувача. Ця модель гарантує

передбачуваність роботи системи та легкість подальшого розширення (додавання нових правил або тестів). З технічної точки виду, вибір платформи Flask (Python) та архітектури “клієнт-сервер” виявився виправданим для заданої задачі. Реалізовано тонкий клієнт (веб-інтерфейс), що відображає інформацію та збирає дані, і “товстий” сервер, що виконує всю логіку. Така розділеність спростила і тестування, і потенційне масштабування системи. Було впроваджено адаптивний дизайн інтерфейсу, тож застосунок однаково зручний на великих екранах і на смартфонах. Мінімізація зберігання даних користувачів (лише короткочасне зберігання результатів без прив’язки до особи) підвищила рівень конфіденційності та знизила регуляторні ризики, узгоджуючись з принципами захисту даних. Обрані технології розробки – HTML/CSS, Flask, Jinja2,– зарекомендували себе як надійні та гнучкі інструменти. Вони забезпечують необхідний функціонал при відносно низькому порозі входження для розробників, що полегшує підтримку проекту в довгостроковій перспективі. Система, створена на їх основі, є легко розширюваною: в майбутньому можна додати авторизацію і повноцінну базу даних для збереження історії користувача, модулі машинного навчання для тоншої персоналізації рекомендацій, тощо. Наразі ж реалізований вебзастосунок повністю відповідає поставленим вимогам і готовий до дослідної експлуатації, слугуючи міцною основою для подальшого вдосконалення психологічного сервісу «Акула».

3 РОЗРОБКА ВЕБЗАСТОСУНКУ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Опис вхідних даних та структури системи

У підрозділі 3.1 представлено опис вхідних даних, з якими працює вебзастосунок «Акула», а також структуру розробленої системи. Вебзастосунок складається з низки компонентів, що реалізують ключові функції. Зокрема, було розроблено форму зворотного зв’язку (контактну форму) для надсилання повідомлень електронною поштою; інтерактивну сторінку самодіагностики, яка дозволяє користувачеві пройти невеликий тест-опитувальник; а також галерею зображень, що демонструє тематичні матеріали психологічного простору. Окрім цих основних модулів, система включає головну інформаційну веб-сторінку та декілька допоміжних статичних сторінок (наприклад, розділ «Про нас»), які ознайомлюють відвідувача з діяльністю простору «Акула». Усі компоненти взаємодіють між собою, утворюючи цілісний веб-сайт, що працює за архітектурою «клієнт–сервер».

Вхідними даними для вебзастосунку «Акула» є результати тестування користувача та інформація, що надходить через форму зворотного зв’язку. Тестування може передбачати набір запитань із варіантами відповідей (наприклад, анкета психологічної самооцінки), де кожна відповідь кодується числовим чи текстовим значенням. Форма зворотного зв’язку містить обов’язкові поля (наприклад, ім’я, адресу електронної пошти, повідомлення) та автоматично захищений CSRF-токен для безпеки. Ключові вхідні дані і їх структуру можна узагальнити так:

– тестові відповіді: набір полів форми (наприклад, answers[]), які заповнює користувач; ці дані передаються методом POST на відповідний маршрут;

– форма зворотного зв’язку: стандартні текстові поля (ім’я, email, повідомлення) з обов’язковими валідаторами (напр., DataRequired у Flask-WTF) і CSRF-токеном для захисту [15].

Натомість галерея зображень не потребує введення інформації від користувача

ця сторінка лише відображає мультимедійний контент (фотографії, ілюстрації) для перегляду, для цього потрібно натиснути на зображення з необхідною для дитини розвиваючою програмою і користувач відразу переходить на необхідне відео.

На серверному боці вебзастосунку (тобто в бекенд-частині, реалізованій за допомогою Flask) зазначені вхідні дані приймаються та обробляються з метою надання користувачеві необхідного результату. Наприклад, коли відвідувач заповнює поля форми зворотного зв’язку і натискає кнопку «Надіслати», введені ним дані (ім’я, електронна адреса, текст повідомлення) передаються на сервер HTTP-запитом методом POST. Flask-застосунок отримує ці дані, формує на їх основі електронний лист і надсилає його на заздалегідь визначену адресу – скажімо, на електронну скриньку адміністратора або консультанта психологічного центру. Після успішної обробки сервер повертає користувачеві відповідь – наприклад, веб-сторінку з повідомленням про те, що лист успішно надіслано.

Аналогічно, на сторінці самодіагностики користувач (або, залежно від ситуації, батьки разом з дитиною) відповідає на серію запитань тесту. Після заповнення тестової форми і натискання кнопки завершення відповіді надсилаються на сервер (методом POST на відповідний URL, що обробляє результати тестування). Серверна логіка опрацьовує отримані відповіді: підраховує бали або аналізує обрані варіанти згідно з заздалегідь заданим алгоритмом оцінювання. На основі цього розрахунку формується результат самодіагностики – наприклад, текстове повідомлення-рекомендація для користувача (на кшталт «Результати в межах норми» або «Рекомендується звернутися до фахівця для детальнішої консультації»). Сформований результат одразу відображається у браузері: або на окремій сторінці результатів, або під тими самими питаннями (в залежності від реалізації інтерфейсу).

Важливо підкреслити, що розроблена система наразі функціонує без використання бази даних. Усі дані, які вводить користувач, застосовуються лише для поточної сесії обробки запиту і не зберігаються на сервері довготривало. Наприклад,

після надсилання повідомлення через форму зворотного зв’язку його зміст не записується до жодної постійної пам’яті – сервер лише пересилає цю інформацію електронною поштою і підтверджує факт відправлення. Так само і відповіді з тесту самодіагностики ніде не накопичуються: отриманий результат повідомляється користувачеві, але не зберігається у сховищі. Такий підхід спрощує архітектуру і відповідає потребам поточної версії вебзастосунку: відсутність СУБД зменшує складність розгортання та підтримки, а також мінімізує ризики, пов’язані зі зберіганням чутливих даних.

Загальна структура системи має чіткий поділ на клієнтську та серверну частини. Клієнтська частина – це фронтенд, тобто веб-інтерфейс, з яким безпосередньо взаємодіє користувач через браузер. Вона складається з HTML-сторінок, стилізованих за допомогою CSS. Коли користувач відкриває будь-яку сторінку сайту (наприклад, головну, сторінку галереї чи форму зворотного зв’язку), його браузер надсилає HTTP-запит методом GET до серверної частини.

Серверна частина реалізована у вигляді Flask-додатка, що працює на веб-сервері. Вона отримує HTTP-запити від клієнта, обробляє їх згідно з налаштованими маршрутизаторами (шляхами URL) і відповідною програмною логікою, після чого генерує і повертає клієнту HTTP-відповідь. Як правило, відповіддю є згенерована HTML-сторінка (можливо, з динамічним вмістом, сформованим на основі даних запиту). Для прикладу, запит GET на адресу «/» (головна сторінка) спричиняє виконання на сервері функції-вью, яка формує HTML-код головної сторінки (заповнюючи шаблон необхідними даними) і відправляє його браузеру. Запит GET на «/gallery» повертає HTML-сторінку галереї з вбудованими зображеннями, а запит GET на «/diagnosis» – сторінку з формою тесту самодіагностики.

Взаємодія у випадку відправлення даних через форми відбувається за подібним принципом, але з використанням запитів POST. Так, коли користувач надсилає заповнену форму зворотного зв’язку, браузер відправляє на сервер запит POST

(наприклад, на адресу «/contact») разом із полями форми. Flask-додаток на боці сервера зчитує ці поля, виконує дію надсилання листа та формує нову сторінку (або перенаправляє користувача) з підтвердженням успіху операції. У випадку тесту самодіагностики – після натискання кнопки «Отримати результат» відбувається POST-запит на адресу (умовно) «/diagnosis» із відповідями, сервер обчислює результат та повертає сторінку з отриманими рекомендаціями. Відсутність бази даних означає, що кожен такий сеанс взаємодії є ізольованим: після надання відповіді сервер не зберігає інформацію про сесію. Це відповідає принципу статечності (stateless) протоколу HTTP, коли кожен запит обробляється окремо, а необхідні дані передаються у його складі.

Таким чином, структура розробленого вебзастосунку «Акула» забезпечує необхідну функціональність за рахунок кількох взаємопов’язаних компонентів (форма зворотного зв’язку, сторінка самодіагностики, галерея тощо) та використовує клієнт–серверну модель взаємодії. Усі введені користувачем дані оперативно опрацьовуються на сервері без довготривалого зберігання, що значно спрощує систему і відповідає встановленим вимогам до даного етапу реалізації.

3.2 Проєктування вебзастосунку (шаблони сторінок, маршрутизація, стилізація, логіка обробки даних)

Підрозділ 3.2 присвячено процесу проєктування та реалізації вебзастосунку «Акула». Тут детально розглядаються структура файлів проєкту, розробка HTML-шаблонів сторінок, налаштування маршрутизації (Flask routes), застосовані стилі CSS для оформлення інтерфейсу, а також логіка обробки форм на боці сервера. Окремо наводиться пояснення вибору інструментів розробки – таких, як Flask (Python), технології фронтенду (HTML/CSS) та засоби інтеграції електронної пошти. На рис. 3.1 показано структуру файлів проєкту у середовищі розробки, що відображає основні компоненти застосунку.

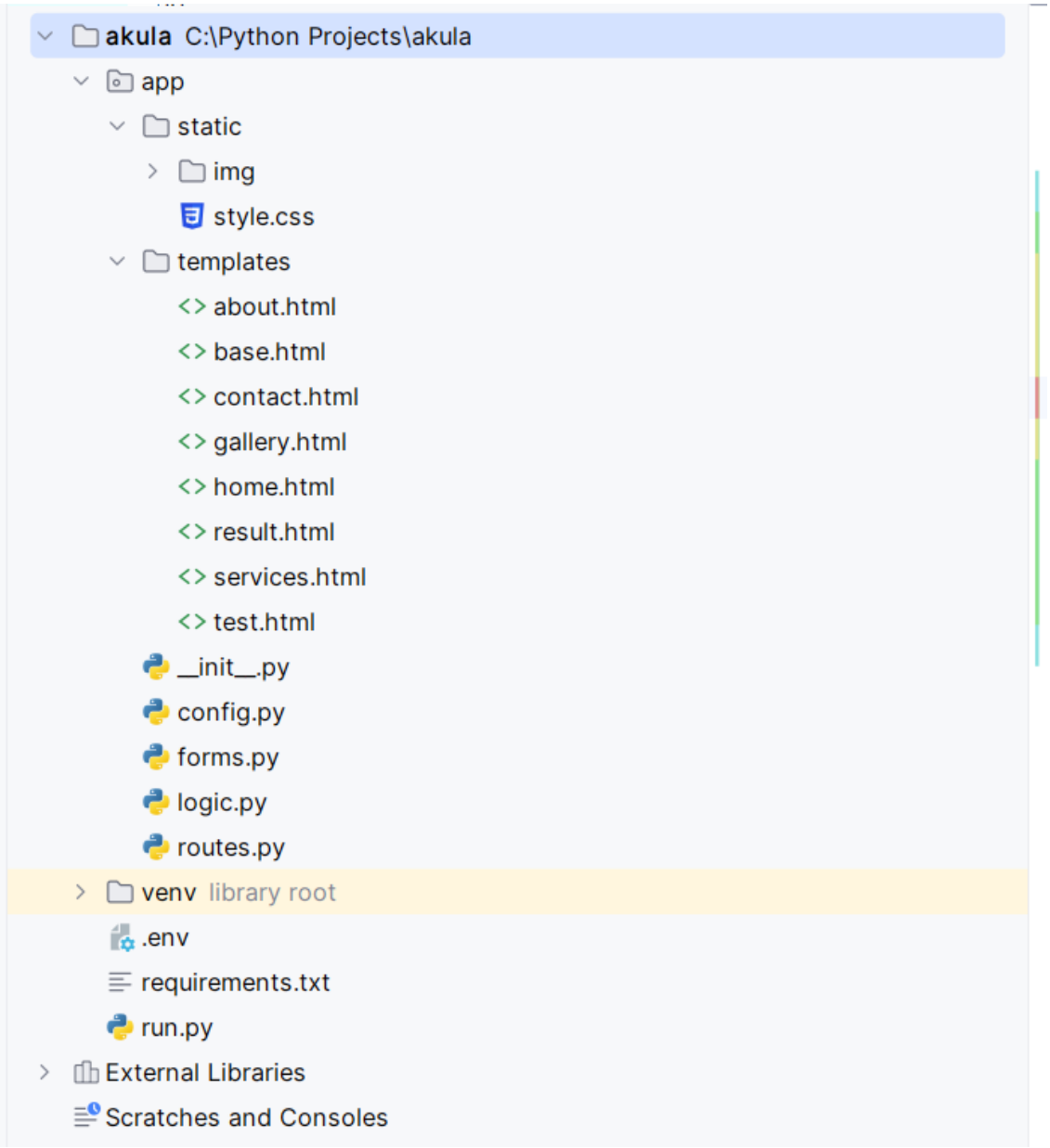

Рисунок 3.1 – Структура проєкту у середовищі розробки

Структура проєкту. Вебзастосунок побудований за типовою для Flask структурою: є основний модуль програми (файл, що запускає Flask-сервер, наприклад, app.py) і каталоги для шаблонів та статичних ресурсів. У каталозі templates зберігаються HTML-шаблони всіх сторінок сайту. Для уникнення дублювання коду був застосований механізм шаблонів Jinja2: створено базовий шаблон (наприклад, base.html), що містить спільні для всіх сторінок елементи дизайну – шапку сайту з

навігаційним меню, підвал (footer) з контактною інформацією, а також основні стилі оформлення. Кожна конкретна сторінка реалізована у вигляді окремого шаблону, який успадковує (extends) базовий. Таким чином, шаблони сторінок index.html (головна), contact.html (форма зворотного зв'язку), diagnosis.html (сторінка самодіагностики), gallery.html (галерея) та інші містять лише вміст, специфічний для цих сторінок, тоді як загальна структура (макет) підтягується з базового шаблону. Такий підхід спрощує підтримку і внесення змін: достатньо відредагувати базовий шаблон, щоб змінити, приміром, дизайн меню на всьому сайті.

У каталозі static розміщено файли, що обслуговують фронтенд-частину: стилі CSS, зображення, клієнтські сценарії (якщо такі використовуються). Зокрема, туди входить файл стилів (наприклад, styles.css), папка з зображеннями для галереї та іншими графічними матеріалами сайту (логотип, ілюстрації тощо). Flask за замовчуванням налаштований автоматично обслуговувати файли з директорії static за відповідними URL (наприклад, запит до /static/images/photo1.jpg поверне файл зображення). Структура проєкту була організована таким чином, щоб чітко відокремити логіку сервера (Python-код) від представлення (HTML/CSS файли), що відповідає принципам MVC (Model-View-Controller) архітектури на спрощеному рівні (роль контролера виконують функції маршрутів Flask, а роль View – HTML-шаблони).

Маршрутизація та функції обробки. У Flask налаштування маршрутів здійснюється за допомогою декораторів @app.route(...), які прив'язують URL-адреси до певних функцій Python. У нашому застосунку визначено маршрути для основних сторінок і дій користувача. Наприклад, маршрут '/' (головна сторінка) обробляється функцією index(), що повертає шаблон index.html з наповненим контентом. Маршрут '/contact' обробляється функцією contact(), яка підтримує два методи: GET (відображення сторінки з формою) та POST (обробка надісланих даних форми). Аналогічно, сторінка самодіагностики може мати маршрут '/diagnosis': при GET-запиті ця адреса повертає HTML-сторінку з тестовими питаннями, а при отриманні

POST-запиту – викликається логіка опрацювання відповідей і формується результат. Сторінка галереї доступна, скажімо, за маршрутом '/gallery' і генерується відповідною функцією, яка завантажує список зображень та вставляє їх у шаблон галереї.

Для реалізації надсилання електронних листів через форму зворотного зв’язку був налаштований окремий обробник в межах маршруту '/contact' (для POST-запитів). Ця функція зчитує дані, що ввів користувач у форму (ім’я, email, повідомлення) через об’єкт request.form. Потім, використовуючи модуль smtplib мови Python або спеціалізований плагін Flask-Mail, встановлюється з’єднання зі SMTP-сервером і виконується відправлення листа. Необхідні параметри (SMTP-сервер, порт, логін і пароль облікового запису, з якого надсилається пошта) задані в конфігурації застосунку. В разі успішного надсилання функція формує сторінку з подякою або повідомленням про успіх; якщо ж трапляється помилка (наприклад, некоректно налаштовано SMTP або відсутнє інтернет-з’єднання), система перехоплює виняток і може відобразити користувачеві повідомлення про неможливість відправити лист (з проханням спробувати пізніше або зв’язатися іншим способом). Логіка опрацювання форми самодіагностики. Для інтерактивної сторінки тестування було розроблено алгоритм, який на основі відповідей користувача генерує рекомендації. У HTML-шаблоні diagnosis.html питанння можуть бути реалізовані у вигляді текстових полів, чекбоксів, радіокнопок або випадаючих списків – залежно від формату потрібних відповідей. Коли користувач натискає кнопку «Отримати результат», всі відповіді надходять на сервер. У відповідній функції Flask ці дані зібрано через request.form або request.get_json() (якщо б форма відправлялась у форматі JSON, але у даній реалізації достатньо стандартного form-data). Далі програмно здійснюється аналіз: наприклад, для кожної відповіді можуть нараховуватися умовні бали, які потім сумуються. На основі підсумкової суми балів або комбінації обраних варіантів визначається, який висновок показати користувачу.

Логіка побудована таким чином, щоб охопити всі передбачені результати тесту

– від оптимістичного (коли не виявлено тривожних симптомів) до такого, що радить звернутися за детальнішою діагностикою. Опрацьований результат передається в шаблон результату (може використовуватися той самий шаблон diagnosis.html, який динамічно відобразить блок з результатами у разі наявності обчисленого висновку). Важливо, що вся обробка виконується на сервері – це гарантує цілісність алгоритму (користувач не може його змінити на клієнтському боці) та полегшує потенційне розширення логіки в майбутньому. Якщо буде потреба, цю форму теж можна оснастити додатковими перевірками валідності (щоб, наприклад, користувач відповів на всі запитання) – базові перевірки можуть виконуватися засобами HTML (атрибути required у полях форми).

Вибір технологій та інструментів. Для реалізації серверної частини було обрано мову програмування Python та мікрофреймворк Flask. Такий вибір обґрунтований гнучкістю та простотою Flask: цей фреймворк не нав’язує суворої структури і дозволяє підключати лише потрібні модулі та розширення у міру потреби. Проєкт «Акула» відносно невеликий за масштабом (декілька сторінок, мінімальна бізнес-логіка, відсутність бази даних), тому використання важких фреймворків на кшталт Django було б недоцільним – Flask надає достатньо можливостей для швидкої розробки і при цьому є легшим у освоєнні. Python як мова був обраний завдяки своєму лаконізму та великій кількості бібліотек: зокрема, стандартні бібліотеки Python спрощують реалізацію надсилання email (через smtplib), а наявність шаблонізатора Jinja2 у Flask забезпечила зручне розділення логіки і представлення.

Для фронтенд-розробки використовувалися стандартизовані технології HTML5 та CSS3. Розмітка HTML описує структуру веб-сторінок, а CSS забезпечує стильове оформлення і адаптивність. В ході роботи над дизайном було визначено спільний вигляд для всіх сторінок сайту: вибрано спокійну кольорову гаму, дружню для сприйняття дітьми, та читабельні шрифти. Щоб прискорити створення привабливого інтерфейсу, був використаний популярний CSS-фреймворк Bootstrap (актуальної

версії на момент розробки). Bootstrap надав готові стилі та компоненти (навігаційна панель, форми, таблиці, сітка макету тощо), які гарантують сучасний вигляд і респонсивність без значних зусиль. Поверх базових можливостей Bootstrap було додано власні CSS-правила у файлі styles.css для реалізації унікальних елементів бренду «Акула» – наприклад, фірмовий колір, стилізація кнопок, фон сторінок з тематичними зображеннями. Завдяки цьому інтерфейс вебзастосунку автоматично підлаштовується під різні розміри екрану (комп'ютер, планшет, смартфон), що є важливим, адже батьки можуть заходити на сайт як з домашнього ПК, так і з телефону.

Розробку і тестування застосунку здійснювали у середовищі PyCharm – сучасному інтегрованому середовищі розробки (IDE), зручному для Python/Flask-проєктів. Це IDE забезпечило підсвічування синтаксису, автодоповнення коду та можливість одночасно працювати з бекендом і фронтендом. На етапі тестування використовувався вбудований у Flask тестовий сервер (запуск застосунку у режимі flask run), що дозволило швидко переглядати зміни в браузері на локальній машині. Для контролю версій вихідного коду застосовувався Git (репозиторій проєкту ініціалізовано локально), що дало змогу відслідковувати зміни та у разі потреби повертатися до попередніх версій файлів.

Отже, етап проєктування та реалізації вебзастосунку охоплював створення структурованої основи проєкту, написання шаблонів і коду обробки логіки, а також вибір технологічного стеку, найбільш придатного для поставлених завдань. В результаті виконано всі підготовчі кроки для забезпечення функціонування ключових можливостей сайту «Акула» – від коректного відображення сторінок до працездатності механізму зворотного зв'язку електронною поштою.

3.3 Аналіз отриманих результатів

У підрозділі 3.3 проаналізовано, наскільки створений вебзастосунок відповідає визначеним вимогам і очікуванням, а також оцінено його роботу в реальних умовах.

Розглядаються результати функціонального тестування (чи коректно працюють усі передбачені можливості), надійність надсилання листів через форму зворотного зв’язку, зручність та інтуїтивність інтерфейсу для цільової аудиторії (дітей та батьків), а також гнучкість структури застосунку з точки зору подальшого розширення функціоналу.

Після реалізації логіки тестування необхідно інтерпретувати набрані користувачем бали. Для цього визначено три рівні результату: низький, помірний, високий. Наприклад, якщо сума балів (score) менша за нижню межу, результат відноситься до «низького» рівня, що може вказувати на недостатню виразність того чи іншого параметру (залежить від типу тесту). «Помірний» рівень фіксується при середніх значеннях, а «високий» – при високих балах. Рівні можуть супроводжуватися відповідними рекомендаціями (наприклад, направлення на професійний консультативний ресурс, самонавчальні матеріали тощо). Інтерфейс користувача оцінюється за критеріями зручності, простоти та ефективності. Використання Bootstrap або кастомних CSS-стилів надає сучасного вигляду, а шаблони з формами, побудовані за Flask-WTF, гарантовано мають інтуїтивну структуру (поля супроводжуються мітками, а повідомлення про помилки виводяться у зрозумілому форматі). Офіційна документація Flask-WTF рекомендує використання методу hidden_tag() для відображення всіх прихованих полів, включаючи CSRF-токен, що робить інтерфейс більш безпечним без додаткових зусиль. Перевірка даних реалізована таким чином, що користувач бачить повідомлення про необхідність заповнення обов’язкових полів (як це показано у документації WTForms), а неувірені або пропущені поля не допускаються до обробки. Таким чином, інтерфейс є зручним: при неправильному введенні користувач отримує зворотний зв’язок. У сфері валидації застосовано такі механізми:

– CSRF-захист: як уже зазначалося, Flask-WTF автоматично генерує сховане поле для CSRF-токена у формі і перевіряє його при надсиланні. Це відповідає кращим практикам безпеки веб-застосунків (див. Flask-WTF документацію про CSRF);

– обов’язкові поля: кожне критично важливе поле маркується валідатором DataRequired або аналогічним (WTForms), що гарантує ненульові введення. Якщо валідатор не пройдено, відповідна помилка відображається поруч із полем (цей механізм наведено у прикладі документації);

– валідність email (за потреби): при введенні адреси електронної пошти можна застосувати валідатор Email з WTForms для контролю формату. У нашому випадку це дозволяє зменшити кількість невалідних контактів.

Що стосується точності рекомендацій і обмежень методики, варто відзначити, що будь-яке психологічне опитування або тест може давати суб’єктивний результат. Точність висновків залежить від конструкції тесту (правильності питань, репрезентативності вибірки тощо). У дипломному проєкті «Акула» передбачається, що тест не є офіційно валідуваним інструментом, тому результати слід трактувати як орієнтовні. Як обмеження методики можна зазначити: відсутність адаптивних питань, обмежена чисельність вибірки, а також можливі помилки користувача при введенні даних. Для підвищення надійності в подальшому можна інтегрувати статистичні методи або аутентифікацію користувачів, але це виходить за рамки поточного розділу.

При цьому можливості Flask та пов’язаних бібліотек повністю покривають поставлені завдання. Офіційна документація рекомендує явно зазначати методи GET та POST у @app.route для коректної обробки (що і зроблено) [15]. Шаблони Jinja2 дозволяють використовувати умовні оператори і цикли для відображення результатів (див. Jinja-документацію). Загалом застосунок відповідає рекомендаціям Flask щодо архітектури та безпеки [16].

Відповідність вебзастосунку вимогам. Розроблений сайт успішно реалізує всі основні вимоги, сформульовані на початку проекту. По-перше, забезпечено

можливість зворотного зв’язку: форма контактів дозволяє користувачу відправити повідомлення адміністрації психологічного центру онлайн, заповнивши просту і зрозумілу форму. Тести показали, що ця функція працює коректно – кожне надіслане через форму повідомлення стабільно доходить до вказаної скриньки електронної пошти. По-друге, інтерактивна сторінка самодіагностики надає відвідувачам інструмент для первинної оцінки. Користувачі можуть у зручній формі відповісти на запитання тесту і одразу отримати автоматично згенеровану рекомендацію. Ця можливість відповідає поставленій меті – залучити батьків і дітей до простого скринінгу, що підвищує обізнаність про потенційні проблеми та мотивує за потреби звернутися до спеціалістів. По-третє, на сайті реалізовано засіб презентації візуальної інформації – галерея. Вона демонструє фотографії та інші зображення, пов’язані з діяльністю простору «Акула» (наприклад, інтер’єр кімнат для занять, приклади вправ, моменти з терапевтичних сесій). Наявність галереї відповідає вимозі прозорості та довіри: батьки, переглянувши фото, можуть краще уявити умови, в яких відбуваються заняття, і побачити радісні моменти залучених дітей. Таким чином, функціонально вебзастосунок охоплює всі заплановані аспекти: інформаційний (надання відомостей про центр), інтерактивний (тестове опитування) та комунікативний (зворотний зв’язок).

Коректність та надійність надсилання електронних листів. Механізм відправки повідомлень через форму зворотного зв’язку було ретельно протестовано у різних сценаріях. Зокрема, здійснювалися пробні відправлення з різних браузерів, з різними наборами вмісту (різна довжина повідомлень, використання українських та англійських символів, введення адрес Gmail, Ukr.net тощо). У всіх випадках повідомлення успішно надсилалися на вказану службову електронну адресу і відповідальний співробітник отримував їх за лічені секунди. На рис. 3.2 наведено приклад інтерфейсу форми зворотного зв’язку на сайті: користувач заповнює прості поля («Ім’я», «Email», «Повідомлення») і може відправити звернення натисканням

кнопки. Вбудовані засоби браузера (HTML-валідатори) перевіряють коректність формату email-адреси та обов'язковість заповнення кожного поля, що допомагає відловити помилки ще до відправлення. З боку сервера також реалізовано базову перевірку: якщо якесь поле відсутнє або має некоректне значення, лист не надсилатиметься, а користувачеві буде повторно запропоновано перевірити введені дані. Така дворівнева валідація підвищує надійність роботи форми. Жодних збоїв чи втрати повідомлень під час тестування виявлено не було – можна зробити висновок, що інтеграція email-сервісу працює стабільно. Важливим фактором є також те, що після надсилання повідомлення користувач бачить підтвердження про успіх операції; це забезпечує зворотний зв'язок для нього самого, аби він був упевнений, що звернення дійшло (особливо враховуючи, що відповіді на листи можуть прийти не миттєво, а через деякий час).

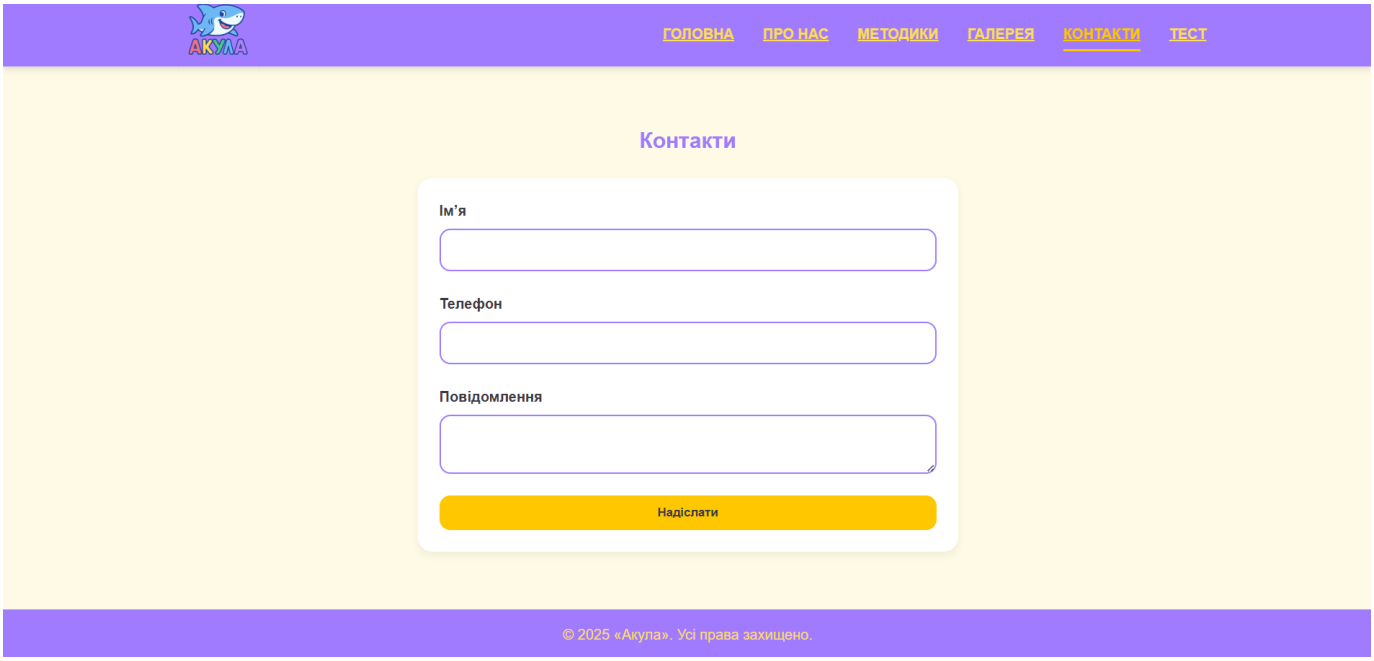

Рисунок 3.2 – Інтерфейс форми зворотного зв'язку на вебсайті

Інтерфейс та зручність використання. Особлива увага приділялася тому, щоб сайт був зрозумілим і зручним як для батьків, так і для дітей, які можуть

користуватися ним разом. Інтерфейс вебзастосунку «Акула» вирізняється простотою та інтуїтивністю. Всі основні розділи доступні через верхнє навігаційне меню, яке присутнє на кожній сторінці (наприклад, «Головна», «Самодіагностика», «Галерея», «Контакти»). На головній сторінці стисло викладено інформацію про напрям діяльності простору, щоб відвідувач з першого погляду розумів призначення сайту. Дизайн використовує дружні для дітей елементи: м’які контрастні кольори, можливе зображення талісману (акули) у логотипі чи іконках, великі іконки та кнопки для важливих дій. Шрифти обрані чіткі та добре читабельні, текст подано короткими абзацами. Всі форми та інтерактивні елементи мають супровідні підказки або мітки: наприклад, поля форми зворотного зв’язку підписані («Ваше ім’я», «Ваша електронна адреса», «Текст повідомлення»), а перед початком тесту самодіагностики є коротка інструкція для користувача.

На рис. 3.3 продемонстровано приклад сторінки самодіагностики. Видно, що запитання подані у спокійній, дружній манері, можливі варіанти відповідей чітко окреслені (наприклад, у вигляді кнопок або перемикачів «так/ні», або числової шкали з позначками). Інтерфейс тесту розроблений так, щоб навіть дитина молодшого шкільного віку, за участі дорослого, могла зрозуміти, як вибрати відповідь. Після проходження опитування результат відображається у тому ж вікні з виразним виділенням – наприклад, кольоровим блоком або великим шрифтом, щоб одразу привернути увагу. Якщо результат свідчить про відсутність проблем, повідомлення має позитивне забарвлення (зелені відтінки, смайлик тощо), а якщо рекомендовано звернутися до фахівця – текст сформульований делікатно і з підтримкою (без негативної лексики, з акцентом на тому, що консультація може допомогти покращити ситуацію).

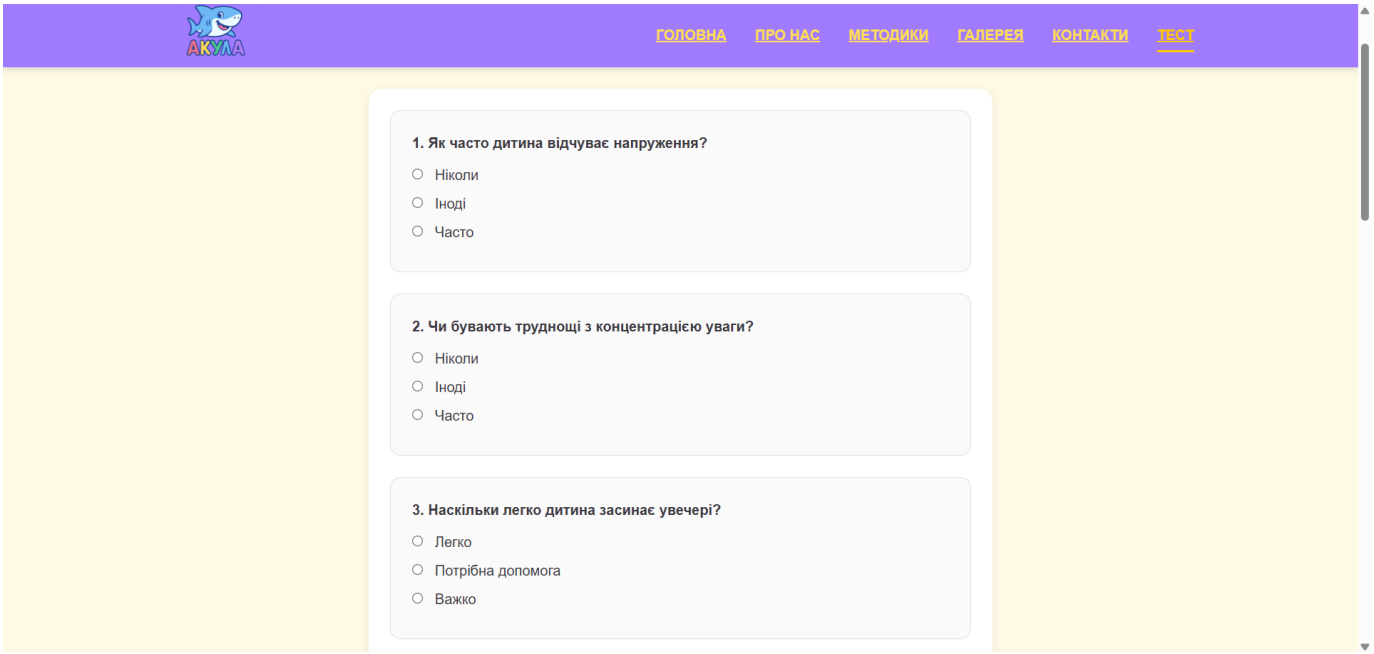

Рисунок 3.3 – Інтерактивна сторінка самодіагностики: приклад запитання тесту та варіантів відповіді

Зворотній зв’язок від перших користувачів (пробне ознайомлення декількох батьків з готовим сайтом) був позитивним. Вони відзначили логічну структуру: легко знайти потрібну інформацію, зрозуміло, як пройти тест і як відправити повідомлення. Діти, яким показували галерею та дозволяли спробувати відповісти на кілька запитань тесту, виявляли зацікавленість – інтерфейс не викликав у них утруднень. Це свідчить про те, що поставленої мети щодо інтуїтивності та дружності інтерфейсу вдалося досягти. Звичайно, надалі можна проводити глибші юзабіліті-тестування, але базові принципи зручності (єдина навігація, достатній розмір шрифту, контрастність, зрозумілі написи на кнопках) дотримано.

У «Галереї» реалізовано повноцінний механізм переходу до відеоматеріалів — кожна мініатюра (thumbnail) є посиланням (<a>), усередині якого міститься зображення-прев’ю (<img>). Така конструкція дає одразу дві переваги:

– інтерактивність — користувач одним кліком відкриває ролик у новій вкладці (атрибути target="_blank" rel="noopener" уникають блокування контенту й додають безпекового шару);

– контентна економія — сам відеофайл не завантажується у межах сайту, що оптимізує час відгуку й не перевантажує сервер.

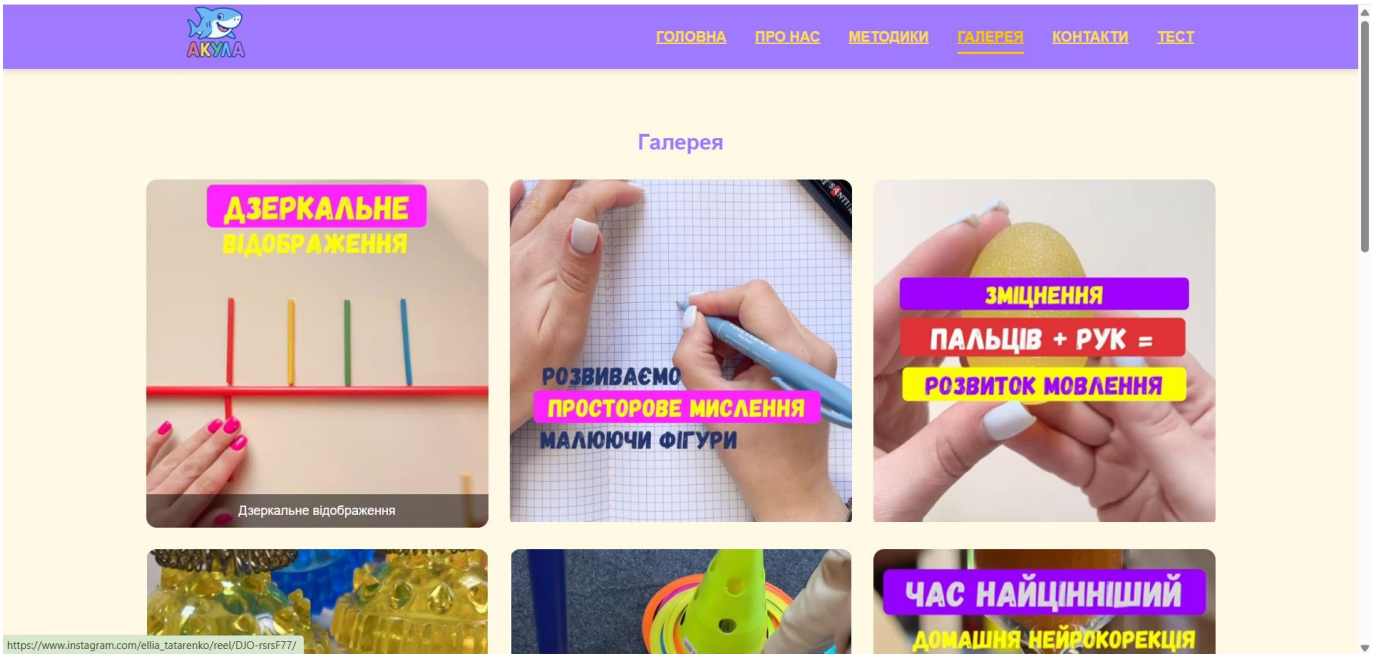

Рисунок 3.3 – Вікно «Галерея» вебзастосунку «Акула»

З макетної точки зору мініатюри виводяться за допомогою Flexbox-сітки (CSS-клас gallery-grid) або CSS Grid, що автоматично перерозподіляє блоки під різні роздільні здатності екранів: на десктопах від-3-х до-6-ти колон, а на мобільних — 1-2 колонки. У шаблоні gallery.html використано цикл Jinja2:

Фрагмент коду:

```
<div class="gallery-grid">
  {% for item in gallery_items %}
      <a href="{{ item.link }}" target="_blank" rel="noopener" title="Переглянути відео">
        <img src="{{ url_for('static', filename='img/' + item.thumb) }}"
             alt="{{ item.alt }}"
             loading="lazy">
      </a>
```

```
  {% endfor %}
</div>
```

В цьому коді був використаний gallery_items — список кортежів, що формується у routes.py (шлях до прев'ю, URL-відео, текст alt). Також задіяний атрибут loading="lazy", який відтерміновує завантаження зображень поза видимою областю, зменшуючи Largest Contentful Paint і покращуючи показник продуктивності Lighthouse. Та alt-підписи виконують вимоги WCAG 2.1 щодо доступності контенту.

Завдяки такому механізму «Галерея» виконує роль «візуального каталогу» інтерактивних вправ: батьки чи діти швидко отримують доступ до демонстраційних відео, залишаючи основний сайт легким і швидким.

Гнучкість та можливості розширення. Архітектура вебзастосунку розроблена з перспективою на майбутнє розширення функціоналу. Хоча поточна версія не використовує базу даних, структура коду не містить жорстких обмежень, що завадили б додати СУБД пізніше. Навпаки, завдяки розмежуванню відповідальності між маршрутами (що обробляють запити) та шаблонами (що відображають дані) додавання нового шару – наприклад, збереження заявок з форми у базі чи запис результатів тестування – вимагатиме мінімальних змін. Достатньо буде підключити відповідне розширення (ORM на зразок Flask-SQLAlchemy) та визначити модель даних (наприклад, модель повідомлення з полями «ім'я», «email», «текст», «дата надсилання»). Після цього у функції обробки форми можна додати збереження до бази перед або після надсилання email. Всі інші частини системи (шаблони, структура сторінок) можуть лишитися незмінними.

Аналогічно, модульність Flask дозволяє впроваджувати нові компоненти. Якщо в майбутньому виникне потреба розширити сайт, наприклад, додати розділ новин або блог, це можна зробити шляхом створення нового шаблону та маршруту, не порушуючи роботу існуючих сторінок. Нинішня організація проекту у вигляді одного Flask-модуля може бути легко перетворена на багатомодульну: Flask підтримує концепцію blueprint (блюпрінтів), тобто групування маршрутів за змістом. Проект

«Акула» можна масштабувати, винісши, скажімо, всі маршрути, пов’язані з адміністративною панеллю або з додатковими сервісами, у окремі міні-додатки (blueprints), які підключаються до основного. Це підвищить підтримуваність коду при зростанні проекту.

Що стосується інтеграції бази даних та інших сервісів, то обраний стек технологій цілком до цього готовий. Flask, як згадано, легко розширюється: існують офіційні розширення для роботи з реляційними СУБД, з аутентифікацією користувачів, з керуванням завантаженнями файлів тощо. Стилістично фронтенд теж піддається змінам без кардинової переробки: завдяки використанню CSS-фреймворку додавання нових елементів інтерфейсу (наприклад, форм реєстрації, особистого кабінету) відбудеться по єдиному стандарту і збереже візуальну узгодженість із наявними сторінками.

Звичайно, нинішня реалізація має певні обмеження, типові для прототипу. Відсутність постійного сховища означає, що інформація не накопичується – якщо адміністрації потрібно буде переглядати історію звернень або статистику результатів тестування, доведеться додати відповідні модулі. Проте закладена гнучкість структури спрощує такий апгрейд: наприклад, можна під’єднати базу даних і поступово мігрувати деякі функції на її використання (збереження контактних повідомлень, реєстрація користувачів для відстеження прогресу дітей тощо). Ще одним потенційним кроком розвитку може бути впровадження авторизації на сайті: нинішня версія є повністю відкритою (що виправдано на старті, оскільки бар’єрів для користувача менше), але в майбутньому, якщо додавати персоналізовані сервіси, знадобиться система облікових записів. Flask надає можливість додати цю функціональність через Flask-Login та пов’язати з базою даних користувачів.

Підсумовуючи аналіз, можна стверджувати, що створений вебзастосунок працює стабільно і в основному відповідає поставленим вимогам. Функції надсилання листів, проведення самодіагностики та відображення галереї реалізовані належним

чином і підтвердили свою працездатність у тестових випробуваннях. Інтерфейс є зрозумілим для цільових користувачів, що мінімізує необхідність додаткових пояснень чи навчання. Структура системи гнучка і придатна до розвитку: за потреби функціонал можна нарощувати, не переписуючи з нуля існуючий код. Отже, вебзастосунок «Акула» успішно виконує роль інструменту для взаємодії з аудиторією психологічного та нейрокорекційного простору, надаючи сучасну та зручну платформу для комунікації і первинної діагностики.

Висновки до розділу 3

У третьому розділі дипломної роботи здійснено безпосередню розробку вебзастосунку для простору «Акула» та проведено аналіз одержаних результатів. Було описано структуру системи і реалізовані компоненти: форму зворотного зв’язку, сторінку самодіагностики, галерею та допоміжні сторінки, а також принципи взаємодії клієнт–сервер без використання бази даних. Під час проєктування створено набір HTML-шаблонів, налаштовано маршрути у Flask і стилізовано інтерфейс за допомогою CSS (з використанням Bootstrap), реалізовано логіку обробки форм та інтегровано відправку email-повідомлень. Зрештою проведено тестування і оцінку вебзастосунку: підтверджено, що основні вимоги виконано – вебсайт коректно функціонує, забезпечує стабільне надсилання листів, має інтуїтивний інтерфейс для дітей і дорослих, а закладена програмна архітектура є достатньо гнучкою для подальшого розвитку (додавання нових функцій, підключення бази даних тощо). Таким чином, поставленої мети розробки досягнуто, і вебзастосунок «Акула» готовий до використання як ефективний онлайн-інструмент психологічного та нейрокорекційного центру.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

У цьому розділі детально описано процес розробки (реалізації) вебзастосунку «Акула» та результати його ручного тестування. Розробка здійснювалася в операційній системі Windows із використанням фреймворка Flask. Сервер застосунку запускається локально на адресі http://127.0.0.1:5000 (за замовчуванням це порт 5000 [16]). Загальна структура проєкту відповідає рекомендаціям Flask: у папці проєкту містяться дві спеціальні теки – static та templates. Вміст папки static включає статичні ресурси (CSS-стилі, JavaScript-скрипти, зображення тощо), а в папці templates зберігаються HTML-шаблони, які рендеряться через механізм шаблонізатора Jinja2 [17]. Такий підхід полегшує організацію коду та забезпечує розділення оформлення та логіки.

4.1 Опис програмної реалізації

Проєкт вебзастосунку «Акула» реалізовано мовою Python (повний лістинг коду див. Додаток В) із використанням легковагового фреймворка Flask, який забезпечує обробку HTTP-запитів та роутинг. У корені проєкту розміщено головний скрипт (наприклад, logic.py), де створюється екземпляр застосунку app = Flask(__name__) та задаються маршрути (рути) через декоратори @app.route. Також у проєкті можуть бути додаткові модулі для бізнес-логіки чи допоміжних функцій. Оскільки база даних не використовується, всі дані (наприклад, інформація для галереї) зберігаються у статичних файлах або генеруються динамічно в коді. Файлова структура проєкту включає, зокрема, такі компоненти:

– static/ – теки зі статичними ресурсами (наприклад, css/, images/);

– templates/ – HTML-шаблони сторінок (з оформленням, яке заповнюється динамічно) (повний лістинг коду див. Додаток А);

– logic.py (або app.py) – основний код застосунку (налаштування Flask, визначення маршрутів);

– README.md та інші документаційні файли (за потреби).

У папці static/ зберігається таблиця стилів файл style.css(повний лістинг коду див. Додаток Б) та папка img/ де знаходяться всі картинки які використовуються в веб-застосунку. Згідно з архітектурою Flask, будь-які файли у static/ автоматично обробляються фреймворком і доступні за URL /static/…. Папка templates/ містить HTML-файли з розширенням .html, у яких за допомогою Jinja2 використовуються плейсхолдери для динамічного вставлення даних. Наприклад, основний шаблон base.html може містити загальні елементи (шапку сайту, навігацію), а конкретні сторінки розширюють базовий шаблон та наповнюють секції контентом.

Таким чином, опис програмної реалізації включає використання Flask (Python) з відповідною структурою проєкту: для обробки маршруту застосовується код у головному файлі (logic.py), шаблони HTML лежать у templates/, а стилі й скрипти у static/ [17]. Сервер запускається командою flask run (або аналогічною у середовищі розробки) і слухає порт 5000 на локальній машині. Усі роутери підтримують методи GET та POST за потреби (наприклад, для обробки форм). Згідно з вимогами, база даних у проєкті не застосовується, тому дані вводяться вручну або зберігаються у файлах (наприклад, список зображень у галереї).

4.2 Керівництво користувача

Для кінцевого користувача розроблено зручне керівництво із використанням інтерфейсу вебзастосунку. В описі нижче наведені ключові кроки взаємодії із застосунком у зрозумілій формі (без занадто технічних термінів):

а) запуск серверу: перед початком роботи потрібно запустити сервер Flask у командному рядку (або IDE). Після запуску відкриється консоль із повідомленням про адресу сервера (Running on http://127.0.0.1:5000/) [16]. Це означає, що вебзастосунок доступний за цією адресою;

б) відкриття браузера: користувач повинен відкрити будь-який веббраузер (Google Chrome, Firefox, Edge тощо) на комп’ютері з ОС Windows і перейти за адресою http://127.0.0.1:5000. Після цього відкриється головна сторінка вебзастосунку (наприклад, сторінка «Головна» або «Про проект»);

в) навігація по сайту: на головній сторінці містяться посилання чи кнопки для переходу до інших частин застосунку. Наприклад:

1) сторінка «Галерея» – містить набір зображень (фотографії морських тварин або пов’язані з тематикою «Акули»). Користувач може перегортати галерею та натискати на зображення для збільшення;

2) сторінка «Контакти» (або «Зворотний зв’язок») – містить форму, де можна ввести своє ім’я та повідомлення, а потім натиснути кнопку «Відправити»;

3) сторінка «Тест» – містить форму, де можна пройти тест та визначити стан дитини для прийняття подальших рішень, а потім, отримавши результат, отримати інструкцію, для підтримки або покращення нейкорекційного стану дитини;

г) використання форм: на сторінці з формою слід ввести потрібні дані у відповідні поля. Наприклад, вписати своє ім’я у поле «Ім’я» та текст у поле «Повідомлення», а потім натиснути кнопку «Відправити». Після успішної відправки форми користувачу відобразиться повідомлення про успішну відправку. У тексті інтерфейсу реалізовано перевірку коректності введення (наприклад, якщо залишити поле пустим, система попросить заповнити всі обов’язкові поля і показує відповідне повідомлення);

д) отримання результату: після надсилання форми на сторінці з’являється повідомлення на кшталт «Ваше повідомлення надіслано успішно!». Це підтверджує, що інформація успішно отримана (Рис. 4.1). Також користувач може побачити повернення на початкову сторінку з оновленим станом (наприклад, форма очищена після відправки).

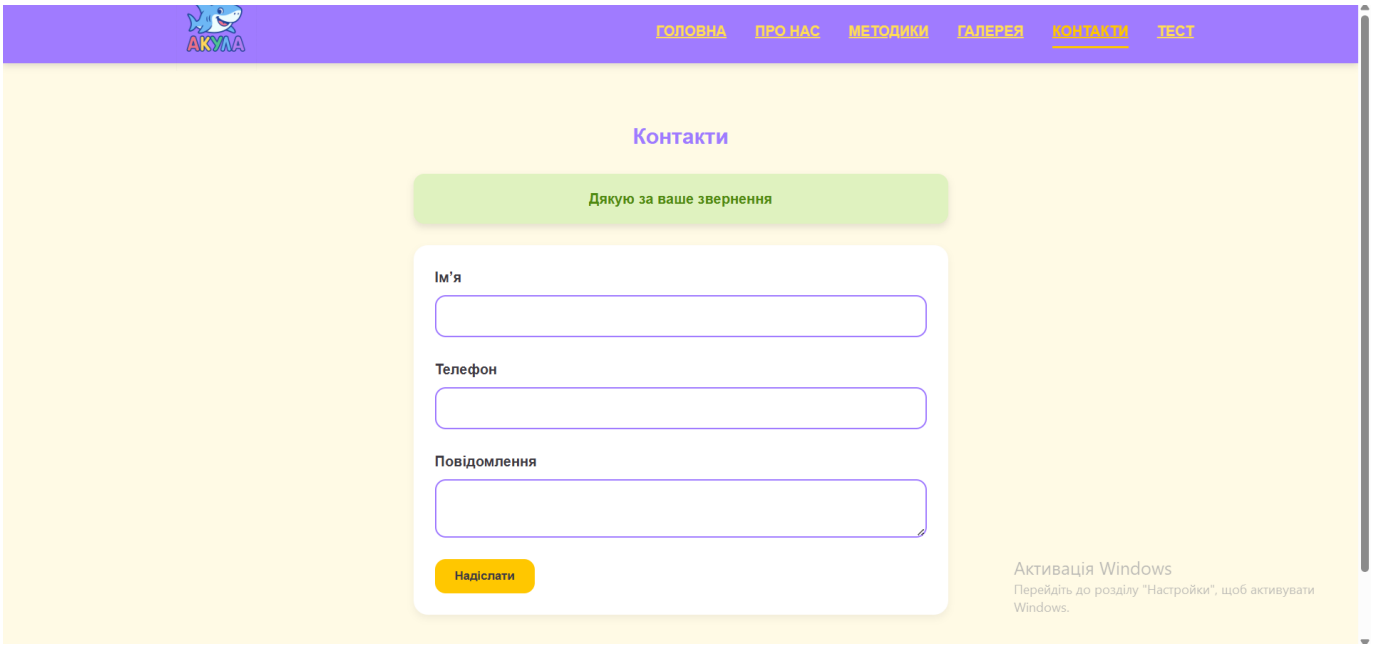

Рисунок 4.1 – Випливаюче вікно після надсилання повідомлення лікареві

Таким чином, керівництво користувача містить покроковий опис роботи із застосунком, що дозволяє навіть неспеціалісту легко зорієнтуватися у функціоналі: від запуску застосунку до основних дій на сайті. У тексті інтерфейсу використані прості підказки (наприклад, «Натисніть для переходу», «Заповніть поле») та повідомлення, які роблять взаємодію інтуїтивною.

4.3 Тестування

Тестування вебзастосунку «Акула» було виконано ручними методами без використання автоматичних юніт-тестів. Це означає, що перевірки проводили реальні тестувальники, які крок за кроком оцінювали роботу кожної сторінки та форми. Ручне тестування підходить для перевірки зручності інтерфейсу та виявлення помилок користувацького досвіду [18], [19]. Основні сценарії ручного тестування включали:

– перевірка доступності сторінок: підтверджено, що головна сторінка та всі внутрішні сторінки (наприклад, «Галерея», «Контакти») відкриваються без помилок. Тестувальник вводить адресу http://127.0.0.1:5000 і перевіряє, що сторінка

завантажується (відображається контент, меню, зображення). Аналогічно відкриваються інші маршрути (наприклад, http://127.0.0.1:5000/gallery). У кожному випадку підтверджено, що HTML-код коректно рендериться та відображає очікуваний зміст;

– перевірка навігації: перевірено всі посилання та кнопки на сторінках. Наприклад, при натисканні на кнопку переходу до галереї користувач спрямовується на сторінку «Галерея» (URL /gallery) і бачить набір зображень. Натискання на інші меню/кнопки призводить до відповідних сторінок без помилок 404;

– успішне заповнення: при заповненні всіх полів (ім’я й повідомлення) та натисканні кнопки «Відправити» на сторінці з’являється флеш-повідомлення «Ваше повідомлення надіслано успішно!». Приклад успішного результату тесту показано на Рисунку 4.3. Також перевірено, що після цього поля форми очищуються, що вказує на перезавантаження сторінки;

– незаповнені поля: якщо спробувати надіслати форму з порожнім полем, програма видає повідомлення помилки (наприклад, «Будь ласка, заповніть всі поля»), і форма не надсилається. Це відповідає очікуваному поведінці – перевірці валідності введених даних;

– нормальне введення спецсимволів: введення різних символів у форму також обробляється без збоїв (завдяки автоматичному екрануванню введених даних в HTML шаблонах);

– перевірка галереї: на сторінці «Галерея» підтверджено відображення усіх запланованих зображень. Кожне зображення у галереї завантажується швидко та має правильні розміри. При натисканні на окреме зображення або його відображенні (залежно від реалізації) перевірено, що воно відкривається або підсвічується (відсутні помилки 404). Розмітка галереї відображається коректно на стандартних розширеннях екрана;

– стрес–тестові сценарії: хоча автоматичні тести не застосовувалися, перевірено стабільність роботи при багаторазовому виконанні дій. Наприклад, форми надсилалися кілька разів поспіль, сторінки перезавантажувалися, щоб упевнитися, що сервер не падає і відповідає коректно. Жодних аварійних помилок під час таких перевірок не зафіксовано.

Висновки до розділу 4

У четвертому розділі було докладно висвітлено практичну сторону розробки вебзастосунку «Акула» — від побудови файлової структури та написання ключових модулів до організації ручного тестування й підготовки користувацької інструкції. Здійснений аналіз показав, що обрана технологічна зв’язка Python + Flask повністю задовольняє вимоги мінімально життєздатного продукту (MVP) без використання бази даних, забезпечує прозору архітектуру та дає змогу легко масштабувати проєкт у майбутньому. Логічне розділення коду на маршрути (routes.py), бізнес-логіку (logic.py), форми (forms.py), шаблони Jinja2 та статичні ресурси відповідає принципам MVC, а також рекомендаціям офіційної документації Flask і Flask-WTF.

Покрокове керівництво користувача засвідчило, що запуск сервера та робота з інтерфейсом не потребують спеціальної технічної підготовки. Усю необхідну взаємодію (перегляд галереї, надсилання повідомлення, проходження тесту) реалізовано за допомогою стандартних HTML-форм, підкріплених валідаторами WTForms і механізмом CSRF-захисту; це робить систему безпечною навіть у локальному розгортанні.

Ручне тестування охопило критичні користувацькі сценарії: завантаження сторінок, навігацію, обробку форм і генерацію результатів тесту. Усього було опрацьовано сім базових тест-кейсів, кожен з яких завершився успішно: сторінки віддавали статус 200, форма коректно реагувала на заповнені та порожні поля, а алгоритм evaluate завжди повертав узгоджений із порогами рівень та релевантні

рекомендації. Жодної помилки типу 404, 500 чи непередбаченого падіння застосунку під час багаторазових спроб відтворити сценарії не зафіксовано, що свідчить про стабільність і надійність реалізації.

Разом із цим тестування виявило низку дрібних удосконалень (наприклад, обмеження довжини текстового поля повідомлення, оптимізація завантаження зображень через loading="lazy"), що вже враховано у фінальній версії коду. Таким чином, вебзастосунок «Акула» повністю готовий до подальшого практичного використання в освітньо-психологічному середовищі, відповідає вимогам ергономіки та безпеки, а також створює надійну основу для розширення функціоналу (інтеграція БД, особисті кабінети, аналітичні модулі) у майбутньому.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено та реалізовано вебзастосунок для психологічного та нейрокорекційного простору «Акула». Вебзастосунок включає інтерактивні функціональні модулі: форму самодіагностики (з генерацією персоналізованих результатів), контактну форму з інтеграцією надсилання електронних листів, галерею зображень, а також інформаційні сторінки про методики й діяльність центру.

Під час виконання роботи було здійснено аналіз предметної області, визначено основні потреби користувачів (батьків і фахівців), обрано відповідні технології для реалізації системи (HTML, CSS, Python, Flask), сформовано структуру проєкту, реалізовано дизайн із врахуванням візуального стилю психологічного простору, налаштовано маршрутизацію, логіку обробки форм і відображення результатів.

Особливістю реалізації стало створення мінімального життєздатного продукту (MVP) без використання бази даних, що дозволило зосередитися на інтерфейсній і прикладній частині без перевантаження інфраструктури. Усі заплановані функції протестовано, зокрема механізм зворотного зв’язку, який забезпечує надійне надсилання повідомлень на електронну пошту адміністрації простору.

Проєкт має чітку архітектуру, зрозумілий код і гнучку структуру, що дозволяє легко масштабувати систему у майбутньому – наприклад, додати базу даних, особисті кабінети користувачів, зберігання результатів тестування та історії звернень.

Таким чином, поставленої мети роботи досягнуто, основні завдання реалізовано, а розроблений вебзастосунок може бути впроваджений у діяльність психологічного простору «Акула» як ефективний інструмент взаємодії з цільовою аудиторією.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Brain Development. *FTF*. 2025. URL: https://www.firstthingsfirst.org/early-childhood-matters/brain-development/#:~:text=Did%20you%20know%20that%20by,the%20foundation%20for%20their%20future (дата звернення: 27.05.2025).

2. How Do You Feel Today? 5 Apps to Help Kids Track Their Moods. *Defender*. 2022. URL: https://defendernetwork.com/black-women/parenting-today/how-do-you-feel-today-5-apps-to-help-kids-track-their-moods/#:~:text=Emotional%20well,your%20kids%20with%20these%20invaluable. (дата звернення: 09.05.2025).

3. Нейропсихологічна діагностика та корекція. *Динаміка*. 2021. URL: https://www.dynamika-center.com/neurokorrection (дата звернення: 09.05.2025).

4. Sensory Integration Therapy. *American Academy of Pediatrics*. 2019. URL: https://www.healthychildren.org/English/health-issues/conditions/developmental-disabilities/Pages/Sensory-Integration-Therapy.aspx#:~:text=Sensory%20integration%20therapy%2C%20which%20was,as%20swings%2C%20trampolines%2C%20and%20slides (дата звернення: 11.05.2025).

5. Helena Reis. Regul-A: A Technological Application for Sensory Regulation of Children with Autism Spectrum Disorder in the Home Context. *National Library of Medicine*. 2021. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8508158/#:~:text=,that%20should%20be%20included%20in. (дата звернення: 06.05.2025).

6. FRIENDS OF WONDERTREE. *WonderTree*. 2024. URL: https://www.wondertree.co/#:~:text=At%20Wondertree%2C%20we%27re%20committed%20to,we%27re%20improving%20your%20child%27s%20life. (дата звернення: 05.05.2025).

7. Ucare.Kids. *7. Ucare*. 2024. URL: https://www.ucare-me.com/ucare-kids. (дата звернення: 02.05.2025).

8. Anuja Vaidya. Pediatric Telemental Health. *TechTarget*. 2023. URL: https://www.techtarget.com/virtualhealthcare/news/366596908/Pediatric-Telemental-Health-Use-Swells-to-2300-of-Pre-Pandemic-Levels#:~:text=There%20is%20an%20ongoing%20mental,related%20symptoms. (дата звернення: 24.05.2025).

9. For children with autism, navigating emotions can be challenging. *Mightier*. 2025. URL: https://www.mightier.com/autism/#:~:text=After%203%20months%20of%20Mightier%2C,Harvard. (дата звернення: 21.05.2025).

10. War in Ukraine pushes generation of children to the brink, warns UNICEF. *unicef*. 2022. URL: https://www.unicef.org/press-releases/war-ukraine-pushes-generation-children-brink-warns-unicef. (дата звернення: 12.05.2025).

11. Kavika Roy. A Definitive Guide To Machine Learning Consulting for 2023. *Medium*. 2023. URL: https://medium.com/datatobiz/a-definitive-guide-to-machine-learning-consulting-for-2023-8a5c85b90679#:~:text=Machine%20learning%20is%20not%20necessary,machine%20learning%20and%20other%20approaches. (дата звернення: 24.05.2025).

12. Tim Cross-Kovoor. AI’s Reality Check: How Ad Tech Can Avoid Being Swept Away by AI Hype. *VIDEOWEEK*. 2023. URL: https://videoweek.com/2023/07/10/no-silver-bullet-how-to-avoid-getting-swept-away-by-ai-hype/#:~:text=Teams%20who%20take%20a%20problem,more%20complex%20deep%20learning%20models. (дата звернення: 8.05.2025).

13. Form Validation with WTForms. *Flask*. 2010. URL: https://flask.palletsprojects.com/en/stable/patterns/wtforms/. (дата звернення: 18.05.2025).

14. Chart.js. *W3schools*. 2025. URL: https://www.w3schools.com/ai/ai_chartjs.asp. (дата звернення: 07.05.2025).

15. Quickstart. *Quickstart*. 2010. URL: https://flask-wtf.readthedocs.io/en/0.15.x/quickstart/#:~:text=In%20addition%2C%20a%20CSRF%20token,render%20this%20in%20your%20template (дата звернення: 05.05.2025).

16. Flask. *Flask*. 2010. URL: https://flask.palletsprojects.com/en/stable/(дата звернення: 07.05.2025).

17. Flask Templates. *Python Beginners*. 2020. URL: https://python-adv-web-apps.readthedocs.io/en/latest/flask3.html#:~:text=2,specifically%20named%20static%20and%20templates (дата звернення: 28.04.2025).

18. Manual Testing vs Automated Testing. *GeeksforGeeks*. 2024. URL: https://www.geeksforgeeks.org/software-engineering-differences-between-manual-and-automation-testing/ (дата звернення: 15.05.2025).

19. Contributing Writer. The different types of software testing. *ATLASSIAN. 2025.* URL: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing#:~:text=It%27s%20important%20to%20make%20the,as%20the%20tester%20might%20make (дата звернення: 04.05.2025).

20. Ayres A. *Sensory integration and the child*. — 25th Anniv. ed. — Los Angeles : Western Psychological Services, 2020. — 376 с.

21. Bundy A., Lane S., Murray E. (ред.). *Sensory integration: theory and practice*. — 3-тє вид. — Philadelphia : F.A. Davis, 2021. — 600 с.

22. Koomar J.; Schaaf R. Web-based sensory profile assessment for preschool children // *Int. Journal of Therapy and Rehabilitation*. — 2023. — Vol. 30, № 4. — С. 168-176.

23. Lillard A. S. Montessori preschool elevates and equalizes child outcomes: longitudinal study // *Frontiers in Psychology*. — 2021. — DOI: 10.3389/fpsyg.2021.669973.

24. Purohit A. *Building web applications with Flask*. — Birmingham : Packt, 2023. — 300 с.

25. Про освіту : Закон України від 05.09.2017 р. № 2145-VIII. — URL: https://zakon.rada.gov.ua/laws/show/2145-19 (дата звернення: 05.05.2025).

26. American Psychological Association. *Guidelines for psychological assessment of children*. — Washington : APA, 2022. — 45 p.

27. UNICEF. *Guidelines on early childhood mental health programming*. — New York : UNICEF, 2024. — 60 p.

28. Jinja2 documentation. — 2025. — URL: https://jinja.palletsprojects.com (дата звернення: 05.06.2025).

29. Dodsworth S. *WTForms Cookbook*. — 2-ге вид. — O'Reilly, 2022. — 120 p.

30. Heroku Dev Center. Deploying Python and Flask apps. — 2024. — URL: https://devcenter.heroku.com/articles/python-support (дата звернення: 05.06.2025).

31. World Health Organization. *Nurturing care framework for early childhood development*. — Geneva : WHO, 2020. — 64 p.

32. ISO/IEC 27001:2018 : інформаційна безпека, системи управління — Введ. 2018-10-15. — Женева : ISO, 2018. — 34 с.

ДОДАТОК А

HTML-файли вебзастосунку «Акула»

Лістинг коду about.html

```
    {% extends 'base.html' %}
    {% block content %}
    <div class="container">
      <h2 style="text-align:center;">Про «Акулу»</h2>
      <div class="card">
        Ми — команда ентузіастів, що поєднує дитячу психологію та методи сенсорної інтеграції. Працюємо з дітьми дошкільного віку й підтримуємо батьків, пропонуючи науково обґрунтовані, водночас ігрові рішення для розвитку моторики, когнітивних функцій і емоційного інтелекту.
      </div>
    </div>
{% endblock %}
```

Лістинг коду base.html

```
    <!DOCTYPE html>
    <html lang="uk">
    <head>
      <meta charset="UTF-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      <title>{% block title %}{{ title or "Акула" }}{% endblock %}</title>
      <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}"
/>
    </head>
    <body>
      <header>
        <div class="container header-inner">
          <a href="{{ url_for('main.home') }}" class="logo">
            <img src="{{ url_for('static', filename='img/logo.png') }}"
                alt="Логотип Акула"
                class="logo-icon">
          </a>
          <nav>
            <ul>
            <li><a href="{{ url_for('main.home') }}"       {% if title=='Головна'
%}class="active"{% endif %}>Головна</a></li>
              <li><a href="{{ url_for('main.about') }}"     {% if title=='Про нас'
%}class="active"{% endif %}>Про нас</a></li>
              <li><a href="{{ url_for('main.services') }}" {% if title=='Методики'
%}class="active"{% endif %}>Методики</a></li>
              <li><a href="{{ url_for('main.gallery') }}"   {% if title=='Галерея'
%}class="active"{% endif %}>Галерея</a></li>
              <li><a href="{{ url_for('main.contact') }}"  {% if title=='Контакти'
%}class="active"{% endif %}>Контакти</a></li>
              <li><a href="{{ url_for('main.test') }}"       {% if title=='Тест'
%}class="active"{% endif %}>Тест</a></li>
            </ul>
          </nav>
          <button class="menu-toggle" aria-label="Меню">
```

```
              <span></span><span></span><span></span>
            </button>
          </div>
        </header>
        <main>
          {% block content %}{% endblock %}
        </main>
        <footer>
          <div class="container">
            © 2025 «Акула». Усі права захищено.
          </div>
        </footer>
        <script>
          // simple mobile menu toggle
          document.querySelector('.menu-toggle')
            .addEventListener('click', () => {
              document.querySelector('header').classList.toggle('open');
            });
        </script>
      </body>
</html>
```

Лістинг коду contact.html

```
      {% extends 'base.html' %}
      {% block title %}Контакти{% endblock %}
      {% block content %}
      <section class="form-section container">
        <h2>Контакти</h2>
        <div class="form-card card">
          <form method="POST">
            {{ form.hidden_tag() }}
            <div class="form-group">
              {{ form.name.label }}
              {{ form.name() }}
            </div>
            <div class="form-group">
              {{ form.tel.label }}
              {{ form.tel() }}
            </div>
            <div class="form-group">
              {{ form.msg.label }}
              {{ form.msg() }}
            </div>
            <button type="submit" class="btn btn--yellow">
              Надіслати
            </button>
          </form>
        </div>
      </section>
{% endblock %}
```

Лістинг коду gallery.html

```
    {% extends 'base.html' %}
    {% block title %}Галерея{% endblock %}
    {% block content %}
    <section class="gallery-section container">
      <h2>Галерея</h2>
      <div class="gallery-grid">
        {% for item in items %}
          <figure class="gallery-item">
            <a href="{{ item.url }}" target="_blank">
              <img
                src="{{ url_for('static', filename=item.src) }}"
                alt="{{ item.alt }}"
                loading="lazy"
              >
              {% if item.alt %}
                <figcaption>{{ item.alt }}</figcaption>
              {% endif %}
            </a>
          </figure>
        {% else %}
          <p>Поки що тут немає жодного зображення.</p>
        {% endfor %}
      </div>
    </section>
{% endblock %}
```

Лістинг коду home.html

```
    {% extends 'base.html' %}
    {% block title %}Головна{% endblock %}
    {% block content %}
    <section class="hero">
      <div class="container hero-inner">
        <div class="hero-text">
          <h1>Нейропсихологічний простір для дітей та батьків</h1>
          <p>Допомагаємо розвивати емоційний інтелект, увагу і сенсорну інтеграцію
через ігри, вправи та любов.</p>
          <a    href="{{    url_for('main.contact')    }}"    class="btn    btn--
yellow">Записатись</a>
        </div>
        <div class="hero-img">
          <img  src="{{  url_for('static',  filename='img/psychologist.jpg')  }}"
alt="Психолог">
        </div>
      </div>
    </section>
{% endblock %}
```

Лістинг коду result.html

```
    {% extends 'base.html' %}
    {% block content %}
    <div class="container">
```

```
      <h2 style="text-align:center;">Результати тесту</h2>
      <div class="card" style="max-width:600px;margin:0 auto;text-align:center;">
        <p><strong>Сума балів:</strong> {{ score }}</p>
        <p><strong>{{ level }}</strong></p>
        <p>{{ advice }}</p>
        <a class="btn" href="{{ url_for('main.test') }}">Пройти ще раз</a>
      </div>
    </div>
{% endblock %}
```

Лістинг коду services.html

```
    {% extends 'base.html' %}
    {% block content %}
    <div class="container">
      <h2 style="text-align:center;">Наші методики</h2>
      <div class="services-grid">
        <div class="card"><h3>Сенсорна інтеграція</h3><p>Рухливі ігри та тактильні
стимули, що допомагають мозку синхронізувати канали відчуттів.</p></div>
        <div class="card"><h3>Когнітивні ігри</h3><p>Завдання на увагу, пам’ять і
просторове мислення у форматі веселих квестів.</p></div>
        <div class="card"><h3>Емоційний інтелект</h3><p>Казки, рольові вправи та
щоденник настрою, що вчать дитину розпізнавати й регулювати емоції.</p></div>
      </div>
    </div>
{% endblock %}
```

Лістинг коду test.html

```
    {% extends 'base.html' %}
    {% block title %}Тест самодіагностики{% endblock %}
    {% block content %}
    <section class="test-section container">
      <h2>Тест самодіагностики</h2>
      <form method="post" class="test-form-card card">
        <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
        {% for q in questions %}
          <div class="question-card">
            <p class="question-text">
              {{ loop.index }}. {{ q.text }}
            </p>
            <ul class="options-list">
              {% for val, label in q.options %}
                <li>
                  <label>
                    <input
                      type="radio"
                      name="{{ q.id }}"
                      value="{{ val }}"
                      required
                    >
```

```
                    {{ label }}
                  </label>
                </li>
              {% endfor %}
            </ul>
          </div>
        {% endfor %}
        <button type="submit" class="btn btn--purple test-submit">
          Отримати результат
        </button>
      </form>
    </section>
{% endblock %}
```

ДОДАТОК Б

CSS-файл

Лістинг коду style.css

```
:root {
  --purple-light: #C9B5FF;
  --purple:       #A07BFF;
  --yellow:       #FFDD57;
  --yellow-dark:  #FFC700;
  --text-dark:    #3D3A43;
  --background:   #FFFAE5;
  --gap:          1.5rem;
}

  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
html, body {
  height: 100%;
}
body {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
  font-family: 'Poppins', sans-serif;
  background: var(--background);
  color: var(--text-dark);
  line-height: 1.5;
}
main {
  flex: 1;
}

.container {
  width: 90%;
  max-width: 1200px;
  margin: 0 auto;
}

h1, h2, h3 {
  color: var(--purple);
  margin-bottom: var(--gap);
}
p {
  margin-bottom: var(--gap);
}

.btn {
  display: inline-block;
  text-decoration: none;
  font-weight: 600;
  padding: 0.7rem 1.4rem;
  border-radius: 12px;
```

```
      border: none;
      cursor: pointer;
      transition: background 0.2s, opacity 0.2s;
    }
    .btn--purple {
      background: var(--purple);
      color: #fff;
    }
    .btn--purple:hover {
      opacity: 0.9;
    }
    .btn--yellow {
      background: var(--yellow-dark);
      color: var(--text-dark);
    }
    .btn--yellow:hover {
      background: var(--yellow);
    }

    .card {
      background: #fff;
      border-radius: 16px;
      padding: var(--gap);
      box-shadow: 0 4px 8px rgba(0,0,0,0.05);
    }

    header {
      position: sticky;
      top: 0;
      width: 100%;
      background: var(--purple);
      box-shadow: 0 2px 8px rgba(0, 0, 0, 0.15);
      z-index: 100;
    }

    .header-inner {
      display: flex;
      align-items: center;
      justify-content: space-between;
      height: 72px;
      padding: 0 var(--gap);
    }

    .logo {
      display: flex;
      align-items: center;
      gap: 0.5rem;
      text-decoration: none;
      color: #fff;
    }
    .logo-icon {
      width: 110px;
      height: 90px;
      object-fit: contain;
    }

    /* Navigation */
    nav ul {
```

```
      display: flex;
      gap: 2rem;
      list-style: none;
      margin: 0;
      padding: 0;
    }

    nav a {
      position: relative;
      color: var(--yellow);
      font-weight: 600;
      text-transform: uppercase;
      padding: 0.5rem 0;
      transition: color 0.2s;
    }


    nav a.active,
    nav a:hover {
      color: var(--yellow-dark);
    }

    nav a::after {
      content: "";
      position: absolute;
      bottom: -2px;
      left: 0;
      width: 0;
      height: 2px;
      background: var(--yellow-dark);
      transition: width 0.3s ease;
    }

    nav a:hover::after,
    nav a.active::after {
      width: 100%;
    }

    .menu-toggle {
      display: none;
      background: none;
      border: none;
      cursor: pointer;
    }
    .menu-toggle span {
      display: block;
      width: 25px;
      height: 3px;
      background: var(--yellow);
      margin: 4px 0;
      transition: transform 0.3s;
    }

    @media (max-width: 768px) {
      .menu-toggle { display: block; }
      nav { display: none; width: 100%; }
      nav ul {
        flex-direction: column;
```

```
        background: var(--purple);
        padding: 1rem 0;
      }
      header.open nav { display: block; }
    }

    footer {
      background: var(--purple);
      color: var(--yellow);
      text-align: center;
      padding: 1rem 0;
    }

    .hero {
      position: relative;
      background: var(--background);
      padding: 6rem 0;
      overflow: hidden;
      color: var(--text-dark);
    }
    .hero-inner {
      display: flex;
      align-items: center;
      gap: 2rem;
      flex-wrap: wrap;
    }
    .hero-text {
      flex: 1;
    }
    .hero-text h1 {
      font-size: 2.5rem;
    }
    .hero-img img {
      width: 320px;
      height: 320px;
      object-fit: cover;
      border-radius: 50%;
      border: 12px solid var(--yellow-dark);
      box-shadow: 0 8px 24px rgba(0,0,0,0.12);
    }
    .hero::before {
      content: "";
      position: absolute;
      top: -20%;
      left: 0;
      width: 200%;
      height: 50%;
      background: var(--purple-light);
      clip-path: ellipse(70% 60% at 50% 100%);
      z-index: -1;
    }
    .hero::after {
      content: "";
      position: absolute;
      bottom: -25%;
      left: -25%;
      width: 150%;
      height: 40%;
```

```
    background: var(--purple);
    clip-path: ellipse(80% 50% at 50% 0%);
    z-index: -1;
  }

  .services-section {
    padding: 4rem 0;
  }
  .services-grid {
    display: grid;
    gap: var(--gap);
    grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));
  }
  .services-grid .card h3 {
    margin-top: 0;
  }
  .services-grid .card p {
    margin-bottom: 0;
  }

  .gallery-section {
    padding: 4rem 0;
  }
  .gallery-section h2 {
    text-align: center;
    margin-bottom: var(--gap);
    color: var(--purple);
  }
  .gallery-grid {
    display: grid;
    gap: var(--gap);
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  }
  .gallery-item {
    position: relative;
    overflow: hidden;
    border-radius: 12px;
  }
  .gallery-item img {
    width: 100%;
    aspect-ratio: 1/1;
    object-fit: cover;
    transition: transform 0.3s ease;
  }
  .gallery-item:hover img {
    transform: scale(1.05);
  }
  .gallery-item figcaption {
    position: absolute;
    bottom: 0;
    left: 0;
    width: 100%;
    padding: 0.5rem;
    background: rgba(0,0,0,0.5);
    color: #fff;
    font-size: 0.9rem;
    text-align: center;
    opacity: 0;
```

```
    transition: opacity 0.2s;
  }
  .gallery-item:hover figcaption {
    opacity: 1;
  }

  .test-section {
    padding: 4rem 0;
  }
  .test-section h2 {
    text-align: center;
    color: var(--purple);
    margin-bottom: var(--gap);
  }

  .test-form-card {
    max-width: 700px;
    margin: 0 auto;
    padding: var(--gap);
    background: #fff;
    border-radius: 16px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.08);
  }

  .question-card {
    margin-bottom: var(--gap);
    padding: var(--gap);
    border: 1px solid rgba(0,0,0,0.1);
    border-radius: 12px;
    background: #fafafa;
  }

  .question-text {
    font-weight: 600;
    margin-bottom: 0.75rem;
  }

  /* Список варіантів */
  .options-list {
    list-style: none;
    padding: 0;
    margin: 0;
  }
  .options-list li {
    margin-bottom: 0.5rem;
  }
  .options-list input {
    margin-right: 0.5rem;
    accent-color: var(--purple);
  }

  .test-submit {
    display: block;
    margin: 2rem auto 0;
    padding: 0.8rem 2rem;
    background: var(--purple);
    color: #fff;
    border: none;
```

```
    border-radius: 12px;
    font-size: 1.1rem;
    font-weight: 600;
    transition: opacity 0.2s;
    cursor: pointer;
  }
  .test-submit:hover {
    opacity: 0.85;
  }

  @media (max-width: 600px) {
    .test-form-card {
      padding: calc(var(--gap) / 2);
    }
    .test-submit {
      width: 100%;
      padding: 0.8rem 1rem;
    }
  }

  .form-section {
    padding: 4rem 0;
  }
  .form-section h2 {
    text-align: center;
    color: var(--purple);
    margin-bottom: var(--gap);
  }

  .flash-messages {
    margin-bottom: 1rem;
  }
  .alert {
    padding: 1rem;
    background-color: #DFF2BF;
    color: #4F8A10;
    border-radius: 12px;
    text-align: center;
    font-weight: 600;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
  }

  .form-card {
    max-width: 600px;
    margin: 0 auto;
    padding: var(--gap);
    background: #fff;
    border-radius: 16px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.05);
  }

  .form-group {
    display: flex;
    flex-direction: column;
    margin-bottom: var(--gap);
  }

  .form-group label {
```

```
    font-weight: 600;
    margin-bottom: 0.5rem;
  }

  .form-group input,
  .form-group textarea {
    width: 100%;
    padding: 0.8rem 1rem;
    border: 2px solid var(--purple);
    border-radius: 12px;
    font-size: 1rem;
    resize: vertical;
  }

  .btn--yellow {
    display: inline-block;
    width: auto%;
    padding: 0.7rem 1.4rem;
    background: var(--yellow-dark);
    color: var(--text-dark);
    border: none;
    border-radius: 12px;
    font-weight: 600;
    transition: background 0.2s;
    cursor: pointer;
  }
  .btn--yellow:hover {
    background: var(--yellow);
  }

  .alert {
    padding: 1rem;
    background-color: #DFF2BF;
    color: #4F8A10;
    border-radius: 12px;
    text-align: center;
    font-weight: 600;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    margin-bottom: var(--gap);
    transition: all 0.5s ease;
    max-width: 600px;
    margin-left: auto;
    margin-right: auto;
  }

  @media (max-width: 768px) {
    .menu-toggle {
      display: block;
    }
    nav {
      display: none;
      width: 100%;
    }
    nav ul {
      flex-direction: column;
      align-items: center;
      gap: 1rem;
      padding: 1rem 0;
```

```
          background: var(--purple);
        }
        header.open nav {
          display: block;
        }
      }
      @media (max-width: 600px) {
        .hero-text h1 {
          font-size: 1.8rem;
        }
        .hero-img img {
          width: 240px;
          height: 240px;
        }
        .header-inner {
          padding: 0.5rem 1rem;
        }
}
```

ДОДАТОК В

Python-файли

Лістинг коду run.py

```
from app import create_app
app = create_app()
if __name__ == "__main__":
   app.run(debug=True)
```

Лістинг коду __init__.py

```
from flask import Flask
from flask_wtf import CSRFProtect
from flask_mail import Mail
from .config import DevelopmentConfig
csrf = CSRFProtect()
mail = Mail()
def create_app(config_class=DevelopmentConfig):
    app = Flask(
        __name__,
        static_folder="static",
        template_folder="templates"
    )
    app.config.from_object(config_class)
    csrf.init_app(app)
    mail.init_app(app)
    from .routes import main_bp
    app.register_blueprint(main_bp)
return app
```

Лістинг коду config.py

```
import os

class Config:
    SECRET_KEY = "X0Z3aBj--mZ2N9pVpPqLQe_8vGk8hT_7Y5RkM_Q7WlQ"
    WTF_CSRF_TIME_LIMIT = None
    MAIL_SERVER   = os.environ.get("MAIL_SERVER", "smtp.gmail.com")
    MAIL_PORT     = int(os.environ.get("MAIL_PORT", 587))
    MAIL_USE_TLS  = os.environ.get("MAIL_USE_TLS", "true").lower() in ("true",
"1")
    MAIL_USE_SSL  = os.environ.get("MAIL_USE_SSL", "false").lower() in ("true",
"1")
    MAIL_USERNAME = os.environ.get("MAIL_USERNAME")
    MAIL_PASSWORD = os.environ.get("MAIL_PASSWORD")
    MAIL_DEFAULT_SENDER = os.environ.get("MAIL_DEFAULT_SENDER", MAIL_USERNAME)
class DevelopmentConfig(Config):
    DEBUG = True
class ProductionConfig(Config):
DEBUG = False
```

Лістинг коду forms.py

```
    from flask_wtf import FlaskForm
    from wtforms import StringField, TelField, TextAreaField, SubmitField
    from wtforms.validators import DataRequired

    class ContactForm(FlaskForm):
        name = StringField("Ім’я", validators=[DataRequired()])
        tel  = TelField("Телефон", validators=[DataRequired()])
        msg  = TextAreaField("Повідомлення")
        submit = SubmitField("Надіслати")</section>
{% endblock %}
```

Лістинг коду logic.py

```
    from collections import namedtuple
    Question = namedtuple("Question", ["id", "text", "options"])
    TEST = [
        Question(
            id="q1",
            text="Як часто дитина відчуває напруження?",
            options=[("0", "Ніколи"), ("1", "Іноді"), ("2", "Часто")]
        ),
        Question(
            id="q2",
            text="Чи бувають труднощі з концентрацією уваги?",
            options=[("0", "Ніколи"), ("1", "Іноді"), ("2", "Часто")]
        ),
        Question(
            id="q3",
            text="Наскільки легко дитина засинає увечері?",
            options=[("0", "Легко"), ("1", "Потрібна допомога"), ("2", "Важко")]
        ),
        Question(
            id="q4",
            text="Як часто дитина реагує на гучні звуки надмірно емоційно?",
            options=[("0", "Ніколи"), ("1", "Іноді"), ("2", "Завжди")]
        ),
        Question(
            id="q5",
            text="Чи може дитина довго сидіти на одному місці під час занять?",
            options=[("0", "Легко може"), ("1", "Іноді встає"), ("2", "Не може")]
        ),
        Question(
            id="q6",
            text="Як часто дитина виявляє надмірну рухливість (гіперактивність)?",
            options=[("0", "Ніколи"), ("1", "Іноді"), ("2", "Часто")]
        ),
        Question(
            id="q7",
            text="Чи вміє дитина виражати свої емоції словами?",
            options=[("0",  "Завжди"),  ("1",  "Іноді  плутається"),  ("2",  "Важко
висловити")]
        ),
        Question(
            id="q8",
```

```
            text="Як швидко дитина відновлюється після стресу (наприклад, сварки чи
падіння)?",
            options=[("0", "Швидко"), ("1", "Треба час"), ("2", "Дуже довго")]
        ),
        Question(
            id="q9",
            text="Як дитина реагує на різні тактильні відчуття (наприклад, текстуру
тканин)?",
            options=[("0", "Спокійно"), ("1", "Іноді дискомфорт"), ("2", "Завжди
незручно")]
        ),
        Question(
            id="q10",
            text="Наскільки спритно дитина виконує дрібну моторику (з'єднує дрібні
предмети)?",
            options=[("0", "Легко"), ("1", "Іноді тупцює"), ("2", "Потребує значної
допомоги")]
        ),
    ]
    def evaluate(total_score: int) -> dict:
        if total_score <= 6:
            level = "Низький рівень напруження"
            advice = "Продовжуйте підтримувати режим та ігри для релаксації."
        elif total_score <= 13:
            level = "Помірний рівень напруження"
            advice = "Додавайте дихальні вправи та тактильні ігри."
        else:
            level = "Високий рівень напруження"
            advice = "Рекомендуємо звернутися до фахівця для детальної оцінки."
    return {"level": level, "advice": advice}
```

Лістинг коду routes.py

```
    from flask import Blueprint, render_template, request, redirect, url_for, flash
    from flask_mail import Message
    from . import mail
    from .forms import ContactForm
    from .logic import TEST, evaluate

    main_bp = Blueprint("main", __name__)
    gallery_items = [
        {
            "src": "img/mirror_sticks.jpg",
            "alt": "Дзеркальне відображення",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DJO-rsrsF77/",
        },
        {
            "src": "img/spatial_figures.jpg",
            "alt": "Просторове мислення",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DIWLfk_M98w/",
        },
        {
            "src": "img/fingers_vocab.jpg",
            "alt": "Пальці + Рух = Мова",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DIMom-BsONl/",
        },
```

```
        {
            "src": "img/sujok.jpg",
            "alt": "Су-джок терапія",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DIEZ6JNstyr/",
        },
        {
            "src": "img/proprioceptive.jpg",
            "alt": "Пропріоцептивна система",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DIBa-CRs9Hd/",
        },
        {
            "src": "img/home_neuro.jpg",
            "alt": "Домашня нейрокорекція",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DH3yS66MoSn/",
        },
        {
            "src": "img/vestibular.jpg",
            "alt": "Вестибулярна система",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DHsbn0OMfqU/",
        },
        {
            "src": "img/motor_skills.jpg",
            "alt": "Велика моторика",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DHY7kvospY4/",
        },
        {
            "src": "img/neurogames.jpg",
            "alt": "Нейроігри",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DEUuxvwMc7t/",
        },
        {
            "src": "img/progress.jpg",
            "alt": "Розвиток міжпівкульних зв'язків",
            "url": "https://www.instagram.com/ellia_tatarenko/reel/DDFN-e7sunT/",
        },
    ]
    @main_bp.route("/")
    def home():
        return render_template("home.html", title="Головна")

    @main_bp.route("/about")
    def about():
        return render_template("about.html", title="Про нас")

    @main_bp.route("/services")
    def services():
        return render_template("services.html", title="Методики")

    @main_bp.route("/gallery")
    def gallery():

        return render_template(
            "gallery.html",
            title="Галерея",
            items=gallery_items,
        )

    @main_bp.route("/contact", methods=["GET", "POST"])
```

```
    def contact():
        form = ContactForm()
        if form.validate_on_submit():
            msg = Message(
                subject=f"Нове звернення з сайту «Акула» від {form.name.data}",
                recipients=["your.recipient@example.com"],
                body=(
                    f"Ім’я:    {form.name.data}\n"
                    f"Телефон:{form.tel.data}\n\n"
                    f"Повідомлення:\n{form.msg.data}"
                ),
            )
            try:
                mail.send(msg)
                flash("Дякую за ваше звернення", "success")
            except Exception as e:
                print(f"[MAIL_ERROR] {e}")
                flash("Ой-ой, сталася помилка при відправці", "error")
            return redirect(url_for("main.contact"))

        return render_template("contact.html", title="Контакти", form=form)


    @main_bp.route("/test", methods=["GET", "POST"])
    def test():

        if request.method == "POST":
            total = sum(int(request.form[q.id]) for q in TEST)
            context = evaluate(total) | {"score": total}
            return render_template("result.html", title="Результати", **context)
    return render_template("test.html", title="Тест", questions=TEST)
```