Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка вебсайту “SleepHelper” з використанням React та Python Fast Api

Виконала:студентка IV курсу, групи СТ-41

спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

_____ Дубчак Л.І.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Яцишин В.В.
(підпис) (прізвище та ініціали)

Тернопіль 2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Дубчак Лілії Іванівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебсайту "SleepHelper" з використанням React та Python Fast Api.

Керівник роботи Дмитроца Леся Павлівна, к.т.н. доц., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*30*» *березня* 2026 року № *4/9-162*

2. Термін подання студентом завершеної роботи 19 червня 2026 р.

3. Вихідні дані до роботи Наукові та навчальні джерела з веброзробки, клієнт-серверної архітектури, технологій React, TypeScript, FastAPI та MongoDB. Документація React, React Router, TypeScript, FastAPI, MongoDB Atlas, JWT Authentication.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області та визначення вимог до вебсайту "sleephelper". 1.1 Актуальність контролю якості сну та покращення відпочинку. 1.2 Огляд сучасних підходів до аналізу та покращення відпочинку. 1.3 Порівняльний аналіз існуючих програмних рішень для трекінгу сну. 1.4 Функціональні та нефункціональні вимоги до вебсайту.1.5 Постановка задачі на розробку системи.Розділ 2. Обґрунтування технологічного стеку та проєктування системи. 2.1 Вибір та обґрунтування технологій для фронтенду. 2.2 Вибір технологій для бекенду та бази даних. 2.3 Проєктування структури бази даних. 2.4 Проєктування REST API та взаємодії компонентів. 2.5 Архітектура вебзастосунку та структура інтерфейсу. Розділ 3. Програмна реалізація та тестування вебсайту. 3.1 Створення серверної частини на FastAPI.3.2 Реалізація системи реєстрації та автоматизації користувачів 3.3Розробка модулів збереження та обробки даних про сон.3.4Реалізація клієнтської частини. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Актуальність теми та проблематика порушень сну. 3. Мета та завдання дипломної роботи. 4. Архітектура вебзастосунку Sleep Helper. 5. Структура клієнтської та серверної частин системи. 6. Реалізація системи реєстрації та авторизації користувачів. 7. Інтерактивний щоденник сну. 8. Модуль додавання та редагування записів про сон.9.Результати тестування вебзастосунку.10. Перспективи розвитку системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я. (к.т.н., доцент)		

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі	27.01.2026-15.02.2026	*Виконано*
	кваліфікаційної роботи		
3.	Виконання дослідження щодо аналогів вебсайтів	16.02.2026-10.05.2026	*Виконано*
	Розроблення клієнтської та серверної частин вебсайту		
	“SleepHelper”		
4.	Оформлення розділу «Аналіз предметної області та	11.05.2026-17.05.2026	*Виконано*
	визначення вимог до вебсайту “sleephelper”»		
5.	Оформлення розділу «Обґрунтування технологічного	18.05.2026-24.05.2026	*Виконано*
	стеку та проєктування системи»		
6.	Оформлення розділу «Програмна реалізація та	25.05.2026-31.05.2026	*Виконано*
	тестування вебсайту»		
7.	Виконання завдання до підрозділу «Безпека	01.06.2026-07.06.2026	*Виконано*
	життєдіяльності»		
8.	Виконання завдання до підрозділу «Основи охорони	01.06.2026-07.06.2026	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	08.06.2026-11.06.2026	*Виконано*
10.	Нормоконтроль	12.06.2026-14.06.2026	*Виконано*
11.	Перевірка на плагіат	16.06.2026	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.06.2026	*Виконано*
13.	Захист кваліфікаційної роботи	22.06.2026	

Студент _____ Дубчак Л.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка вебсайту “SleepHelper” з використанням React та Python Fast Api // Кваліфікаційна робота освітнього ступеня «Бакалавр» // Дубчак Лілія Іванівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СТ-41 // Тернопіль, 2026 // С. 94 , рис. – 34 , табл. – 6, кресл. – 16 , додат. – 5 , бібліогр. – 39 .

Ключові слова: вебзастосунок, сон, моніторинг сну, React, TypeScript, FastAPI, MongoDB, статистика сну.

Кваліфікаційна робота присвячена розробці вебзастосунку для моніторингу та аналізу якості сну користувачів «SleepHelper».

У першому розділі кваліфікаційної роботи описано теоретичні аспекти моніторингу сну та сучасні підходи до створення вебзастосунків для збору й аналізу даних. Висвітлено особливості використання сучасних вебтехнологій для розробки клієнтських і серверних частин програмних систем. Розглянуто можливості фреймворків React, Angular та бібліотеки React для створення користувацьких інтерфейсів. Проаналізовано сучасні технології та інструменти веброзробки.

У другому розділі кваліфікаційної роботи виконано проєктування вебзастосунку «SleepHelper». Досліджено архітектуру системи, структуру бази даних та принципи взаємодії між компонентами. Подано структуру REST API, схеми взаємодії клієнтської та серверної частин, а також логіку роботи основних функціональних модулів.

У третьому розділі кваліфікаційної роботи описано процес програмної реалізації вебзастосунку. Проаналізовано особливості розробки серверної частини на FastAPI та клієнтської частини на React і TypeScript. Проведено тестування функціональності системи та оцінено відповідність реалізованого програмного забезпечення поставленим вимогам.

ANNOTATION

Development of the "SleepHelper" Website Using React and Python FastAPI // Qualification work of the educational level «Bachelor» // Dubchak Liliia // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group ST-41 // Ternopil, 2026 // P. 94 , fig. – 34 , tabl. – 6 , chair. – 16 , annexes. – 5 , references – 39.

Keywords: web application, sleep, sleep monitoring, React, TypeScript, FastAPI, MongoDB, sleep statistics.

The qualification thesis is devoted to the development of the SleepHelper web application for monitoring and analyzing users' sleep quality.

The first chapter of the qualification thesis describes the theoretical aspects of sleep monitoring and modern approaches to the development of web applications for data collection and analysis. The features of using modern web technologies for developing client-side and server-side software systems are highlighted. The capabilities of React, Angular, and modern web development tools are considered. Current technologies and approaches to web application development are analyzed.

The second chapter of the qualification thesis focuses on the design of the SleepHelper web application. The system architecture, database structure, and principles of interaction between software components are investigated. The structure of the REST API, client-server interaction schemes, and the logic of the main functional modules are presented.

The third chapter of the qualification thesis describes the implementation process of the web application. The features of developing the server side using FastAPI and the client side using React and TypeScript are analyzed. Functional testing of the system is carried out, and the compliance of the developed software with the specified requirements is evaluated.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – програмний інтерфейс застосунку.

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів.

HTML (англ. HyperText Markup Language) – мова розмітки гіпертекстових документів.

HTTP (англ. HyperText Transfer Protocol) – протокол передачі гіпертексту.

JSON (англ. JavaScript Object Notation) – формат обміну даними.

JWT (англ. JSON Web Token) – токен автентифікації користувача.

MongoDB – документоорієнтована NoSQL система керування базами даних.

NoSQL (англ. Not Only SQL) – клас нереляційних систем керування базами даних.

REST API – архітектурний стиль взаємодії між клієнтом і сервером через HTTP-запити.

SPA (англ. Single Page Application) – односторінковий вебзастосунок.

TS (англ. TypeScript) – типізована мова програмування, що є надбудовою над JavaScript.

UI (англ. User Interface) – користувацький інтерфейс.

URL (англ. Uniform Resource Locator) – адреса ресурсу в мережі Інтернет.

БД – база даних.

ВООЗ – Всесвітня організація охорони здоров’я.

СУБД – система керування базами даних.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку інформаційних технологій та зростання темпу життя все більше людей стикаються з проблемами порушення режиму сну. Недостатня тривалість або низька якість сну негативно впливають на фізичне здоров'я, працездатність, концентрацію уваги, емоційний стан та загальне самопочуття людини. За даними численних досліджень, регулярне недосипання може призводити до зниження продуктивності праці, погіршення когнітивних функцій та підвищення ризику розвитку різних захворювань.

Водночас розвиток цифрових технологій та широке використання вебзастосунків створюють нові можливості для контролю та аналізу показників здоров'я. Використання спеціалізованих програмних засобів дозволяє користувачам накопичувати інформацію про власний сон, відстежувати зміни його якості та отримувати рекомендації щодо покращення режиму відпочинку. Особливо актуальним є створення вебзастосунків, які забезпечують доступ до інформації з будь-якого пристрою без необхідності встановлення додаткового програмного забезпечення.

Сучасні вебтехнології дозволяють поєднати можливості збору даних, їх аналізу та наочного відображення результатів у єдиній інформаційній системі. Використання інтерактивних графіків, статистичних звітів та допоміжних інструментів сприяє підвищенню зацікавленості користувачів у контролі власного режиму сну та формуванні здорових звичок.

Тому розробка вебзастосунків для моніторингу та аналізу якості сну є актуальним напрямком сучасних досліджень у галузі інформаційних технологій та веброзробки.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка вебзастосунку для моніторингу та аналізу якості сну користувачів, який забезпечує ведення щоденника сну, накопичення статистичних даних та надання рекомендацій щодо покращення

режиму відпочинку. Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати сучасні підходи та програмні засоби для моніторингу сну;
- дослідити технології розробки клієнтської та серверної частин вебзастосунків;
- обґрунтувати вибір React, TypeScript, FastAPI та MongoDB для реалізації системи;
- спроєктувати архітектуру вебзастосунку та структуру бази даних;
- реалізувати систему реєстрації та авторизації користувачів;
- розробити інтерактивний щоденник сну для збереження інформації про сон;
- реалізувати калькулятор часу засинання та пробудження;
- розробити модуль статистики та візуалізації даних про сон;
- реалізувати модуль рекомендацій та релаксації;
- провести тестування функціональних можливостей вебзастосунку.

Об’єкт дослідження. Об’єктом дослідження є процес моніторингу та аналізу показників сну користувачів із використанням сучасних вебтехнологій.

Предмет дослідження. Предметом дослідження є методи, засоби та програмні технології розробки вебзастосунків для збору, зберігання, обробки та візуалізації даних про сон користувачів.

Практичне значення одержаних результатів. Практичне значення роботи полягає у створенні вебзастосунку «SleepHelper», який може використовуватися для ведення щоденника сну, аналізу статистичних показників та контролю режиму відпочинку користувачів. Розроблена система дозволяє накопичувати дані про сон, відображати їх у вигляді графіків і діаграм, а також надавати рекомендації щодо покращення якості сну.

Результати роботи можуть бути використані як основа для подальшого розвитку програмного продукту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ ДО ВЕБСАЙТУ “SLEEPHELPER”

1.1 Актуальність контролю якості сну та покращення відпочинку

Сон – це природний процес, який допомагає організму відновлювати енергію, підтримує навчання та пам’ять, а також зберігає здоров’я [1]. Під час сну відбуваються процеси регенерації тканин, зміцнення імунної системи, нормалізація обмінних процесів, а також обробка та систематизація інформації, отриманої протягом дня. Перш за все, якість сну впливає на працездатність людини, її когнітивні здібності, емоційний стан та загальний рівень здоров’я.

У сучасному суспільстві проблема порушення сну стає все більш актуальною. Постійний розвиток інформаційних технологій, високий рівень стресу, значне робоче навантаження, конкурентне середовище та активне використання цифрових пристроїв призводять до порушення режиму відпочинку. Багато людей стикаються з труднощами засинання, недостатньою тривалістю сну або частими пробудженнями вночі. Внаслідок цього знижується концентрація уваги, погіршується пам’ять, зростає ризик виникнення серцево-судинних захворювань, цукрового діабету, ожиріння та інших проблем зі здоров’ям.

Відповідно до рекомендацій Центру контролю та профілактики захворювань США (CDC), для підтримання належного рівня фізичного та психічного здоров’я дорослій людині рекомендується спати не менше семи годин на добу. Тривалість сну є одним із ключових факторів, що впливають на працездатність, концентрацію уваги, емоційний стан та загальне самопочуття людини. Водночас потреба у сні змінюється залежно від віку, тому для різних вікових категорій встановлено окремі рекомендації щодо оптимальної тривалості відпочинку. На рисунку 1.1 наведено стовпчасту діаграму, яка

відображає рекомендовану кількість годин сну для різних вікових груп відповідно до рекомендацій CDC [2].

Рисунок 1.1 – Тривалість сну для різних вікових груп

Найбільша потреба у сні спостерігається у новонароджених та немовлят. Оскільки, в цей період відбувається активний фізичний розвиток організму, формування нервової системи та розвиток головного мозку. Саме під час сну відбуваються процеси росту, зміцнення імунної системи та закріплення нових навичок.

Для дітей дошкільного та молодшого шкільного віку достатня тривалість сну є необхідність для формування нормального фізичного та психічного розвитку. Недосипання в цьому віці може негативно впливати на здатність до навчання, концентрацію уваги, пам’ять та емоційний стан дитини.

Підлітковий вік характеризується інтенсивними фізіологічними змінами та підвищеним навантаженням на нервову систему. Саме тому, підліткам рекомендується спати від восьми до десяти годин на добу. Недостатній сон у цей період може призводити до погіршення успішності, підвищеної втомлюваності та емоційної нестабільності.

Для дорослих людей сон залишається важливим чинником підтримки здоров’я та працездатності. Повноцінний нічний відпочинок сприяє відновленню енергетичних ресурсів організму, покращує концентрацію уваги, знижує рівень стресу та ризик розвитку багатьох захворювань. Згідно з сучасними рекомендаціями, більшості дорослих достатньо від семи до дев’яти годин сну на добу.

Для людей похилого віку рекомендована тривалість сну дещо зменшується, однак його якість залишається надзвичайно важливою. У цьому віці повноцінний сон сприяє підтриманню когнітивних функцій, покращує самопочуття та допомагає зберігати активний спосіб життя.

Сон людини складається з послідовних циклів, кожен із яких включає декілька стадій. Середня тривалість одного циклу становить приблизно 90–110 хвилин, а протягом ночі людина проходить від чотирьох до шести таких циклів. Кожна стадія виконує певну функцію та відіграє важливу роль у процесах відновлення організму [3].

На початковому етапі людина переходить у стан сонливості, під час якого поступово знижується активність нервової системи та сповільнюється реакція на зовнішні подразники. Далі настає фаза легкого сну, яка характеризується зниженням частоти серцевих скорочень, температури тіла та розслабленням м’язів.

Після легкого сну організм переходить до більш глибоких стадій відпочинку. У цей період відбуваються процеси фізичного відновлення, регенерації тканин, зміцнення імунної системи та накопичення енергетичних ресурсів. Важливе значення має фаза глибокого сну, яка забезпечує найбільш ефективне відновлення організму після фізичних та розумових навантажень.

Завершальною стадією циклу є REM-сон (Rapid Eye Movement), або фаза швидкого руху очей. Саме під час цієї стадії спостерігається висока активність головного мозку, виникає більшість сновидінь та відбувається обробка інформації, отриманої протягом дня. REM-сон відіграє важливу роль у процесах навчання, формування пам’яті та підтримання психоемоційного

здоров’я людини. Схематичне представлення основних стадій сну наведено на рисунку 1.2.

Рисунок 1.2 – Основні стадії циклу сну людини

Порушення структури сну або недостатня тривалість окремих його фаз можуть призводити до підвищеної втомлюваності, погіршення концентрації уваги, зниження продуктивності та розвитку різноманітних захворювань. Звідси і причина створення сучасних систем моніторингу сну, які орієнтовані не лише на визначення загальної тривалості відпочинку, а й на аналіз його якості, регулярності та структури.

На якість сну впливає об’ємна кількість зовнішніх та внутрішніх факторів, серед яких: режим дня, рівень фізичної активності, харчові звички, емоційний стан людини та умови навколишнього середовища. Ось чому для забезпечення повноцінного відпочинку важливо дотримуватися рекомендацій щодо здорового сну, які допомагають покращити процес засинання, підтримувати стабільну структуру сну та зменшувати вплив негативних

чинників. Основні рекомендації щодо організації здорового режиму сну наведено на рисунку 1.3.

Рисунок 1.3 – Рекомендації щодо здорового сну

Дотримання наведених правил дозволяє покращити процес засинання, підвищити якість нічного відпочинку та забезпечити повноцінне відновлення організму.

Серед найбільш важливих рекомендацій можна виділити дотримання постійного режиму сну, створення комфортних і чистих умов у спальні, використання розслаблюючих процедур перед відпочинком [4]. Водночас, використання електронних пристроїв перед сном, вживання кофеїновмісних напоїв, надмірне емоційне навантаження та інтенсивна фізична активність у вечірній час можуть спричиняти порушення структури сну та погіршення його якості.

Незважаючи на доступність таких рекомендацій, значна кількість людей не дотримується здорового режиму відпочинку. Саме тому виникає потреба у

використанні сучасних інформаційних систем, які дозволяють здійснювати моніторинг режиму сну, накопичувати статистичні дані та аналізувати фактори, що впливають на його якість.

1.2 Огляд сучасних підходів до аналізу та покращення відпочинку

Розробка вебсайту «SleepHelper» дозволяє втілити сучасну інформаційну систему для моніторингу якості сну, яка буде доступною для широкого кола користувачів. Вебсайт не потребує встановлення додаткового програмного забезпечення та може використовуватися на будь-якому пристрої, що має веббраузер і доступ до мережі Інтернет. Це однозначно відрізняється від програми, яку потрібно завантажувати.

Вебсайт «SleepHelper» має безліч переваг, які роблять його зручним інструментом для відстеження якості сну та формування здорових звичок відпочинку. Використання веборієнтованого підходу дозволяє забезпечити доступність системи для широкого кола користувачів незалежно від типу пристрою та місця перебування [5].

Завдяки адаптивному інтерфейсу користувач може працювати із системою за допомогою персонального комп’ютера, ноутбука, планшета або смартфона. Такий підхід підвищує зручність використання системи та дозволяє отримувати доступ до даних у будь-який момент часу. На рисунку 1.4 продемонстровано різні ґаджети та їх використання вебсайту.

Рисунок 1.4 – Використання вебсайту на різних пристроях

Однією з основних переваг вебсайту є можливість роботи через браузер. Користувачеві не потрібно встановлювати додаткове програмне забезпечення або регулярно оновлювати застосунок, оскільки всі зміни та покращення реалізуються на серверній частині системи. Це значно спрощує процес використання вебресурсу та забезпечує постійний доступ до актуальної версії програмного забезпечення. Основні переваги використання веб сайту продемонстровано на рисунку 1.5.

Рисунок 1.5 – Основні переваги вебсайту

Централізоване зберігання інформації забезпечує надійність роботи системи та спрощує керування даними користувачів. Усі записи про сон, статистичні показники та персональні налаштування зберігаються на сервері, що дозволяє уникнути втрати інформації у випадку пошкодження або заміни пристрою користувача.

Однією з ключових функцій вебсайту є ведення щоденника сну. Користувач може фіксувати час засинання та пробудження, оцінювати якість відпочинку, додавати власні нотатки та спостереження. Накопичення таких

даних дозволяє більш детально аналізувати особливості режиму сну та виявляти фактори, які впливають на його якість.

Для підвищення інформативності системи передбачено модуль статистики та історії записів. За допомогою графіків і діаграм користувач отримує можливість відстежувати зміни тривалості сну, аналізувати власні звички та оцінювати ефективність заходів, спрямованих на покращення режиму відпочинку. Візуалізація даних значно спрощує процес їх аналізу та сприяє прийняттю більш обґрунтованих рішень.

Додатковою функціональною можливістю системи є калькулятор сну, який допомагає визначити оптимальний час засинання або пробудження з урахуванням тривалості циклів сну. Використання такого інструменту дозволяє користувачам планувати власний режим відпочинку більш ефективно та зменшувати негативний вплив недосипання на організм.

Не дивлячись на значну кількість переваг, вебсайт «SleepHelper» має певні обмеження, характерні для більшості веборієнтованих інформаційних систем. Їх врахування є важливим етапом проєктування та подальшого розвитку програмного продукту.

Одним із основних недоліків є залежність від стабільного підключення до мережі Інтернет. Оскільки система працює за клієнт-серверною архітектурою, доступ до більшості функцій можливий лише за наявності мережевого з'єднання. У разі відсутності доступу до Інтернету користувач не зможе переглядати статистику, створювати нові записи або синхронізувати дані з сервером.

Ще одним обмеженням є необхідність створення облікового запису для збереження персональних даних. З одного боку, реєстрація забезпечує індивідуальний доступ до інформації та її захист, однак частина користувачів може сприймати цей процес як додаткову перешкоду під час початку роботи із системою [6]. На рисунку 1.6 зібрані основні недоліки створення вебсайту.

Рисунок 1.6 – Основні недоліки вебсайтів

Особливістю вебсайту «SleepHelper» є те, що він не здійснює автоматичний збір фізіологічних показників користувача. На відміну від спеціалізованих смартгодинників або фітнес-браслетів, система не має прямого доступу до датчиків серцевого ритму, рівня насичення крові киснем чи показників рухової активності під час сну. У зв'язку з цим аналіз базується переважно на інформації, яку користувач вводить самостійно.

Також точність отриманих результатів переважною мірою залежить від достовірності введених даних. Помилки під час внесення часу засинання, пробудження або оцінки якості сну можуть впливати на коректність статистичних розрахунків та подальших рекомендацій системи.

Важливо зазначити, що вебсайт «SleepHelper» призначений для самостійного моніторингу режиму сну та формування корисних звичок, та не є медичною діагностичною системою. Оскільки розробка здійснюється без участі лікарів або спеціалістів зі сну, надані рекомендації мають загальний інформаційний характер. У разі наявності серйозних або тривалих порушень сну користувач повинен звернутися до лікаря! Звідси і виникають можливі ризики створення вебсайту, пов’язаних із темою здоров’я.

Під час розробки будь-якої веборієнтованої інформаційної системи варто враховувати потенційні ризики, які можуть впливати на її надійність, безпеку та коректність функціонування. Вебсайт «SleepHelper» не є винятком, оскільки його робота пов'язана зі зберіганням персональних даних користувачів, обробкою статистичної інформації та взаємодією між клієнтською та серверною частинами системи.

Одним із найбільш поширених ризиків є несанкціонований доступ до облікових записів користувачів [7]. Причинами виникнення таких ситуацій можуть бути використання слабких паролів, компрометація облікових даних або вразливості в механізмах аутентифікації. Для зменшення ймовірності виникнення подібних ситуацій необхідно використовувати сучасні методи захисту, насамперед хешування паролів та механізми авторизації на основі токенів.

Важливим аспектом також є захист персональних даних. Оскільки система зберігає інформацію про режим сну, статистичні показники та індивідуальні налаштування користувачів, витік або несанкціоноване розголошення таких даних може негативно вплинути на довіру до системи та порушити вимоги інформаційної безпеки.

Окремим ризиком є можливість втрати даних унаслідок помилок бази даних, апаратних збоїв або некоректного резервного копіювання. Для мінімізації наслідків таких ситуацій необхідно використовувати механізми резервного зберігання інформації та регулярного створення резервних копій.

Певну загрозу становлять також помилки під час взаємодії клієнтської частини, реалізованої за допомогою React, та серверної частини на базі FastAPI [8]. Некоректна передача даних, помилки API або проблеми сумісності можуть призводити до збоїв окремих функцій системи та погіршення користувацького досвіду.

Зі збільшенням кількості користувачів можуть виникати проблеми масштабування системи. Зростання навантаження на сервер, базу даних та мережеву інфраструктуру може негативно впливати на швидкодію вебсайту.

Саме тому під час проєктування необхідно передбачити можливість подальшого розширення ресурсів та оптимізації продуктивності. На рисунку 1.7 продемонстровані основні ризики розбробки вебсайту “SleepHelder”.

Рисунок 1.7 – Можливі ризики розробки вебсайту

Особливу увагу слід приділити ризику відсутності медичної верифікації рекомендацій. Вебсайт «SleepHelper» створюється як інформаційна система для моніторингу сну та формування корисних звичок, а не як медичний програмний засіб. Розробка здійснюється без безпосередньої участі лікарів-сомнологів або інших медичних спеціалістів, тому рекомендації системи мають виключно інформаційний характер і не можуть використовуватися для встановлення діагнозу чи призначення лікування. У разі виявлення серйозних порушень сну користувач повинен звернутися до кваліфікованого медичного фахівця.

Таким чином, врахування можливих ризиків ще на етапі проєктування дозволяє підвищити надійність, безпечність та якість функціонування вебсайту «SleepHelper», а також забезпечити більш комфортне та безпечне використання системи кінцевими користувачами.

1.3 Порівняльний аналіз існуючих програмних рішень для трекінгу сну

Для формування функціональних вимог до вебсайту «SleepHelper» варто провести аналіз існуючих веб- та мобільних застосунків, призначених для моніторингу якості сну. Такий аналіз дозволяє визначити найбільш затребувані функціональні можливості, оцінити переваги та недоліки наявних рішень, а також виявити підходи, які можуть бути використані під час розробки власного програмного продукту.

Крім того, дослідження аналогів сприяє формуванню обґрунтованих вимог до майбутньої системи та дозволяє уникнути типових недоліків, характерних для існуючих застосунків. Для проведення аналізу було обрано популярні веб- та мобільні рішення, які використовуються для відстеження тривалості сну, оцінювання його якості, збору статистичних даних та формування рекомендацій щодо покращення режиму відпочинку.

Результати аналізу дають можливість визначити функції, які є найбільш корисними для користувачів, а також виявити напрями для вдосконалення існуючих підходів до моніторингу сну. На основі отриманих результатів формується перелік вимог до вебсайту «SleepHelper», який забезпечує поєднання зручного інтерфейсу, інструментів аналізу даних та засобів підтримки здорового режиму відпочинку.

1.3.1 Дослідження функціональних можливостей Sleep Cycle

Sleep Cycle є одним із найвідоміших мобільних застосунків для моніторингу сну, який використовується мільйонами користувачів у всьому світі [9]. Основною особливістю системи є використання технологій аналізу звуків та рухів користувача під час сну без необхідності застосування спеціалізованих медичних пристроїв або фітнес-браслетів. Для збору інформації додаток використовує мікрофон смартфона, акселерометр або смартгодинник, що дозволяє відстежувати поведінку користувача під час

відпочинку та визначати особливості його сну. На рисунку 1.8 зображено інтерфейс застосунку Sleep Cycle.

Рисунок 1.8 – Інтерфейс програми Sleep Cycle

Однією з ключових функцій Sleep Cycle є аналіз якості сну. Після завершення кожної ночі користувач отримує детальний звіт, який містить інформацію про тривалість сну, час засинання, кількість пробуджень, оцінку якості сну та зміну фаз сну протягом ночі. Зібрані дані відображаються у вигляді графіків та статистичних звітів, що дозволяє аналізувати зміни режиму сну в довгостроковій перспективі.

Важливою перевагою застосунку є функція Smart Alarm, тобто «розумний будильник». На відміну від традиційних будильників, система аналізує стан користувача під час сну та намагається розбудити його у фазі легкого сну протягом заданого часового інтервалу. Це дозволяє зменшити відчуття втоми після пробудження та зробити процес прокидання більш комфортним.

Додатково застосунок підтримує запис звуків під час сну. Система може фіксувати хропіння, кашель, розмови уві сні та інші нічні шуми, які потенційно

впливають на якість відпочинку. Отримані записи допомагають користувачам визначити можливі причини порушення сну та відстежувати їхню динаміку.

У 2024 році компанія представила технологію Sleep Stages, яка дозволяє визначати стадії сну, зокрема легкий сон, глибокий сон та REM-фазу. Для цього використовуються алгоритми аналізу дихання та рухів користувача, що збираються за допомогою смартфона. Таким чином, користувач отримує більш детальне уявлення про структуру власного сну та фактори, які впливають на його якість.

До основних переваг Sleep Cycle можна віднести сучасний інтерфейс, автоматичний збір даних, наявність розумного будильника, довгострокову статистику та відсутність потреби у додаткових пристроях. Серед недоліків можна виділити обмеження безкоштовної версії, залежність від датчиків смартфона, а також можливі похибки аналізу у шумному середовищі або при некоректному розташуванні пристрою під час сну.

1.3.2 Аналіз застосунку BetterSleep

BetterSleep допомагає користувачам у покращенні якості сну та процесу засинання. Застосунок зосереджується не лише на зборі статистики, а й на створенні комфортних умов для відпочинку.

Однією з головних особливостей BetterSleep є велика бібліотека аудіоматеріалів [10]. Користувач може обирати звуки природи, потріскування дерева, пташиний спів, шум дощу, морських хвиль або спеціально створені розслаблюючі мелодії. Такий підхід дозволяє створити індивідуальну атмосферу для засинання та зменшити вплив зовнішніх подразників.

Крім аудіоматеріалів, застосунок пропонує медитації, дихальні вправи та історії для сну, які допомагають користувачам розслабитися перед відпочинком. Для формування персоналізованих рекомендацій система враховує вік користувача, його звички та попередню активність у застосунку. Також передбачено функції відстеження сну та аналізу його тривалості, що дозволяє оцінювати ефективність обраних методів покращення відпочинку.

Завдяки поєднанню інструментів моніторингу сну та засобів релаксації BetterSleep виступає комплексним рішенням для користувачів, які прагнуть покращити якість свого відпочинку та сформувати здорові звички сну. Інтерфейс програми продемонстровано на рисунку 1.9.

Рисунок 1.9 – Початкове меню застосунка BetterSleep

Крім звукового супроводу, застосунок містить медитації, дихальні вправи та різноманітні програми для релаксації. Вони допомагають знизити рівень стресу, заспокоїти нервову систему та підготувати організм до нічного відпочинку. Саме тому, BetterSleep часто використовується людьми, які мають труднощі із засинанням або відчувають підвищене емоційне навантаження протягом дня.

Також, варто виділити сучасний дизайн застосунку та зручний користувацький інтерфейс. Усі основні функції розташовані у зрозумілій формі, що дозволяє швидко знаходити необхідні налаштування та використовувати можливості системи без додаткового навчання. Слід зайважити, що застосунок

пропонує персоналізовані рекомендації, які формуються відповідно до вподобань користувача та його історії використання.

У системи також присутні певні обмеження. Основний акцент BetterSleep зроблено на релаксації та допомозі під час засинання, тому можливості статистичного аналізу сну є менш розвиненими порівняно зі спеціалізованими системами моніторингу. Крім того, значна частина медитацій, аудіоматеріалів та додаткових програм доступна лише у платній версії застосунку. Ще одним обмеженням є відсутність повноцінного вебінтерфейсу, через що користувач може працювати із системою лише за допомогою мобільного пристрою.

Отже, BetterSleep є ефективним інструментом для покращення процесу засинання та формування здорових звичок відпочинку. Варто враховувати, що застосунок більше орієнтований на релаксацію користувача, ніж на детальний аналіз параметрів сну, що відрізняє його від інших рішень, розглянутих у даній роботі.

1.3.3 Функціональні можливості SleepScore

SleepScore є сучасним застосунком для моніторингу сну, який поєднує збір даних із подальшим аналізом та оцінюванням якості відпочинку користувача. Основною особливістю системи є використання показника Sleep Score, який формується на основі тривалості сну, часу засинання, регулярності режиму відпочинку та інших параметрів [11]. Завдяки цьому користувач отримує не лише статистичні дані, а й узагальнену оцінку якості свого сну.

Важливою перевагою застосунку є можливість автоматичного збору інформації про сон без необхідності постійного ручного введення даних. На основі отриманої інформації система формує детальні звіти та надає рекомендації щодо покращення режиму відпочинку. Користувач може переглядати історію своїх показників, відстежувати зміни якості сну та аналізувати фактори, які можуть впливати на самопочуття.

Аналітична складова системи у застосунку також не відстає. SleepScore відображає результати моніторингу у вигляді графіків, діаграм та статистичних

показників, що дозволяє легко оцінювати власний прогрес протягом тривалого періоду часу. Такий підхід, допомагає користувачам краще розуміти особливості свого режиму сну та своєчасно виявляти можливі проблеми. Інтерфейс додатку зображений на рисунку 1.10.

Рисунок 1.10 – Вікно застосунку SleepScore

Окрім переваг, застосунок має певні обмеження. Частина функціональних можливостей доступна лише у платній версії, що може обмежувати користувачів у використанні повного набору інструментів аналізу. Також, система орієнтована переважно на мобільні пристрої, тому можливості роботи через веббраузер є обмеженими. Окремі функції та сервіси можуть бути недоступними залежно від країни використання, що зменшує універсальність програмного продукту.

Отже, SleepScore є потужним інструментом для оцінювання якості сну та аналізу режиму відпочинку. Завдяки використанню автоматизованого збору даних і розвинених аналітичних можливостей застосунок надає користувачам корисну інформацію щодо стану їхнього сну. Проте є залежність від платної підписки та орієнтація переважно на мобільні платформи.

1.3.4 Аналіз застосунку Sleep Monitor

Sleep Monitor є мобільним застосунком для відстеження сну, який поєднує функції моніторингу, ведення журналу сну та аналізу статистичних даних. Основною метою системи є допомога користувачам у зборі та збереженні інформації про власний режим відпочинку, що дозволяє краще розуміти особливості сну та відстежувати його зміни з часом.

Однією з найбільш корисних функцій застосунку є можливість запису звуків під час сну. Система може фіксувати хропіння, розмови уві сні та інші звуки, які виникають протягом ночі. Це дозволяє користувачеві отримати додаткову інформацію про якість свого відпочинку та виявити фактори, що можуть негативно впливати на сон. Однак, це за умови платної підписки [12].

Важливе місце у функціоналі Sleep Monitor займає журнал сну. Користувач може переглядати історію власних записів, аналізувати час засинання та пробудження, а також відстежувати зміни режиму відпочинку протягом тривалого періоду. Для зручності аналізу система формує статистичні звіти та графіки, які наочно відображають основні показники сну. Основні функціональні можливості застосунку Sleep Monitor зображені на рисунку 1.11.

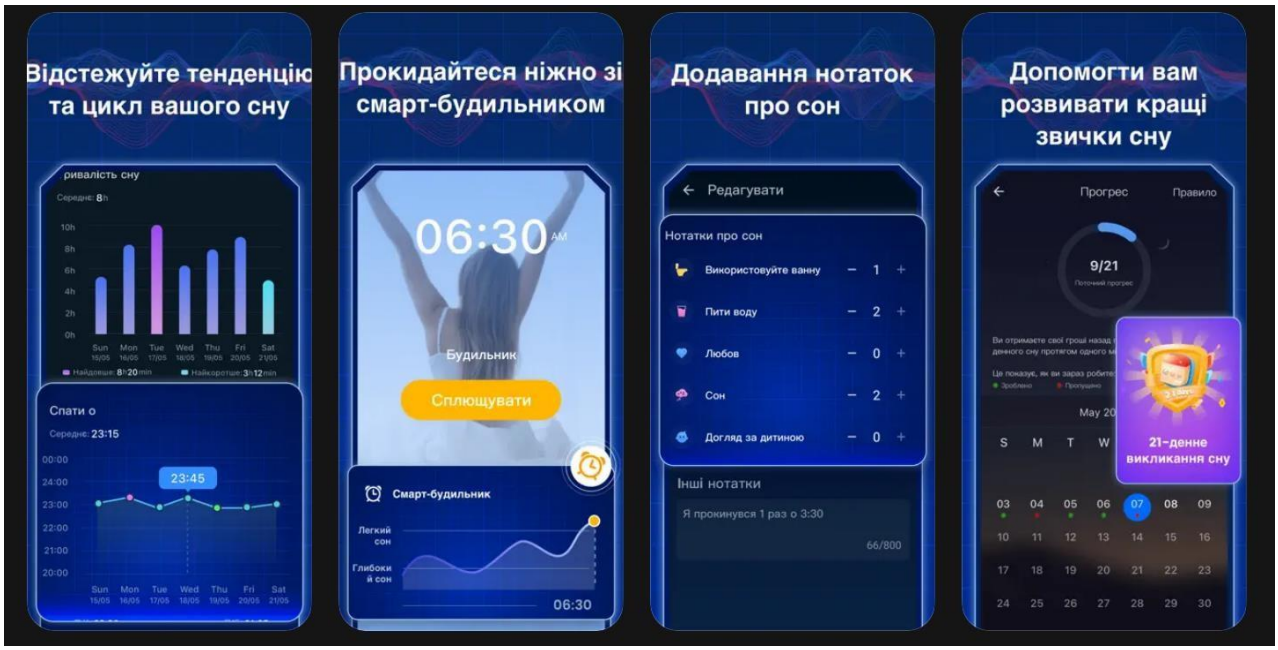

Рисунок 1.11 – Інтерфейс та функціональні модулі застосунку Sleep Monitor.

Ще однією перевагою застосунку є простий та зрозумілий інтерфейс. Більшість функцій доступні без складних налаштувань, що робить систему зручною для широкого кола користувачів незалежно від їхнього досвіду роботи з подібними програмами.

Одним із найбільш помітних є значна кількість рекламних матеріалів у безкоштовній версії застосунку, що може негативно впливати на комфорт використання. Крім того, частина додаткових функцій доступна лише після оформлення платної підписки. Також, система не має вебсайту.

Отже, Sleep Monitor є зручним інструментом для ведення журналу сну та аналізу основних показників відпочинку. Застосунок надає корисні можливості для відстеження змін режиму сну та формування статистичних звітів, проте наявність реклами, обмеження безкоштовної версії та відсутність вебінтерфейсу, знижують його функціональність порівняно з іншими сучасними рішеннями.

1.3.5 Визначення переваг та недоліків аналогів для формування концепції власного рішення

Для узагальнення результатів проведеного аналізу виконано порівняння розглянутих систем за основними функціональними характеристиками. Під час оцінювання враховувалися найбільш поширені функції сучасних систем моніторингу сну, зокрема: можливість ведення щоденника сну, наявність статистичних звітів, інструментів планування режиму відпочинку, засобів релаксації, а також доступність вебверсії та підтримка роботи через браузер.

Обрані критерії дозволяють оцінити не лише функціональні можливості програмних продуктів, а й зручність їх використання для кінцевого користувача, рівень доступності основних функцій та можливість щоденного застосування для контролю режиму сну [13]. Крім того, проведене порівняння дає змогу визначити сильні та слабкі сторони існуючих рішень, виявити найбільш ефективні підходи до організації користувацького інтерфейсу та роботи з даними, а також сформувати перелік функціональних можливостей,

які доцільно реалізувати у власному програмному продукті. Особливу увагу приділено можливостям, які можуть бути реалізовані у вебсайті «SleepHelper» та відрізнятимуть його від існуючих аналогів. Результати порівняння досліджуваних систем наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння досліджуваних систем

Функція	Sleep Cycle	BetterSleep	SleepScore	SleepMonitor
Реєстрація користувачів	✓	✓	✓	✓
Робота через браузер	✗	✗	✗	✗
Щоденник сну	✓	✓	✓	✓
Статистика сну	✓	частково	✓	✓
Калькулятор сну	✗	✗	✗	✗
Розумний будильник	✓	✗	✓	✗
Медитації та релаксація	✗	✓	✗	✗

На основі результатів, наведених у таблиці 1.1, можна зробити висновок, що більшість сучасних рішень для моніторингу сну мають схожий базовий функціонал, який включає ведення журналу сну, збір статистичних даних та аналіз основних показників відпочинку. Незважаючи на аналіз, кожен із розглянутих застосунків орієнтований на вирішення окремих завдань.

1.4 Функціональні та нефункціональні вимоги до вебсайту

Після аналізу існуючих систем моніторингу сну визначено основні функціональні можливості, які є найбільш корисними для користувачів та потрібними для реалізації у вебсайті «SleepHelper». Формування вимог

здійснювалося на основі дослідження функціоналу аналогів, особливостей предметної області та потреб потенційних користувачів.

В інформаційних технологіях слід поділяти вимоги на функціональні та нефункціональні. Функціональні вимоги визначають, які дії повинна виконувати система та які можливості надавати користувачу. Нефункціональні вимоги описують якість роботи системи, її безпеку, продуктивність, надійність та зручність використання [14].

До функціональних вимог вебсайту «SleepHelper» належать можливості, які безпосередньо використовуються користувачем під час роботи із системою. Отже, основними функціональними вимогами є:

- реєстрація та авторизація користувачів;
- ведення особистого щоденника сну;
- додавання, редагування та видалення записів про сон;
- перегляд статистики тривалості сну;
- побудова графіків та діаграм на основі накопичених даних;
- використання калькулятора сну для визначення оптимального часу засинання та пробудження;
- перегляд рекомендацій щодо покращення якості сну;
- музичний супровід для заспокоєння нервової системи;
- збереження історії записів користувача;
- адаптивне відображення інтерфейсу на різних пристроях.

Перераховані функції сформовані на основі аналізу існуючих систем Sleep Cycle, BetterSleep, SleepScore та Sleep Monitor, а також з урахуванням особливостей веборієнтованої реалізації системи.

Окрім функціональних можливостей, важливе значення мають нефункціональні вимоги, які визначають якість роботи програмного продукту. Вони впливають на продуктивність, безпеку та зручність використання системи. Для вебсайту «SleepHelper» визначено такі нефункціональні вимоги:

- зрозумілий та простий інтерфейс користувача;

- підтримка сучасних веббраузерів;
- захист персональних даних користувачів;
- безпечне зберігання паролів;
- стабільна робота системи при одночасному використанні декількома користувачами;
- швидке завантаження сторінок;
- можливість подальшого розширення функціоналу;
- надійне зберігання даних у базі даних.

Дотримання зазначених вимог дозволить створити вебсайт, який буде не лише функціональним, але й зручним для використання, безпечним та готовим до подальшого розвитку.

1.5 Постановка задачі на розробку системи

Вебсайт «SleepHelper» являє собою клієнт-серверну інформаційну систему, призначену для збору, збереження, обробки та візуалізації даних про сон користувачів. Клієнтська частина системи забезпечує взаємодію користувача з функціональними модулями через вебінтерфейс, тоді як серверна частина відповідає за обробку запитів, керування даними та взаємодію з базою даних.

Основними функціональними компонентами системи є модуль реєстрації та авторизації користувачів, інтерактивний щоденник сну, калькулятор часу засинання та пробудження, модуль статистики й аналітики, а також розділ рекомендацій та релаксації. Кожен із модулів виконує окремі функції, проте всі вони працюють у межах єдиного програмного середовища та використовують спільну базу даних.

Результатом виконання кваліфікаційної роботи є веборієнтована інформаційна система «SleepHelper», яка забезпечує користувачам можливість ведення щоденника сну, аналізу статистичних показників, візуалізації даних та отримання рекомендацій щодо покращення якості відпочинку. Розроблена

система може використовуватися як самостійний програмний продукт та виступати основою для подальшого розширення функціональних можливостей.

1.6 Висновок до першого розділу

В першому розділі кваліфікаційної роботи проведено аналіз предметної області моніторингу якості сну та досліджено актуальність створення сучасних інформаційних систем для контролю режиму відпочинку. Розглянуто особливості сну людини, його основні стадії та фактори, що впливають на якість відпочинку. Встановлено, що недостатня тривалість або порушення структури сну можуть негативно впливати на фізичний та психоемоційний стан людини, тому використання цифрових засобів моніторингу є актуальним напрямом розвитку інформаційних технологій.

У процесі дослідження проаналізовано переваги та недоліки веборієнтованого підходу до реалізації системи, а також визначено основні ризики, які можуть виникати під час розробки та експлуатації вебсайту «SleepHelper». Проведений аналіз існуючих рішень, зокрема Sleep Cycle, BetterSleep, SleepScore та Sleep Monitor, дозволив визначити найбільш затребувані функціональні можливості та врахувати їх під час проєктування власної системи.

На основі виконаного аналізу сформовано функціональні та нефункціональні вимоги до вебсайту «SleepHelper», а також поставлено основні завдання його розробки. Отримані результати створюють необхідну теоретичну основу для подальшого проєктування архітектури системи, вибору технологій реалізації та розробки програмного продукту з використанням React та Python FastAPI.

РОЗДІЛ 2. ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНОГО СТЕКУ ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Вибір та обґрунтування технологій для фронтенду

Frontend – це публічна частина web-додатків (вебсайтів), з якою користувач може взаємодіяти і контактувати напряму. У Фронтенд входить відображення функціональних завдань призначеного для користувача інтерфейсу, що виконуються на стороні клієнта, а також обробка запитів користувачів. Отже, фронтенд – це все те, що бачить користувач при відкритті web-сторінки [15].

Клієнтська частина вебсайту «SleepHelper» відповідає за взаємодію користувача із системою, відображення даних, побудову графіків, роботу форм введення та навігацію між сторінками.

Під час вибору технологій для реалізації клієнтської частини особливу увагу було приділено комфортності, продуктивності, масштабованості, можливості повторного використання компонентів та сумісності із сучасними вебстандартами. Також, важливим фактором була підтримка активною спільнотою розробників, наявність готових бібліотек для побудови інтерфейсів та можливість інтеграції із серверною частиною на базі FastAPI.

Для реалізації фронтенду вебсайту «SleepHelper» було обрано бібліотеку React у поєднанні з мовою TypeScript. Таке рішення прийнято після аналізу найбільш популярних сучасних інструментів розробки вебінтерфейсів, серед яких React, Vue.js, Angular та Svelte.

2.1.1 Порівняння React з іншими сучасними UI-фреймворками

React є однією з найпопулярніших бібліотек для створення користувацьких інтерфейсів вебзастосунків. Вона була розроблена компанією Meta (Facebook) у 2013 році та використовується для створення динамічних односторінкових застосунків (SPA). Основними особливостями React є

компонентний підхід до розробки інтерфейсів, використання Virtual DOM та можливість повторного використання окремих елементів інтерфейсу [16].

React використовує декларативний підхід до побудови інтерфейсів. Розробник описує кінцевий стан елементів інтерфейсу, а бібліотека самостійно визначає, які зміни необхідно внести до сторінки. Це спрощує підтримку проєкту та зменшує кількість помилок при розробці складних вебзастосунків.

Одним із ключових механізмів React є Virtual DOM. На відміну від традиційного оновлення елементів сторінки, React спочатку виконує зміни у віртуальному представленні документа, після чого оновлює лише ті елементи реального DOM, які фактично змінилися. Такий підхід дозволяє підвищити швидкодію застосунку та зменшити навантаження на браузер. На рисунку 2.1 представлено переваги використання React.

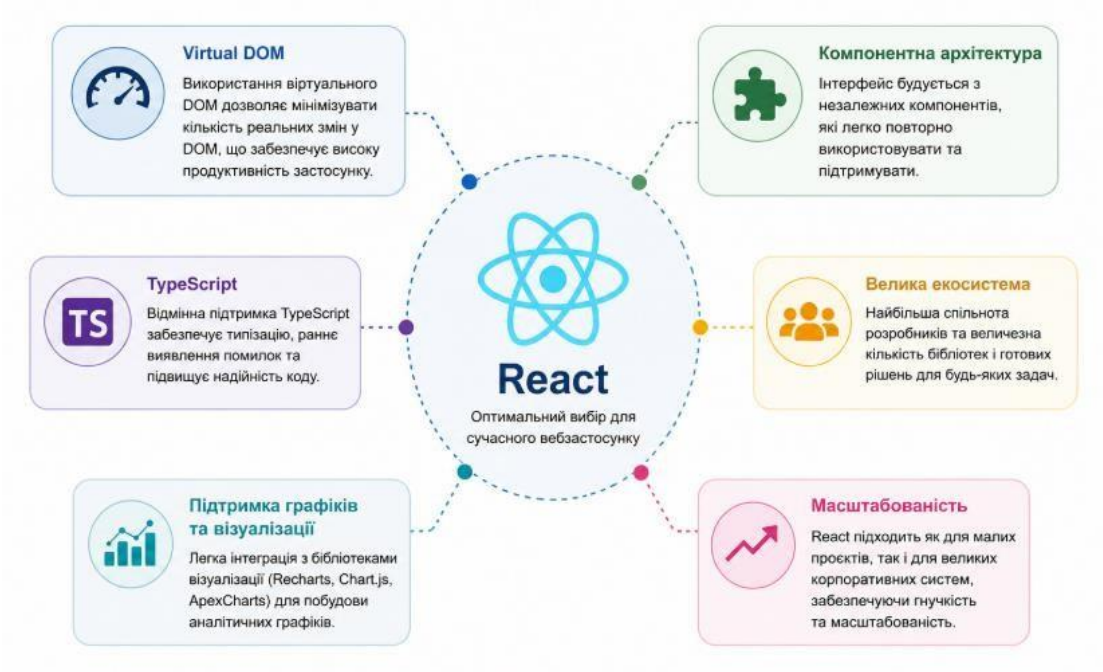

Рисунок 2.1 – Переваги React для реалізації вебсайту

Під час вибору технології для розробки вебсайту «SleepHelper» було проведено порівняння React із сучасними альтернативами, серед яких Vue.js, Angular та Svelte.

Vue.js є прогресивним JavaScript-фреймворком із простим синтаксисом та невисоким порогом входження. Він добре підходить для невеликих та середніх проєктів, але має меншу екосистему порівняно з React [17].

Незважаючи на це, Vue.js є одним із найближчих конкурентів React. Розробники можуть швидко створювати інтерфейси без необхідності глибокого вивчення додаткових концепцій. Vue також підтримує компонентний підхід та роботу з Virtual DOM.

Екосистема Vue є меншою порівняно з React. Кількість готових бібліотек, навчальних матеріалів та прикладів реалізації складних систем поступається рішенням екосистеми React. Саме тому для довгострокових проєктів із перспективою розвитку частіше обирають React.

Angular є повноцінним фреймворком для розробки вебзастосунків, який підтримується компанією Google. На відміну від React, що використовується переважно для побудови користувацького інтерфейсу, Angular надає розробнику готовий набір інструментів для створення повноцінної клієнтської частини застосунку. До складу фреймворку входять засоби маршрутизації, роботи з формами, HTTP-запитами, керування залежностями та організації архітектури проєкту [18].

Однією з головних переваг Angular є його придатність для розробки великих корпоративних систем, де важливими є стандартизація коду та чітка структура проєкту. Використання TypeScript за замовчуванням також сприяє підвищенню надійності та підтримуваності програмного забезпечення.

Проте, Angular має досить високий поріг входження. Для ефективної роботи з фреймворком розробнику необхідно опанувати значну кількість вбудованих механізмів і концепцій. Крім того, складніша архітектура може збільшувати тривалість розробки та підтримки проєкту. Вебсайт «SleepHelper» належить до середніх за складністю вебзастосунків, використання Angular може бути надмірним і призвести до зайвого ускладнення структури системи.

Svelte використовує принцип компіляції компонентів у чистий JavaScript ще на етапі збірки проєкту. Завдяки цьому створюються компактні застосунки з

високою продуктивністю. Однак екосистема Svelte є менш розвиненою, а кількість готових бібліотек та прикладів значно поступається React [19].

Svelte використовує принципово інший підхід до розробки інтерфейсів. На відміну від React та Vue, які виконують частину логіки у браузері під час роботи застосунку, Svelte переносить значну частину обробки на етап компіляції. Це дозволяє отримувати компактніші та швидші застосунки.

Результати порівняння основних характеристик сучасних фронтенд-рішень наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння фронтенд-фреймворків

Характеристика	React	Vue.js	Angular	Svelte
Тип рішення	Бібліотека	Прогресивний фреймворк	Повний фреймворк	Компілятор
Virtual DOM	✓	✓	✓	✗(компіляція)
TypeScript Підтримка	Відмінна	Добра	Вбудована	Добра
Крива навчання	Середня	Низька	Висока	Низька
Екосистема	Дуже велика	Велика	Велика	Середня
Повторне використання компонентів	✓	✓	✓	✓
Популярність серед розробників	~40 млн	~15 млн	~9 млн	~2 млн

На основі проведеного аналізу було прийнято рішення використовувати React як основну технологію фронтенду вебсайту «SleepHelper». Через велику кількістю готових бібліотек, високу продуктивність та зручним компонентним підходом до побудови інтерфейсів.

2.1.2 Переваги використання TypeScript для надійності інтерфейсу

TypeScript – це мова програмування, розроблена компанією Microsoft, яка являє собою надмножину JavaScript. Основною метою TypeScript є покращення розробки великих додатків на JavaScript шляхом додавання суворої типізації та інших можливостей, які відсутні в стандартному JavaScript. Код на TypeScript компілюється в чистий JavaScript, який може бути виконаний у будь-якому браузері або середовищі, що підтримує JavaScript [20].

У сучасній веброзробці TypeScript активно використовується разом із React для створення великих та середніх вебзастосунків. Використання типізації дозволяє виявляти значну кількість помилок ще під час розробки, до запуску програми. Завдяки цьому підвищується надійність програмного забезпечення та зменшуються витрати часу на налагодження.

Однією з ключових переваг TypeScript є контроль типів даних. Розробник може чітко визначити, які значення можуть передаватися до функцій або компонентів. Якщо в коді буде використано неправильний тип даних, середовище розробки одразу повідомить про помилку. Такий підхід значно знижує ймовірність виникнення помилок під час виконання програми.

Важливою перевагою є також підтримка сучасних інструментів розробки. TypeScript забезпечує автоматичне доповнення коду, підказки щодо властивостей об'єктів, швидку навігацію між компонентами та рефакторинг без ризику пошкодження структури проєкту. Це корисно для вебсайту «SleepHelper», який містить велику кількість форм, сторінок та модулів обробки даних.

Крім того, TypeScript значно покращує підтримуваність програмного коду. У процесі подальшого розвитку проєкту розробник може швидше знаходити помилки, змінювати структуру компонентів та додавати новий функціонал без ризику порушення вже реалізованих можливостей системи. На рисунку 2.2 зображено надійність інтерфейсу.

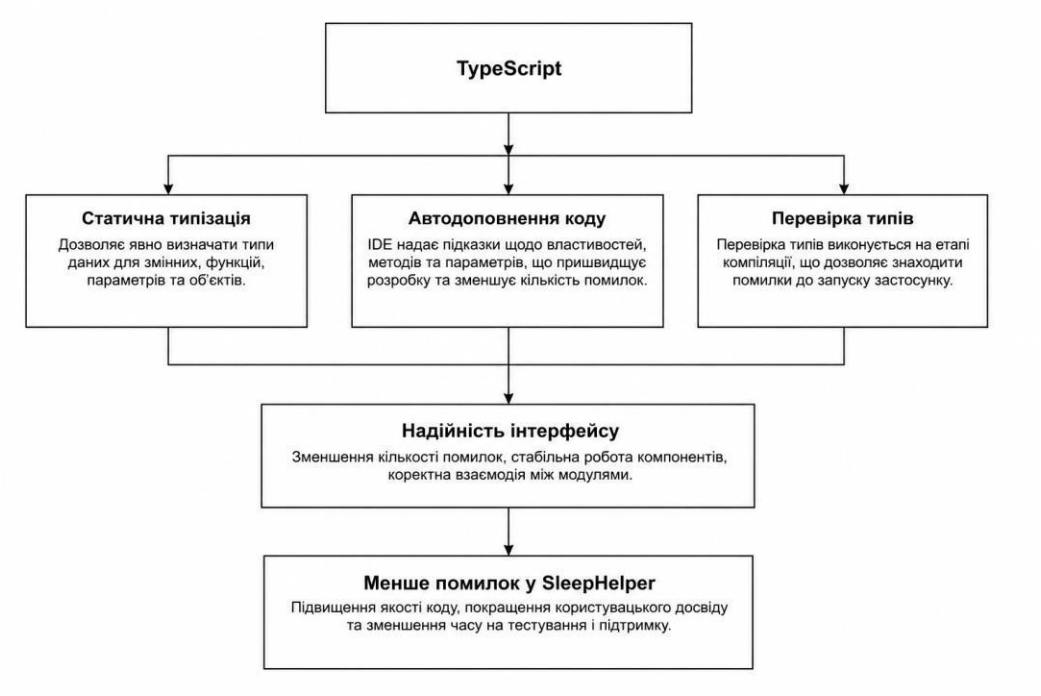

Рисунок 2.2 – Переваги використання TypeScript

Таким чином, використання TypeScript у поєднанні з React забезпечує підвищення якості програмного коду, спрощує процес розробки та підтримки вебзастосунку, а також сприяє створенню більш надійного та масштабованого програмного продукту.

2.2 Вибір технологій для бекенду та бази даних

Бекенд – це ядро, або “мозок” будь-якого застосунку. Він відповідає за зберігання, обробку та організацію даних, реалізацію бізнес-логіки та забезпечення безпеки. Серверна частина слугує посередником між користувацьким інтерфейсом (фронтендом) та базами даних, API та іншими сервісами, гарантуючи стабільну та надійну роботу застосунку [21].

Під час вибору технологій для реалізації серверної частини розробний орієнтується на: продуктивності, масштабованості, підтримці асинхронної обробки запитів, простоті інтеграції з React та можливостям роботи з сучасними базами даних. Крім того, важливим критерієм є наявність якісної документації.

Після аналізу сучасних серверних технологій для реалізації вебсайту «SleepHelper» було обрано фреймворк FastAPI та систему керування базами даних MongoDB. Таке поєднання забезпечує високу швидкодію, зручну розробку REST API та ефективну роботу з даними користувачів.

2.2.1 Переваги фреймворку FastAPI для розробки асинхронних вебсервісів

FastAPI є сучасним вебфреймворком для розробки серверних застосунків мовою Python. Проєкт був створений Себастьяном Раміресом у 2018 році та швидко набув популярності завдяки високій продуктивності, підтримці асинхронного програмування та автоматичній генерації документації API [22].

Однією з ключових особливостей FastAPI є використання сучасних можливостей Python, зокрема анотацій типів (Type Hints). Завдяки цьому фреймворк автоматично виконує перевірку даних, формує документацію та спрощує розробку серверної логіки. Такий підхід дозволяє зменшити кількість помилок та підвищити якість програмного коду.

Для вебсайту «SleepHelper» потрібно забезпечити швидку обробку великої кількості запитів, пов'язаних із реєстрацією користувачів, збереженням записів про сон, формуванням статистики та отриманням рекомендацій. Саме тому критерієм вибору стала підтримка асинхронного програмування.

Асинхронна модель роботи дозволяє серверу обробляти декілька запитів одночасно без блокування основного потоку виконання. Це потрібно під час звернення до бази даних або виконання операцій введення-виведення. У результаті зменшується час очікування користувача та підвищується загальна продуктивність системи.

Серед основних переваг FastAPI можна виділити високу швидкодію. Згідно з результатами незалежних тестувань, FastAPI належить до найшвидших Python-фреймворків і поступається лише низькорівневим рішенням на кшталт Node.js або Go. Це дозволяє використовувати його для створення сучасних вебсервісів із великою кількістю одночасних користувачів.

Ще однією важливою перевагою є автоматичне створення інтерактивної документації API за допомогою Swagger UI та ReDoc. Після запуску сервера розробник отримує готовий вебінтерфейс для тестування всіх маршрутів API, що значно спрощує налагодження та інтеграцію клієнтської частини з сервером.

FastAPI також добре інтегрується із сучасними базами даних, механізмами аутентифікації та бібліотеками обробки даних. Це дозволяє реалізувати безпечну систему авторизації користувачів, зберігання інформації про сон та формування аналітичних звітів без використання великої кількості додаткового програмного забезпечення.

Під час вибору серверної технології для вебсайту «SleepHelper» також були розглянуті популярні Python-фреймворки Django та Flask.

Django — це високорівневий вебфреймворк з відкритим вихідним кодом, написаний мовою Python, який сприяє швидкій розробці безпечних та масштабованих вебзастосунків [23]. Він реалізує концепцію «batteries included», що передбачає наявність великої кількості вбудованих інструментів для роботи з базами даних, системою аутентифікації користувачів, адміністративною панеллю та механізмами безпеки. Завдяки цьому Django добре підходить для створення великих інформаційних систем та корпоративних вебпроєктів.

Іншим популярним рішенням є Flask. Це легкий Python-фреймворк для створення вебзастосунків, який надає базовий набір інструментів та дозволяє розробнику самостійно обирати необхідні компоненти системи [24].

Однак мінімалістичний підхід Flask має і певні недоліки. Для реалізації повноцінного REST API, аутентифікації користувачів, валідації даних та автоматичної документації необхідно підключати додаткові розширення. У результаті зі збільшенням складності проєкту зростає кількість залежностей та обсяг конфігураційного коду.

Порівняння FastAPI з іншими популярними Python-фреймворками також, наведено в таблиці 2.2.

Таблиця 2.2 – Порівняння Python-фреймворків для API

Характеристика	FastAPI	Django	Flask
Асинхронність	Вбудована (async/await)	Обмежена	Обмежена
Швидкодія	Висока	Середня	Висока
Автоматична документація API	✓	✗(частково)	✗
Простота створення REST API	Висока	Середня	Висока
Типізація Python	Повна	Часткова	Часткова
Масштабованість	Висока	Висока	Середня
Розмір проєкту	Малі та великі	Великі	Малі та середні

На основі проведеного аналізу прийнято рішення використовувати FastAPI як основу серверної частини вебсайту «SleepHelper». Фреймворк поєднує високу продуктивність, підтримку асинхронного програмування та сучасні засоби розробки API. Крім того, використання FastAPI дозволяє ефективно взаємодіяти з клієнтською частиною, реалізованою на React, та забезпечувати швидкий обмін даними між компонентами системи.

2.2.2 Вибір NoSQL СУБД MongoDB та використання асинхронного драйвера Motor

MongoDB є документоорієнтованою NoSQL системою керування базами даних, яка зберігає інформацію у вигляді документів формату BSON (Binary JSON). На відміну від реляційних баз даних, де дані розміщуються у таблицях із фіксованою структурою, MongoDB дозволяє використовувати гнучкі схеми документів. Це дає можливість змінювати структуру даних без складних операцій міграції та перебудови бази даних [25].

Під час проєктування вебсайту «SleepHelper» було враховано, що різні користувачі можуть зберігати різну кількість записів про сон, додаткові примітки, статистичні показники та іншу інформацію. Використання документоорієнтованої бази даних дозволяє зручно зберігати такі дані у вигляді окремих документів без необхідності створення великої кількості взаємопов'язаних таблиць.

Серед альтернативних рішень розглядалися PostgreSQL та MySQL. Дані системи керування базами даних широко використовуються для створення вебзастосунків та забезпечують високий рівень надійності. Проте їх використання передбачає проєктування жорсткої структури таблиць та зв'язків між ними. Для системи моніторингу сну, де структура даних може змінюватися під час розвитку проєкту, більш доцільним є використання NoSQL підходу.

Однією з головних переваг MongoDB є висока швидкість роботи з документами, можливість горизонтального масштабування та простота інтеграції із сучасними вебфреймворками. Крім того, формат JSON-подібних документів добре поєднується з даними, які передаються між React та FastAPI через REST API.

Для забезпечення асинхронної роботи з MongoDB у проєкті використовується драйвер Motor. Він є офіційним асинхронним драйвером для Python, побудованим на основі PyMongo та оптимізованим для роботи з асинхронними фреймворками [26].

Використання Motor дозволяє виконувати операції читання та запису даних без блокування основного потоку виконання сервера. Завдяки цьому FastAPI може одночасно обслуговувати велику кількість запитів користувачів, що позитивно впливає на продуктивність системи.

У вебсайті «SleepHelper» драйвер Motor використовується для роботи з обліковими записами користувачів, записами щоденника сну, статистичними даними та іншою інформацією, яка зберігається у базі даних. Поєднання FastAPI, MongoDB та Motor дозволяє створити сучасну серверну архітектуру з високою швидкодією та можливістю подальшого масштабування.

Використання MongoDB та асинхронного драйвера Motor є доцільним рішенням для реалізації серверної частини вебсайту «SleepHelper», оскільки забезпечує гнучкість структури даних, високу продуктивність та ефективну інтеграцію з технологіями, обраними для розробки проєкту.

Під час вибору системи керування базами даних для вебсайту «SleepHelper» було розглянуто декілька популярних рішень, серед яких MongoDB, PostgreSQL та MySQL.

PostgreSQL є об'єктно-реляційною системою керування базами даних з відкритим вихідним кодом. Вона підтримує складні SQL-запити, транзакції, механізми забезпечення цілісності даних та високий рівень надійності. PostgreSQL широко використовується у корпоративних системах, де важливими є складні зв'язки між даними та підтримка великих обсягів інформації. Крім того, сучасні версії PostgreSQL підтримують роботу з JSON-документами, що частково наближає її можливості до NoSQL-рішень [27].

MySQL є однією з найпоширеніших реляційних систем керування базами даних. Основними перевагами MySQL є простота використання, висока швидкість виконання запитів та велика кількість інструментів для адміністрування. Дана СУБД часто використовується для створення вебзастосунків різної складності та добре інтегрується з більшістю сучасних серверних технологій [28].

Разом із тим реляційні бази даних передбачають використання таблиць із заздалегідь визначеною структурою та зв'язками між ними. У процесі розвитку проєкту зміна структури даних може потребувати додаткових міграцій та модифікації схеми бази даних. Для системи моніторингу сну, де набір збережених даних може змінюватися залежно від подальшого розвитку функціоналу, більш доцільним є використання документоорієнтованого підходу.

MongoDB належить до класу NoSQL систем керування базами даних та використовує модель зберігання інформації у вигляді документів BSON. Такий підхід дозволяє створювати гнучкі структури даних без жорсткої прив'язки до

схеми таблиць. Для вебсайту «SleepHelper» це є важливою перевагою, оскільки записи щоденника сну, статистичні дані та профілі користувачів можуть містити різний набір атрибутів.

Порівняння розглянутих систем керування базами даних наведено в таблиці 2.3.

Таблиця 2.3 – Основні характеристики MongoDB, PostgreSQL та MySQL

Характеристика	MongoDB	PostgreSQL	MySQL
Тип БД	NoSQL(документна)	Реляційна	Реляційна
Формат даних	Документи BSON	Таблиці	Таблиці
Гнучкість структури	Висока	Середня	Середня
Масштабування	Горизонтальне	Вертикальне	Обмежене
Підтримка JSON	Вбудована	Часткова	Часткова
Інтеграція з FastAPI	Відмінна	Добра	Добра

Зрозуміло, що PostgreSQL та MySQL є потужними реляційними системами керування базами даних, які добре підходять для проєктів із чітко визначеною структурою даних. Проте для вебсайту «SleepHelper» більш доцільним є використання MongoDB, оскільки вона забезпечує гнучкість структури документів, просту інтеграцію з JSON-форматом даних та зручну роботу з асинхронними технологіями.

Додатковою перевагою MongoDB є ефективна інтеграція з драйвером Motor та фреймворком FastAPI. Це дозволяє реалізувати високопродуктивну серверну частину застосунку та забезпечити швидкий доступ до даних користувачів навіть при збільшенні навантаження на систему.

Завдяки зазначеним перевагам MongoDB, Motor та FastAPI утворюють ефективний технологічний стек для розробки сучасних вебзастосунків, що

потребують швидкої обробки даних, гнучкої структури зберігання інформації та високої продуктивності серверної частини.

2.3 Проєктування структури бази даних

Для зберігання даних вебсайту «SleepHelper» використовується документоорієнтована база даних MongoDB [29]. Такий підхід є зручним для системи моніторингу сну, оскільки записи користувачів мають природну структуру документа: дата, час засинання, час пробудження, тривалість сну, оцінка якості та нотатки.

У межах проєкту база даних містить основні колекції, які забезпечують роботу облікових записів користувачів та щоденника сну. Колекція користувачів зберігає дані для авторизації, а колекція записів сну містить інформацію про щоденні записи користувача. Зв’язок між ними реалізується через поле user_id, що дозволяє відокремити дані різних користувачів.

2.3.1 Структура документів для щоденника сну

Основною колекцією для реалізації щоденника сну є sleep_records. Вона призначена для зберігання всіх записів користувача про сон. Кожен документ у цій колекції відповідає одному запису та містить набір полів, необхідних для подальшого аналізу й відображення інформації в інтерфейсі вебсайту.

До основних полів документа належать ідентифікатор запису, ідентифікатор користувача, дата сну, час засинання, час пробудження, тривалість сну, оцінка якості та додаткові нотатки. Наявність поля user_id є важливою, оскільки воно забезпечує прив’язку запису до конкретного користувача та не дозволяє змішувати дані різних облікових записів.

Така структура дозволяє зручно виконувати операції створення, перегляду, редагування та видалення записів. Крім того, збереження часу засинання та пробудження дає можливість формувати статистику тривалості сну, а поле mood використовується для оцінювання якості відпочинку.

Поле notes є необов’язковим і може використовуватися для ведення коротких особистих спостережень. Наприклад, користувач може вказати, що вплинуло на якість сну: пізнє засинання, стрес, фізична активність або використання електронних пристроїв перед сном.

2.3.2 Організація збереження статистики

Статистичні дані у вебсайті «SleepHelper» формуються на основі записів, що зберігаються у колекції sleep_records. Окрема колекція для статистики не є обов’язковою, через те що всі необхідні показники можна обчислювати динамічно під час запиту користувача або під час відкриття сторінки статистики.

До основних статистичних показників належать середня тривалість сну за обраний період, кількість записів, середня оцінка якості сну, розподіл якості сну та зміна тривалості сну за днями. Такий підхід дозволяє не дублювати дані в базі, а використовувати вже наявні записи щоденника. Приклад реалізації функції визначення загальної оцінки якості сну наведено в лістингу 2.1.

Лістинг 2.1 – Реалізація шкали оцінювання якості сну

```
const MOOD_SCORE: Record<string, number> = {
    Excellent: 5,
    Good: 4,
    Average: 3,
    Bad: 2,
    Terrible: 1,
};

const calcOverallQuality = (records: SleepRecord[]): string | null
=> {
    if (records.length === 0) return null;

    const total = records.reduce((sum, r) => sum +
(MOOD_SCORE[r.mood] ?? 3), 0);
    const avg = total / records.length;
    if (avg >= 4.5) return "Excellent";
    if (avg >= 3.5) return "Good";
    if (avg >= 2.5) return "Average";
    if (avg >= 1.5) return "Bad";
    return "Terrible";
};
```

У наведеному фрагменті коду створено об’єкт MOOD_SCORE, який встановлює відповідність між текстовою оцінкою якості сну та числовим значенням. Найвища оцінка Excellent відповідає значенню 5, а найнижча оцінка Terrible – значенню 1. Такий підхід дозволяє перетворити суб’єктивну оцінку користувача у числовий показник, який можна використовувати для подальших розрахунків.

Функція calcOverallQuality приймає масив записів сну типу SleepRecord. Спочатку виконується перевірка на наявність записів: якщо масив порожній, функція повертає значення null. Далі за допомогою методу reduce обчислюється загальна сума балів якості сну. Якщо для певного запису оцінка не визначена або відсутня у словнику MOOD_SCORE, використовується значення за замовчуванням 3, що відповідає середній якості сну.

Після обчислення загальної суми балів визначається середнє значення якості сну за обраний період. Отримане число порівнюється з установленими діапазонами та перетворюється назад у текстову характеристику: Excellent, Good, Average, Bad або Terrible. Завдяки цьому користувач бачить не лише окремі оцінки за кожен день, а й загальний підсумок якості сну за вибраний проміжок часу.

Такий алгоритм є зручним для сторінки статистики вебсайту «SleepHelper», оскільки дозволяє швидко обробляти записи щоденника сну та формувати зрозумілий для користувача результат.

2.4 Проєктування REST API та взаємодії компонентів

Для забезпечення взаємодії між клієнтською та серверною частинами вебсайту «SleepHelper» передбачено використання REST API. REST є архітектурним підходом, який застосовується для організації обміну даними між різними частинами вебсистеми через HTTP-запити [30]. У межах проєкту клієнтська частина, реалізована за допомогою React, надсилає запити до

серверної частини на FastAPI, після чого сервер обробляє отримані дані, звертається до MongoDB та повертає відповідь у форматі JSON.

Основна перевага такого підходу полягає у розділенні відповідальності між компонентами системи. React відповідає за відображення інтерфейсу користувача, FastAPI — за обробку запитів і бізнес-логіку, а MongoDB — за збереження даних. HTTP-методи, такі як GET, POST, PUT і DELETE, використовуються для визначення типу операції, яку потрібно виконати над даними [31].

Загальну схему взаємодії компонентів вебсайту «SleepHelper» наведено на рисунку 2.3.

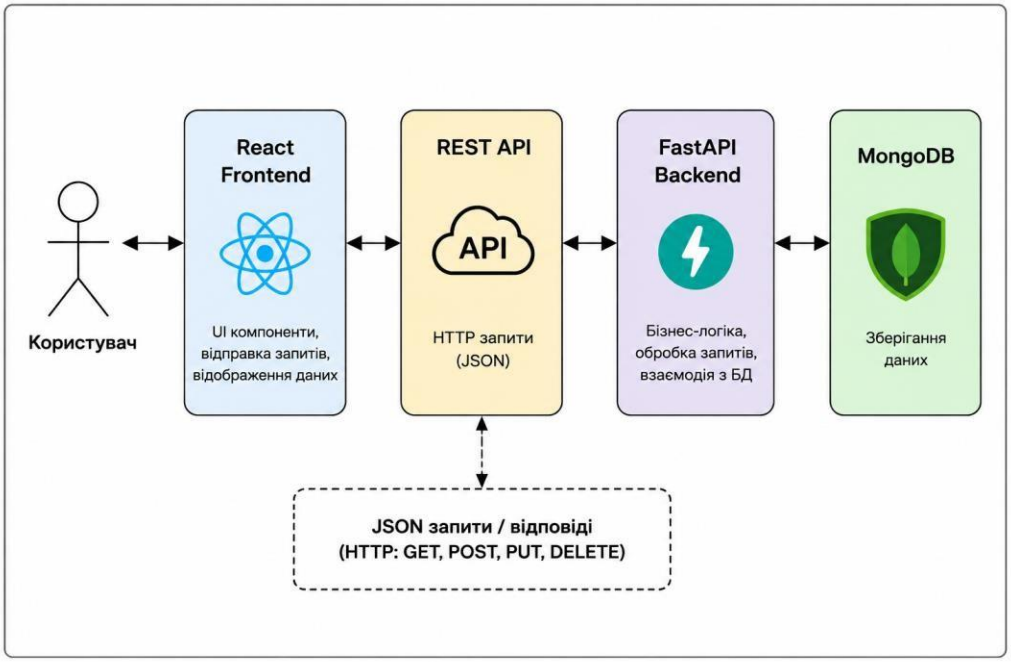

Рисунок 2.3 – Загальна схема взаємодії компонентів вебсайту

Користувач взаємодіє з вебсайтом через клієнтську частину React. Усі запити передаються через REST API до серверної частини FastAPI. Після обробки запиту сервер звертається до бази даних MongoDB, виконує необхідну операцію та повертає результат назад клієнту у форматі JSON.

Отримані дані використовуються React-компонентами для оновлення інтерфейсу користувача без перезавантаження сторінки, що забезпечує швидку

та зручну роботу із системою. Така організація взаємодії дозволяє розділити логіку представлення даних та логіку їх обробки, підвищує масштабованість програмного продукту та спрощує подальший супровід і розвиток вебсайту.

2.4.1 Едпоінти для аутентифікації та облікових записів

Одним із перших етапів взаємодії користувача з вебсайтом є створення облікового запису та вхід до системи. Для цього в REST API передбачається група ендпоінтів, які відповідають за реєстрацію, авторизацію, перегляд профілю та вихід із системи.

Під час реєстрації користувач вводить основні облікові дані, зокрема ім’я, електронну пошту та пароль. Серверна частина повинна перевірити правильність введених даних, створити новий запис користувача в базі даних та забезпечити безпечне зберігання пароля. Після авторизації користувач отримує доступ до особистого кабінету та власних записів щоденника сну.

У FastAPI для створення таких маршрутів використовуються механізми обробки HTTP-запитів, валідації даних та повернення відповідей у форматі JSON [31]. Проєктування модуля аутентифікації зображено на рисунку 2.4.

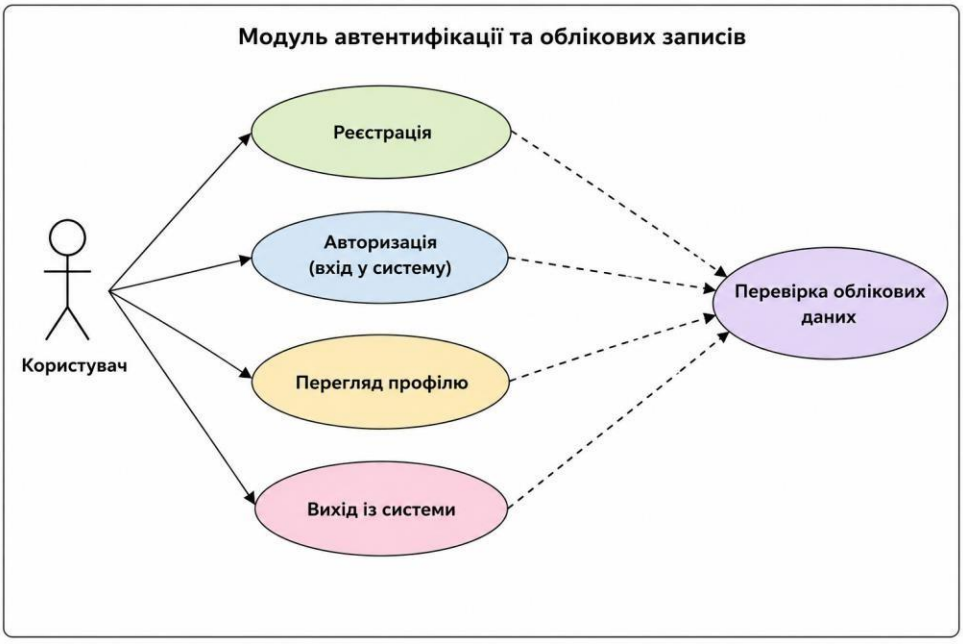

Рисунок 2.4 – Діаграма варіантів використання модуля аутентифікації

На рисунку 2.4 показано, що користувач може виконувати основні дії, пов’язані з обліковим записом: реєстрацію, авторизацію, перегляд профілю та вихід із системи. Усі ці операції пов’язані з перевіркою облікових даних, що дозволяє забезпечити захист персональної інформації та доступ лише до власних даних користувача.

2.4.2 Маршрути API для керування записами щоденника

Основним функціональним модулем вебсайту «SleepHelper» є щоденник сну. Для його роботи необхідно передбачити маршрути API, які забезпечують створення, перегляд, редагування та видалення записів. Такі операції відповідають базовим CRUD-діям, що широко використовуються під час проєктування вебзастосунків.

Кожен запис щоденника сну повинен бути пов’язаний з конкретним користувачем. Це потрібно для того, щоб користувач бачив лише власні дані та не мав доступу до записів інших облікових записів. Під час створення або редагування запису система має працювати з такими даними, як дата, час засинання, час пробудження, тривалість сну, оцінка якості та нотатки.

Оскільки записи зберігаються у MongoDB, вони можуть мати гнучку документну структуру, зручну для збереження даних щоденника сну [32]. Основні операції над записами щоденника сну наведено на рисунку 2.5.

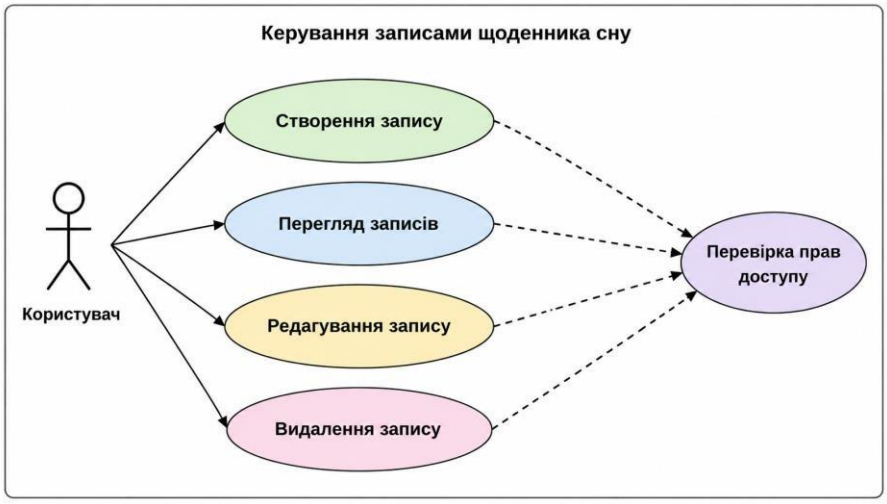

Рисунок 2.5 – Діаграма варіантів керування записами щоденника сну

Користувач може створювати нові записи, переглядати раніше збережені дані, редагувати інформацію та видаляти непотрібні записи. Перед виконанням цих дій система повинна перевірити права доступу користувача, що є важливою умовою для захисту персональних даних.

2.4.3 Запити для отримання аналітичних даних

Окрему групу REST API становлять запити, пов’язані з отриманням аналітичних даних. Вони необхідні для формування сторінки статистики, де користувач може переглядати середню тривалість сну, загальну оцінку якості, розподіл оцінок та динаміку сну за певний період.

Аналітичні дані формуються на основі записів щоденника сну, що зберігаються у базі даних. Клієнтська частина може надсилати запит на отримання записів за певний проміжок часу, після чого система виконує обробку даних і відображає результат у вигляді графіків, діаграм та інформаційних блоків.

Для формування статистики використовуються дані про дату сну, тривалість відпочинку та суб’єктивну оцінку якості сну, яку користувач вказує під час створення запису. На основі цих показників система може визначати середню тривалість сну за обраний період, обчислювати загальну оцінку якості сну та відображати зміни режиму відпочинку в динаміці.

Особливістю запропонованого підходу є те, що аналітичні показники не зберігаються окремо в базі даних, а формуються безпосередньо на основі наявних записів користувача. Це дозволяє уникнути дублювання інформації, зменшити обсяг даних, що зберігаються в системі, та забезпечити актуальність статистики після кожного додавання, редагування або видалення запису.

Отримані результати використовуються для побудови графіків тривалості сну, діаграм розподілу оцінок якості та інформаційних блоків із середніми показниками. Проєктування модуля аналітики показано на рисунку 2.6.

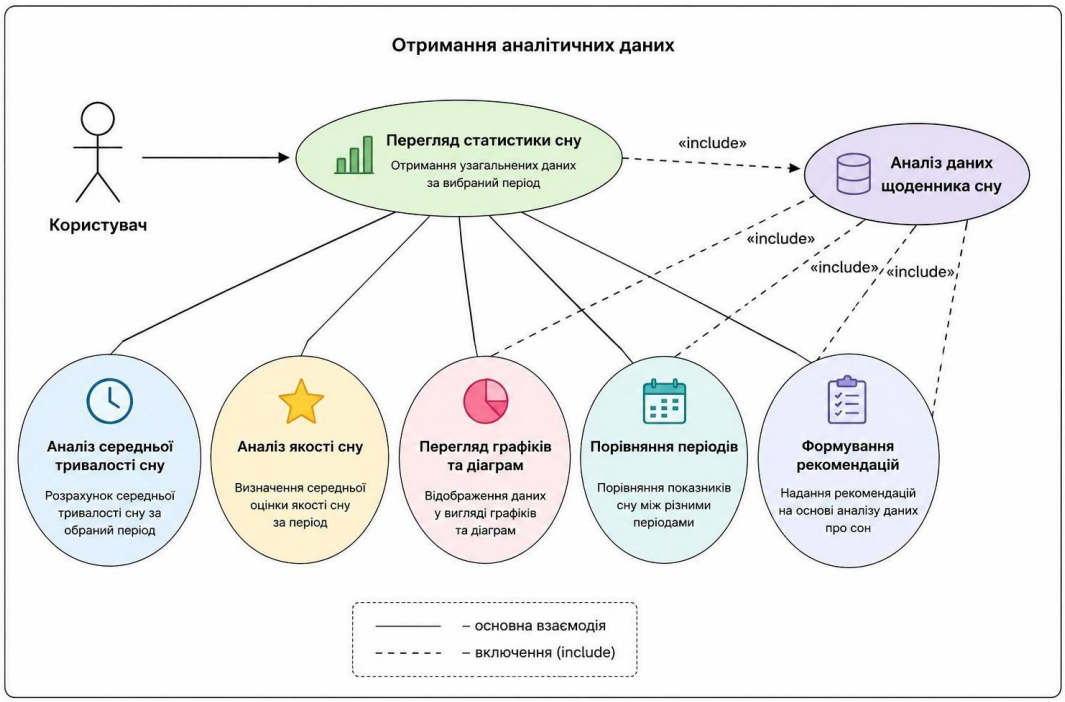

Рисунок 2.6 – Діаграма варіантів використання модуля аналітики

На рисунку 2.6 показано основні аналітичні можливості системи: отримання середньої тривалості сну, визначення оцінки якості сну, формування розподілу оцінок та перегляд динаміки сну за обраний період. Результати цих операцій використовуються для побудови графіків і надання користувачу зрозумілої інформації про його режим відпочинку.

2.5 Архітектура вебзастосунку та структура інтерфейсу

Під час проєктування вебсайту «SleepHelper» важливим етапом стало формування архітектури системи та структури користувацького інтерфейсу. Правильно спроєктована архітектура дозволяє забезпечити стабільну роботу застосунку, зручність підтримки програмного коду та можливість подальшого розширення функціоналу.

Для реалізації вебсайту використовується клієнт-серверна архітектура, яка передбачає розподіл функцій між клієнтською частиною, сервером та базою

даних. Такий підхід є одним із найбільш поширених у сучасній веброзробці та використовується більшістю сучасних вебзастосунків [33].

Крім архітектури програмної системи, важливим аспектом є проєктування структури інтерфейсу та логіки переходів між сторінками. Продумана навігація забезпечує швидкий доступ користувача до основних функцій системи та підвищує зручність використання вебсайту.

2.5.1 Схема взаємодії клієнтської та серверної частин

Для реалізації вебсайту «SleepHelper» використовується трирівнева архітектура, що складається з клієнтської частини, серверної частини та бази даних. Такий підхід дозволяє розділити функції відображення інформації, обробки бізнес-логіки та зберігання даних, що позитивно впливає на масштабованість, продуктивність і зручність супроводу програмного продукту.

Клієнтська частина реалізована за допомогою бібліотеки React і відповідає за відображення інтерфейсу користувача, обробку дій користувача та формування HTTP-запитів. Саме на клієнтському рівні реалізовано сторінки реєстрації та авторизації, щоденник сну, калькулятор часу засинання та пробудження, модуль статистики й інші елементи інтерфейсу. Використання React дозволяє створити динамічний інтерфейс із швидким оновленням даних без повного перезавантаження сторінок.

Серверна частина створюється на основі FastAPI та виконує функції обробки запитів, перевірки даних, авторизації користувачів та взаємодії з базою даних. На сервері реалізовано REST API, через яке здійснюється обмін даними між компонентами системи. Крім цього, сервер забезпечує перевірку прав доступу користувачів, обробку статистичних даних та формування відповідей для клієнтської частини.

Для зберігання інформації про користувачів та записи щоденника сну використовується MongoDB. Використання документоорієнтованої бази даних дозволяє зберігати інформацію у гнучкому форматі та спрощує роботу зі структурами даних, які можуть змінюватися в процесі розвитку системи. У базі

даних зберігаються облікові записи користувачів, записи про сон, показники якості сну та інші службові дані, необхідні для роботи вебзастосунку.

Обмін даними між клієнтською та серверною частинами здійснюється через REST API з використанням формату JSON. Такий підхід дозволяє забезпечити незалежність компонентів системи, спрощує інтеграцію між різними технологіями та робить архітектуру більш гнучкою для подальшого розвитку. Крім того, використання REST API дозволяє в майбутньому створити мобільний застосунок або інші клієнтські сервіси без необхідності змінювати серверну логіку. Схему взаємодії клієнтської та серверної частин вебсайту наведено на рисунку 2.7.

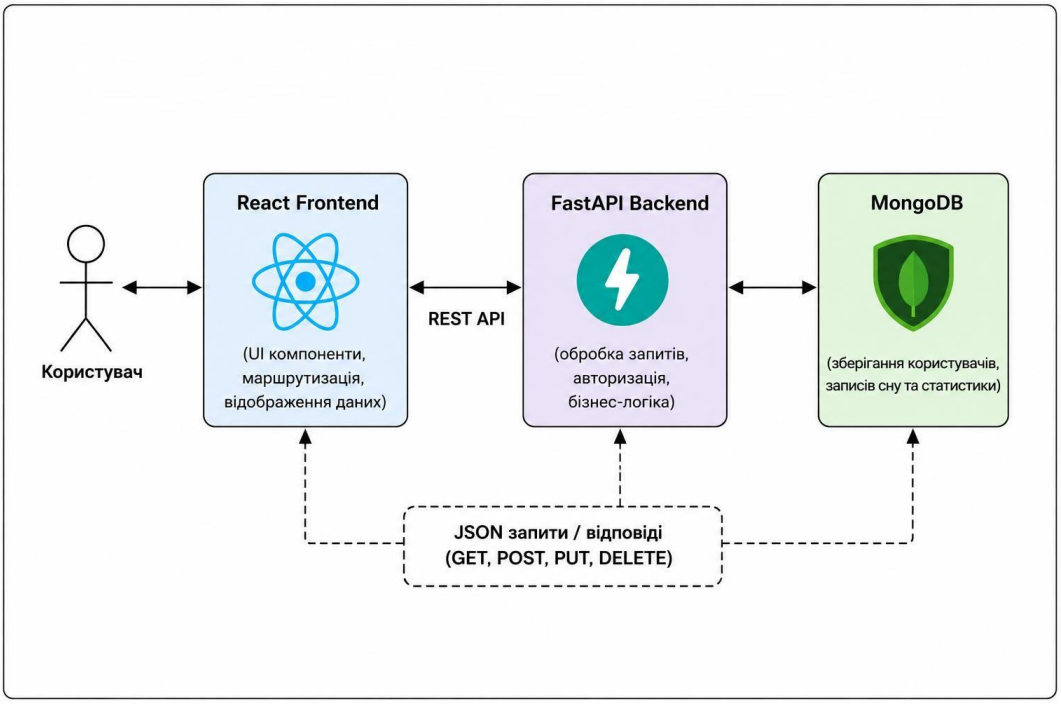

Рисунок 2.7 – Схема взаємодії компонентів вебсайту “SleepHelper”

Після надсилання запиту користувачем клієнтська частина передає дані до серверної частини через REST API. Сервер виконує необхідну обробку інформації, взаємодіє з MongoDB та повертає результат у форматі JSON. Отримані дані відображаються в інтерфейсі користувача у вигляді сторінок, таблиць, графіків та статистичних блоків.

2.5.2 Карта екранів та логіка навігації сайту

Під час проєктування інтерфейсу вебсайту було сформовано структуру навігації між основними сторінками системи. Головною метою є забезпечення швидкого доступу користувача до основного функціоналу та спрощення взаємодії із системою.

Після відкриття вебсайту користувач може перейти до сторінок входу або реєстрації. Після успішної авторизації відкривається головна сторінка, яка є центральним елементом навігації та забезпечує доступ до всіх функціональних модулів системи.

Основними функціональними модулями вебсайту є щоденник сну, статистика, калькулятор сну та розділ рекомендацій щодо покращення якості відпочинку. Усередині кожного модуля передбачено додаткові сторінки та функції, необхідні для роботи користувача із системою. Карту екранів вебсайту «SleepHelper» наведено на рисунку 2.8.

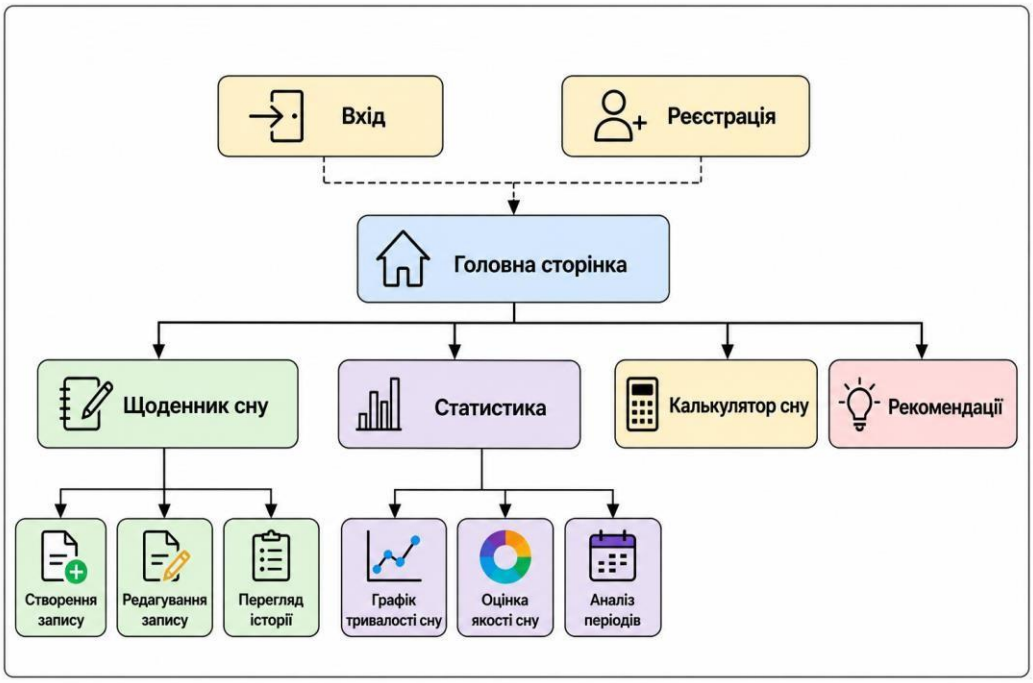

Рисунок 2.8 – Карта екранів вебсайту “SllepHelper”

Як показано на рисунку 2.8, центральне місце в навігаційній структурі займає головна сторінка. Саме через неї користувач отримує доступ до всіх

основних можливостей системи. Такий підхід забезпечує логічну структуру переходів між екранами та зменшує кількість дій, необхідних для доступу до потрібної функції.

Запропонована структура навігації дозволяє легко масштабувати систему шляхом додавання нових модулів або сторінок без суттєвих змін загальної архітектури інтерфейсу.

2.6 Висновки до другого розділу

У другому розділі було обґрунтовано вибір технологічного стеку та виконано проєктування основних компонентів вебсайту «SleepHelper». Для реалізації клієнтської частини обрано бібліотеку React у поєднанні з мовою TypeScript, що забезпечує компонентний підхід до розробки інтерфейсу, високу продуктивність роботи застосунку та підвищення надійності програмного коду завдяки статичній типізації. Для серверної частини було обрано фреймворк FastAPI, який підтримує асинхронну обробку запитів, має високу швидкодію та зручні засоби створення REST API.

У процесі проєктування системи було виконано аналіз різних підходів до організації зберігання даних та обґрунтовано вибір документоорієнтованої СУБД MongoDB. Розроблено структуру документів для зберігання записів щоденника сну та визначено підхід до формування статистичних показників без дублювання інформації в окремих колекціях. Також було спроєктовано REST API для реалізації функцій аутентифікації користувачів, керування записами щоденника та отримання аналітичних даних.

Також, приділено увагу архітектурі вебзастосунку та структурі інтерфейсу користувача. Сформовано схему взаємодії клієнтської та серверної частин, а також карту екранів і логіку навігації вебсайту. Отримані результати створюють основу для подальшої програмної реалізації системи та дозволяють перейти до розробки функціональних модулів вебсайту «SleepHelper» у наступному розділі.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБСАЙТУ

3.1 Створення серверної частини на FastAPI

Для реалізації серверної частини вебсайту «SleepHelper» було використано фреймворк FastAPI. Серверна частина відповідає за обробку запитів користувачів, виконання операцій авторизації, роботу із записами щоденника сну та взаємодію з базою даних MongoDB. Усі дані між клієнтською та серверною частинами передаються у форматі JSON через REST API.

Для запуску серверної частини використовується ASGI-сервер Uvicorn [34]. Під час розробки застосовується режим автоматичного перезавантаження, що дозволяє миттєво застосовувати зміни в коді без ручного перезапуску сервера. Запуск серверної частини вебсайту наведено на рисунку 3.1.

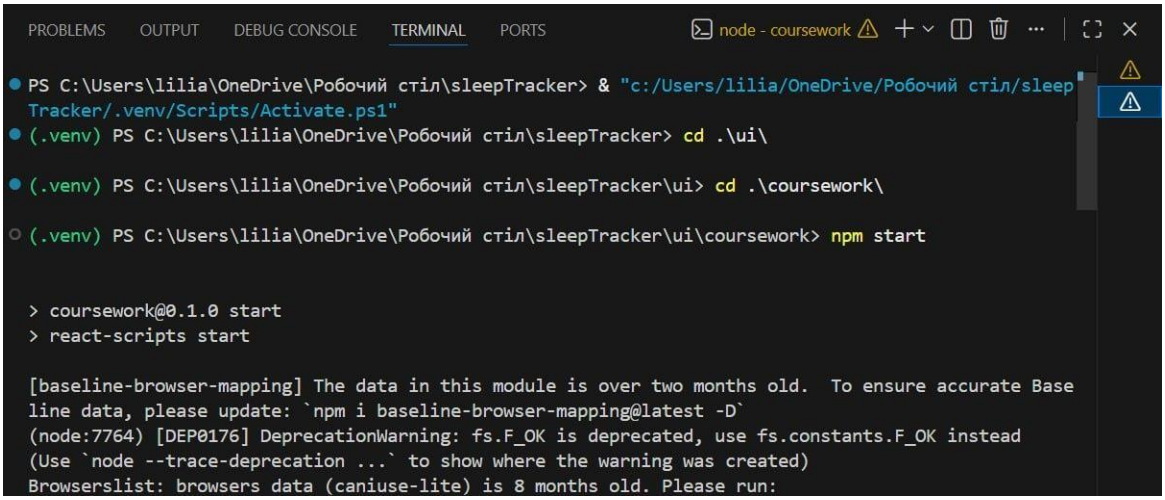

Рисунок 3.1 – Запуск серверної частини вебсайту

Після запуску сервер стає доступним за локальною адресою http://127.0.0.1:8000. Як видно з рисунка 3.1, сервер успішно приймає HTTP-запити та повертає відповіді клієнтській частині. Також у журналі відображаються результати виконання запитів і коди відповідей сервера, що значно спрощує процес тестування та налагодження вебзастосунку.

У лістингу 3.1 наведено приклад реалізації REST API для роботи із записами щоденника сну. Метод GET використовується для отримання записів користувача за обраний період, метод PUT забезпечує редагування існуючих записів, а метод DELETE дозволяє видаляти записи із бази даних. Для захисту даних використовується механізм аутентифікації через залежність get_current_user, що забезпечує доступ лише до власних записів авторизованого користувача.

Лістинг 3.1 – Реалізація API для роботи із записами щоденника сну

```
@app.get("/api/sleep", response_model=List[SleepRecord])
async def get_records(date: Optional[str] = None, user_id: str =
Depends(get_current_user)):
    query = {"user_id": user_id}
    if date:
        query["date"] = date
    records = await collection.find(query, {"_id": 0, "user_id":
0}).to_list(length=1000)
    return records

@app.put("/api/sleep/{record_id}", response_model=SleepRecord)
async def update_record(record_id: int, record: SleepRecord,
user_id: str = Depends(get_current_user)):
    data = record.model_dump()
    data["user_id"] = user_id
    result = await collection.replace_one({"id": record_id,
"user_id": user_id}, data)
    if result.matched_count == 0:
        raise HTTPException(status_code=404, detail="Запис не
знайдено")
    return record

@app.delete("/api/sleep/{record_id}")
async def delete_record(record_id: int, user_id: str =
Depends(get_current_user)):
    result = await collection.delete_one({"id": record_id,
"user_id": user_id})
    if result.deleted_count == 0:
        raise HTTPException(status_code=404, detail="Запис не
знайдено")
    return {"status": "deleted", "id": record_id}
```

Таким чином, у межах даного етапу було створено серверну частину вебсайту «SleepHelper» на основі фреймворку FastAPI. Реалізована архітектура

забезпечує обробку HTTP-запитів, взаємодію з базою даних MongoDB та підтримку REST API для роботи з даними користувачів і записами щоденника сну. Використання асинхронного підходу дозволяє підвищити продуктивність системи та забезпечити швидке виконання запитів.

3.2 Реалізація системи реєстрації та автоматизації користувачів

Для забезпечення захисту персональних даних користувачів у вебсайті «SleepHelper» реалізовано механізм реєстрації та авторизації. Наявність облікового запису дозволяє зберігати записи щоденника сну, формувати індивідуальну статистику та забезпечувати доступ до персональних даних лише їх власнику.

Під час реєстрації користувач вводить ім’я, адресу електронної пошти та пароль. Після успішного створення облікового запису сервер повертає токен доступу, який використовується для подальшої роботи із захищеними розділами системи. Для входу в систему користувач використовує електронну пошту та пароль, після чого сервер перевіряє правильність введених даних та виконує авторизацію. Фрагмент програмної реалізації процесу авторизації наведено в лістингу 3.2.

Лістинг 3.2 – Реалізація автоматизації користувача

```
const handleSubmit = async () => {
    if (!email || !password) {
        setError("Заповніть всі поля");
        return;
    }
    setIsLoading(true);
    try {
        const res = await fetch(
            "http://127.0.0.1:8000/api/auth/login",
            {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
                },
                body: JSON.stringify({
                    email,
```

```
                    password
                }),
            }
        );
        const data = await res.json();
        if (!res.ok) {
            setError(data.detail || "Помилка входу");
            return;
        }
        login(data.token, {
            name: data.name,
            email: data.email
        });
        navigate("/diary");
    } catch {
        setError("Сервер недоступний");
    }
};
```

У лістингу 3.2 наведено реалізацію функції авторизації користувача. Після перевірки правильності заповнення полів клієнтська частина надсилає POST-запит до серверного ендпоінта авторизації. У разі успішної відповіді отриманий токен та інформація про користувача зберігаються в контексті авторизації, після чого виконується перехід до сторінки щоденника сну.

Фрагмент програмної реалізації процесу реєстрації нового користувача наведено в лістингу 3.3.

Лістинг 3.3 – Реалізація реєстрації користувача

```
const handleSubmit = async () => {
    if (!name || !email || !password) {
        setError("Заповніть всі поля");
        return;
    }
    if (password.length < 6) {
        setError("Пароль має бути мінімум 6 символів");
        return;
    }
    setIsLoading(true);
    try {
        const res = await fetch(
            "http://127.0.0.1:8000/api/auth/register",
            {
                method: "POST",
                headers: {
                    "Content-Type": "application/json"
```

```
                },
                body: JSON.stringify({
                    name,
                    email,
                    password
                }),
            }
        );
        const data = await res.json();
        if (!res.ok) {
            setError(data.detail || "Помилка реєстрації");
            return;
        }
        login(data.token, {
            name: data.name,
            email: data.email
        });
        navigate("/diary");
    } catch {
        setError("Сервер недоступний");
    }
};
```

У лістингу 3.3 показано реалізацію реєстрації нового користувача. Перед відправленням запиту виконується перевірка коректності введених даних та мінімальної довжини пароля. Після успішної реєстрації користувач автоматично авторизується в системі та отримує доступ до персональних можливостей вебсайту.

Отже, реалізований механізм реєстрації та авторизації забезпечує захист персональних даних користувачів, підтримує роботу з обліковими записами та створює основу для функціонування інших модулів вебсайту «SleepHelper».

3.3 Розробка модулів збереження та обробки даних про сон

Основним функціональним модулем вебсайту «SleepHelper» є підсистема ведення щоденника сну. Вона забезпечує збереження інформації про час засинання та пробудження, оцінку якості сну, а також додаткові нотатки користувача. На основі накопичених даних надалі формується статистика та аналітична інформація.

Зберігання даних у системі реалізовано на двох рівнях. Довготривале зберігання здійснюється у базі даних MongoDB, де зберігаються всі записи користувачів. Для підвищення швидкодії застосунку використовується локальне кешування у стані React. Після отримання даних із сервера записи зберігаються у пам’яті клієнтської частини та використовуються для подальшої фільтрації й аналізу без додаткових запитів до сервера. Інтерфейс сторінки щоденника сну наведено на рисунку 3.2.

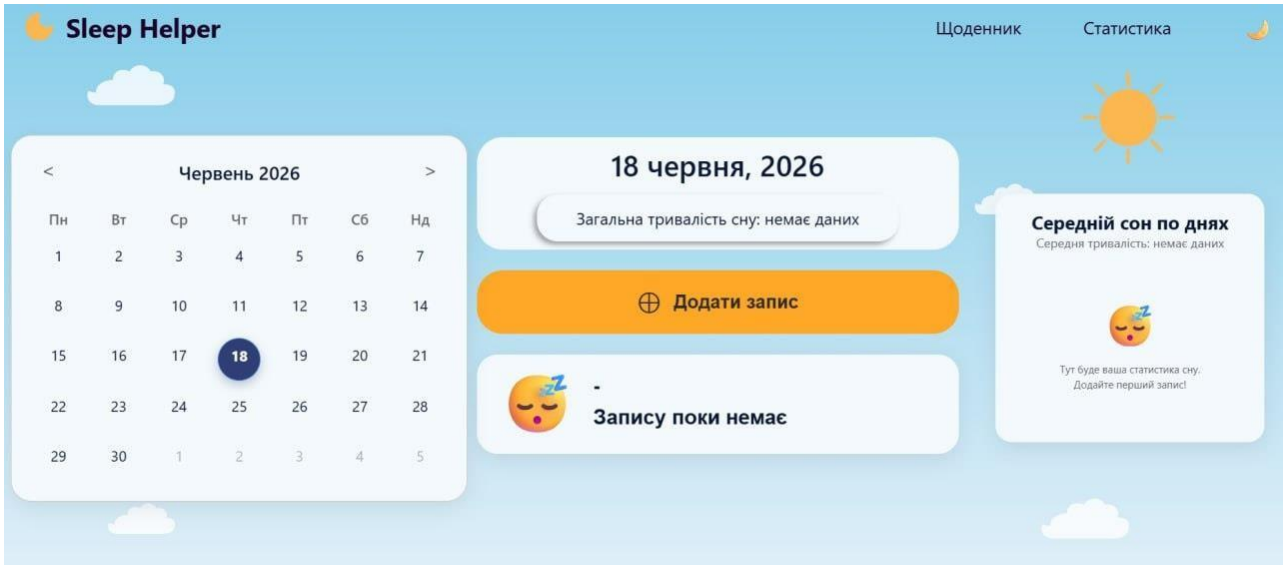

Рисунок 3.2 – Головна сторінка щоденника сну

Як показано на рисунку 3.2, користувач може переглядати календар записів, отримувати інформацію про загальну тривалість сну за обраний день, додавати нові записи та переглядати коротку статистику. Відображення інформації здійснюється на основі даних, отриманих із серверної частини та збережених у базі даних MongoDB.

Для введення інформації про сон реалізовано окрему форму створення запису, яка наведена на рисунку 3.3.

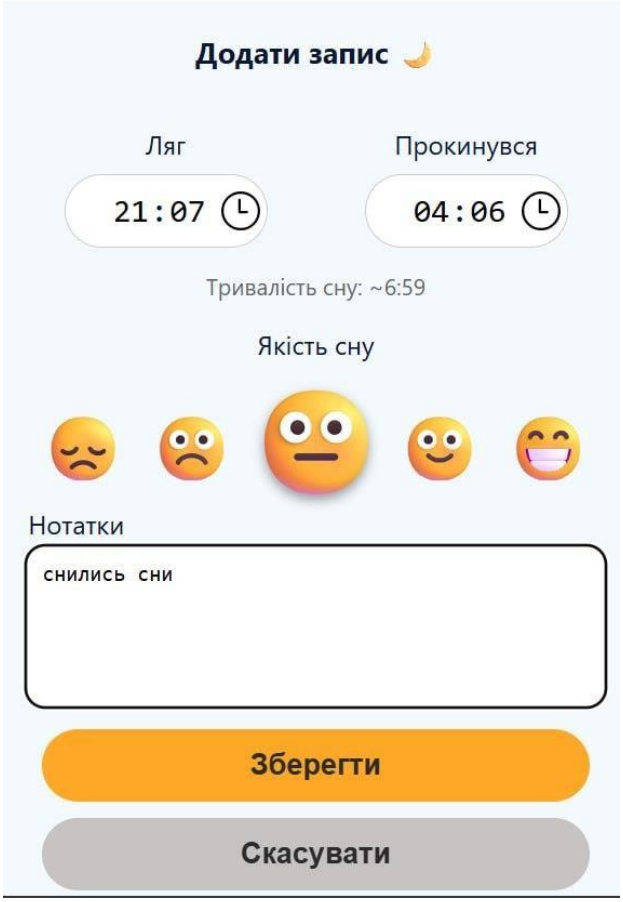

Рисунок 3.3 – Форма додавання запису про сон

Форма додавання запису дозволяє користувачу вказати час засинання та пробудження, оцінити якість сну за п’ятибальною шкалою та додати власні нотатки. Після натискання кнопки «Зберегти» інформація передається до серверної частини та записується до бази даних.

Для автоматизації введення даних реалізовано механізм автоматичного розрахунку тривалості сну. Користувачеві не потрібно самостійно виконувати розрахунки, оскільки система визначає тривалість відпочинку на основі введеного часу засинання та пробудження.

Для керування записами щоденника сну реалізовано функції додавання, редагування та видалення записів. Після виконання операцій клієнтська частина оновлює локальний стан, що дозволяє відразу відобразити зміни в інтерфейсі без повного перезавантаження сторінки.

Фрагмент реалізації видалення запису зі щоденника сну наведено в лістингу 3.4.

Лістинг 3.4 – Реалізація видалення запису зі щоденника сну

```
const handleDelete = async () => {
    if (!selectedRecord) return;

    try {
        const res = await fetch(
            `${API_URL}/sleep/${selectedRecord.id}`,
            {
                method: "DELETE",
                headers: {
                    Authorization: `Bearer ${token}`,
                },
            }
        );

        if (res.ok) {
            setAllRecords(prev =>
                prev.filter(r => r.id !== selectedRecord.id)
            );
            setIsModalOpen(false);
        } else {
            const message = await parseErrorMessage(res);
            alert(`Помилка сервера при видаленні: ${message}`);
            console.error("Delete failed:", res.status, message);
        }
    } catch (e) {
        alert("Помилка мережі. Перевірте чи запущений сервер.");
        console.error(e);
    }
};
```

У лістингу 3.4 показано реалізацію видалення запису зі щоденника сну. Спочатку виконується перевірка, чи обрано запис для видалення. Після цього клієнтська частина надсилає DELETE-запит до серверної частини із зазначенням ідентифікатора запису. Для підтвердження прав доступу до запиту додається токен авторизації.

Якщо сервер успішно виконує операцію, запис видаляється з локального стану allRecords за допомогою методу filter, а модальне вікно закривається. У разі помилки серверної частини або відсутності з’єднання користувач отримує відповідне повідомлення. Такий підхід дозволяє забезпечити зручну взаємодію з даними та швидке оновлення інтерфейсу після виконання операції.

3.4 Реалізація клієнтської частини на React та TypeScript

Клієнтська частина вебсайту «SleepHelper» реалізована з використанням бібліотеки React та мови програмування TypeScript. Такий підхід дозволив створити сучасний односторінковий вебзастосунок (Single Page Application, SPA), у якому взаємодія користувача із системою відбувається без повного перезавантаження сторінок. Завдяки цьому забезпечується швидке завантаження даних, плавна навігація між розділами та покращується загальний користувацький досвід.

Використання TypeScript додатково підвищує надійність програмного коду за рахунок статичної типізації. Це дозволяє виявляти частину помилок ще на етапі розробки, забезпечує кращу підтримку великих проєктів та спрощує взаємодію між окремими компонентами застосунку. Для даного проєкту використання TypeScript було особливо корисним під час роботи з об'єктами записів сну, даними користувачів та статистичними показниками.

Навігація між сторінками реалізована за допомогою бібліотеки React Router. Вона забезпечує маршрутизацію між окремими модулями вебсайту та дозволяє реалізувати механізм обмеження доступу до сторінок, що працюють із персональними даними користувача. Для цього використовується компонент PrivateRoute, який перевіряє наявність авторизації перед відкриттям відповідного розділу системи.

Архітектура клієнтської частини побудована за компонентним принципом. Кожен функціональний елемент вебсайту реалізований у вигляді окремого React-компонента, який відповідає лише за визначену частину інтерфейсу або бізнес-логіки. Такий підхід забезпечує повторне використання програмного коду, спрощує тестування окремих модулів та дозволяє легко масштабувати систему в майбутньому.

Для зберігання інформації про авторизованого користувача використовується React Context API. Після входу в систему токен авторизації та базова інформація про користувача зберігаються у локальному сховищі

браузера (localStorage), що дозволяє зберігати активну сесію навіть після закриття вкладки або перезавантаження сторінки. При повторному відкритті вебсайту дані автоматично відновлюються, а користувачеві не потрібно повторно виконувати вхід до системи.

У процесі розробки було створено набір взаємопов'язаних компонентів, які забезпечують роботу всіх функціональних модулів вебсайту. До них належать компоненти календаря, щоденника сну, статистичних графіків, аналітичних карток, модальних вікон, калькулятора сну, блоку рекомендацій та релаксаційних вправ. Основні компоненти клієнтської частини наведено в таблиці 3.1.

Таблиця 3.1 – Основні компоненти клієнтської частини

Компонент	Призначення
Calendar	Місячний календар із підсвічуванням дат записів
Modal	Модальне вікно для створення/редагування запису сну
WeeklyStats	Зведена статистика за тиждень
DreamJournal	Щоденник сновидінь
SleepDurationChart	Графік тривалості сну (лінійний/стовпчастий)
SleepQualityDistributionChart	Кругова діаграма розподілу якості сну
SleepScheduleChart	Графік розкладу сну(час засинання/пробудження)
OverallQuality	Картка загальної якості сну за період
PeriodSelector	Вибір аналітичного періоду
DailyTip	Модальне вікно щоденної поради

Більшість функціональності вебсайту реалізована за допомогою спеціалізованих компонентів. Такий підхід дозволяє відокремити логіку роботи окремих модулів та підвищити рівень повторного використання програмного коду. Завдяки цьому кожен компонент може розроблятися, тестуватися та вдосконалюватися незалежно від інших частин системи.

3.4.1 Головна сторінка (Інформаційна панель користувача)

Головна сторінка є центральним елементом вебзастосунку та виконує роль навігаційного центру для користувача. Саме з неї користувач отримує доступ до основних функцій системи: ведення щоденника сну, перегляду статистики, використання калькулятора сну та розділу релаксації.

У верхній частині сторінки розміщено навігаційне меню, яке дозволяє швидко переходити між доступними модулями системи. Також у шапці сайту знаходиться кнопка перемикання теми оформлення, що забезпечує можливість використання світлого або темного режиму роботи.

Для реалізації сторінки використано компоненти Layout, Header та набір навігаційних карток. Управління переходами між сторінками здійснюється за допомогою бібліотеки React Router. На рисунку 3.4 зображено головну сторінку вебсайту.

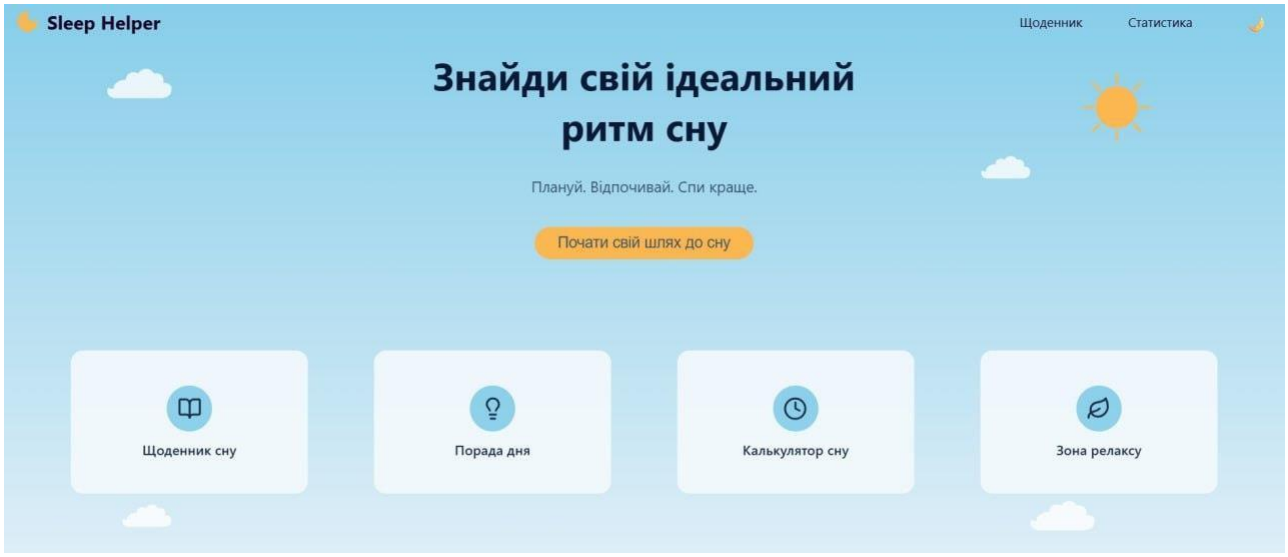

Рисунок 3.4 – Головна сторінка застосунку

Основна частина сторінки містить короткий опис призначення застосунку та набір інтерактивних карток, кожна з яких відповідає окремому функціональному модулю. При натисканні на картку користувач перенаправляється до відповідного розділу системи.

3.4.2 Сторінки входу та реєстрації в системі

Сторінка входу призначена для авторизації вже зареєстрованого користувача. Вона містить поля для введення електронної пошти та пароля, а також кнопку підтвердження входу. Після натискання кнопки система надсилає запит до серверної частини та перевіряє введені дані. Інтерфейс сторінки входу наведено на рисунку 3.5.

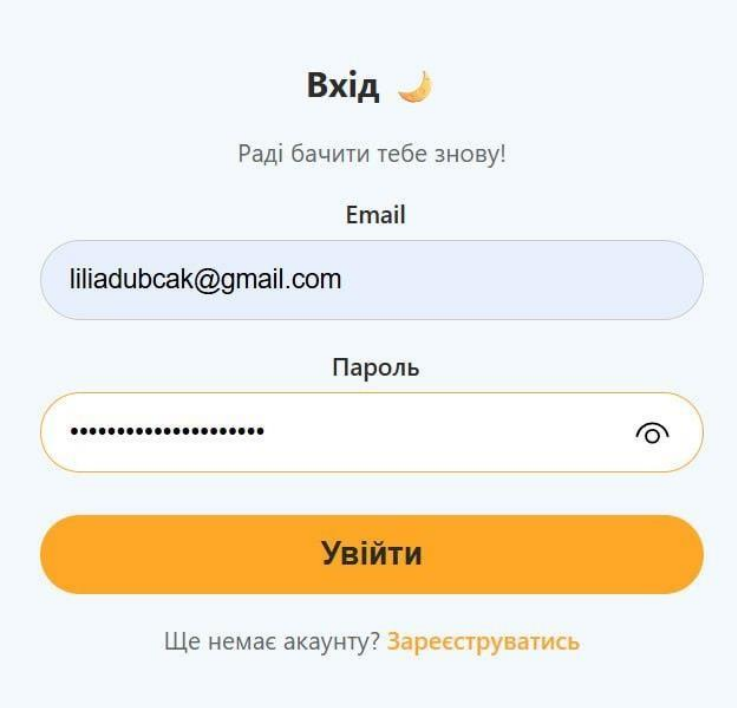

Рисунок 3.5 – Сторінка входу в систему

Як видно з рисунка 3.5, форма входу має просту структуру та не перевантажена зайвими елементами. Також передбачено посилання для переходу до сторінки реєстрації, якщо користувач ще не має облікового запису.

Сторінка реєстрації використовується для створення нового облікового запису. Користувач вводить ім’я, електронну пошту та пароль. Після успішної реєстрації система створює обліковий запис і надає доступ до персональних розділів вебсайту. Інтерфейс сторінки реєстрації наведено на рисунку 3.6.

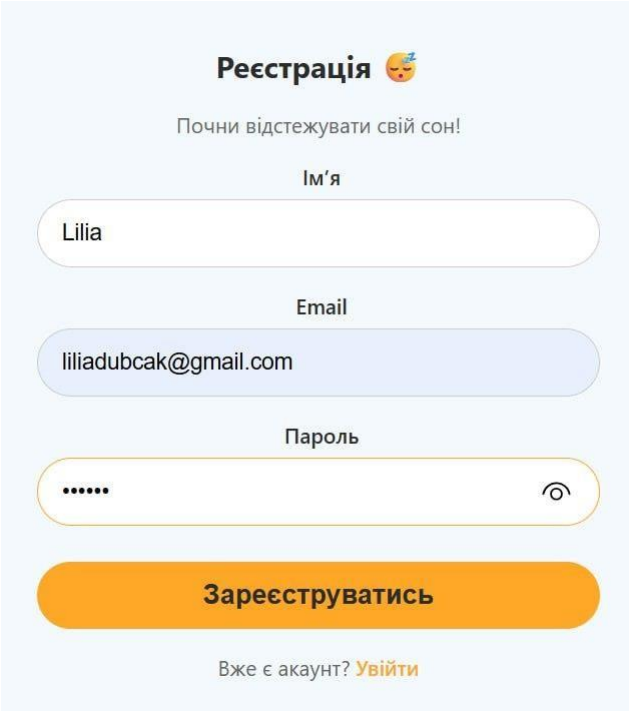

Рисунок 3.6 – Сторінка реєстрації користувача

На рисунку 3.6 показано, що форма реєстрації містить мінімально необхідні поля для створення облікового запису. Такий підхід спрощує початок роботи із системою та не ускладнює процес реєстрації для нового користувача.

3.4.3 Інтерактивний щоденник сну

Інтерактивний щоденник сну є основним функціональним модулем вебсайту «SleepHelper». Його призначення полягає у збереженні інформації про сон користувача, веденні історії записів та накопиченні даних для подальшого статистичного аналізу. Саме цей модуль забезпечує збір основної інформації, яка використовується іншими компонентами системи, зокрема модулем статистики та аналітики.

Після відкриття сторінки система автоматично завантажує записи поточного користувача із серверної частини та відображає їх у зручному для перегляду вигляді. Для навігації між записами використовується календар, який дозволяє обирати конкретну дату та переглядати інформацію про сон за відповідний день. Загальний вигляд сторінки щоденника сну наведено на рисунку 3.7.

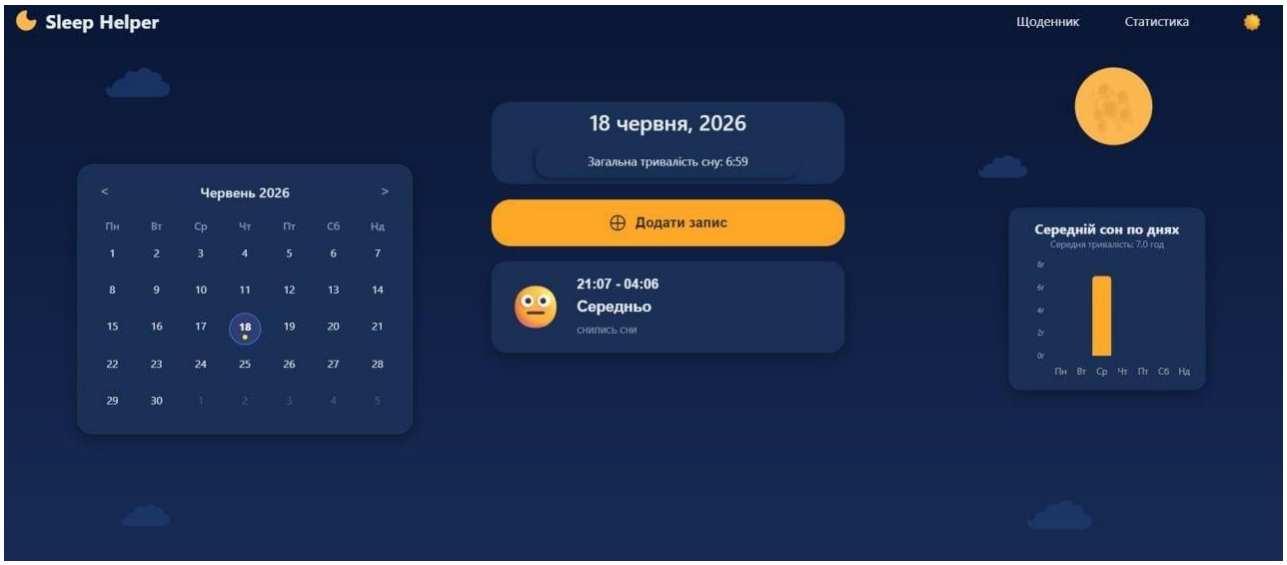

Рисунок 3.7 – Інтерактивний щоденник сну

Сторінка складається з декількох функціональних блоків. У лівій частині розташований календар для вибору дати, у центральній частині відображається загальна тривалість сну та список записів за обраний день, а праворуч знаходиться блок короткої статистики. Така структура дозволяє користувачу швидко отримувати необхідну інформацію та працювати із записами без переходу між різними сторінками.

Використання календаря значно спрощує навігацію між записами та забезпечує швидкий доступ до інформації за будь-який день. Після вибору дати система автоматично відображає всі записи, пов'язані з обраним днем, а також розраховує сумарну тривалість сну. Це дозволяє користувачу контролювати власний режим відпочинку та відстежувати зміни показників упродовж тривалого часу.

Додатково на сторінці реалізовано механізм перегляду та редагування існуючих записів. Кожен запис відображається у вигляді окремої інформаційної картки, що містить основні дані про сон. Завдяки цьому користувач може швидко переглянути внесену інформацію та за потреби внести зміни або видалити запис. Наявність короткого статистичного блоку дозволяє одразу отримати узагальнену інформацію про сон без необхідності переходу до окремого модуля статистики.

Для створення нового запису передбачено спеціальну форму введення даних, яка відкривається після натискання кнопки «Додати запис». Користувач може вказати час засинання, час пробудження, оцінити якість сну та додати власні нотатки. У процесі введення даних система автоматично перевіряє коректність заповнення полів та забезпечує зручний механізм взаємодії з формою. Інтерфейс створення запису наведено на рисунку 3.8.

Форма створення запису розроблена таким чином, щоб мінімізувати кількість дій користувача під час внесення інформації. Усі необхідні поля згруповані в одному діалоговому вікні, що спрощує процес заповнення та зменшує ймовірність помилок. Після збереження дані автоматично передаються на сервер, записуються до бази даних та відображаються у щоденнику сну без необхідності оновлення сторінки.

Рисунок 3.8 – Форма додавання запису про сон

На рисунку 3.8 показано форму введення даних про сон. Після заповнення полів користувач зберігає запис, а система автоматично надсилає відповідний запит до серверної частини. Усі дані зберігаються у базі даних MongoDB та стають доступними для подальшого перегляду й аналізу.

Однією з особливостей реалізації є автоматичне визначення тривалості сну. Користувачеві достатньо вказати час засинання та пробудження, після чого система самостійно обчислює тривалість відпочинку. Такий підхід дозволяє уникнути помилок під час введення даних та спрощує процес створення записів.

Для відображення інформації за обрану дату використовується механізм фільтрації записів. Після вибору дня у календарі система відбирає відповідні записи та відображає їх у вигляді окремих карток. Приклад відображення записів наведено на рисунку 3.9.

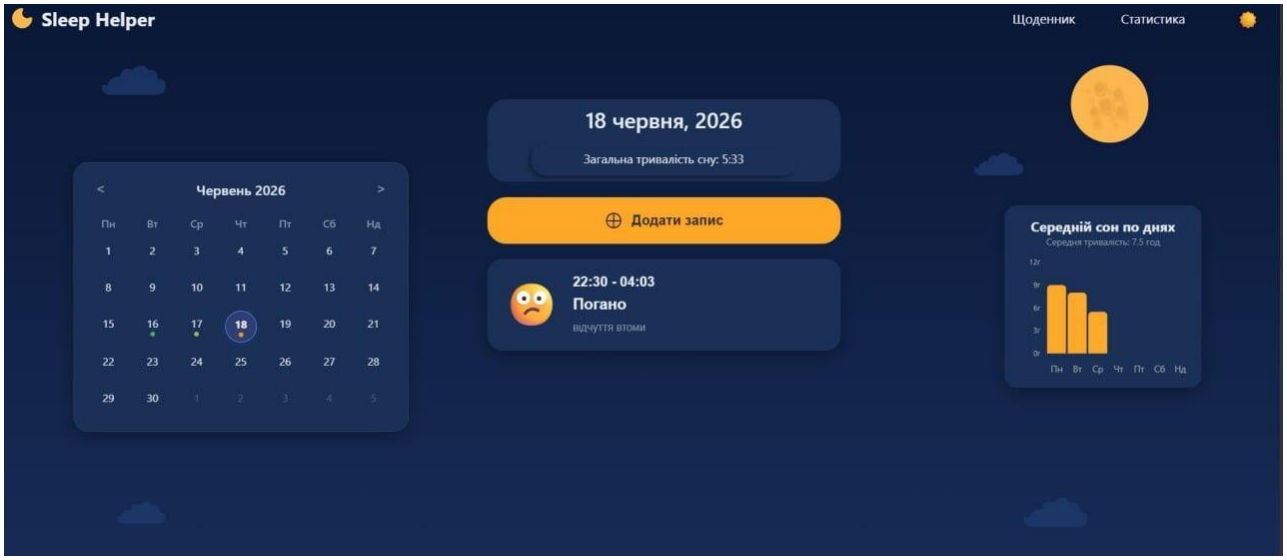

Рисунок 3.9 – Відображення записів користувача за обрану дату

Як видно з рисунка 3.9, кожен запис містить інформацію про час засинання, час пробудження, оцінку якості сну та додаткові нотатки користувача. Для кожного запису доступні функції редагування та видалення, що дозволяє підтримувати актуальність збережених даних.

Таким чином, модуль щоденника сну забезпечує повний цикл роботи з даними користувача: створення, перегляд, редагування та видалення записів. Накопичені дані використовуються для формування статистики та подальшого аналізу режиму сну користувача.

3.4.4 Калькулятор часу відпочинку та пробудження

Одним із додаткових функціональних модулів вебсайту «SleepHelper» є калькулятор сну. Його призначення полягає у допомозі користувачеві під час планування режиму відпочинку шляхом визначення рекомендованого часу засинання або пробудження. Робота калькулятора базується на концепції циклів сну, середня тривалість яких становить приблизно 90 хвилин.

Перед початком розрахунку користувач обирає свій віковий діапазон, оскільки рекомендована тривалість сну залежить від віку людини. Крім того, необхідно вказати один із двох режимів роботи калькулятора: визначення часу засинання або визначення часу пробудження. Після введення часу користувач може виконати розрахунок натисканням відповідної кнопки. Загальний вигляд сторінки калькулятора наведено на рисунку 3.10.

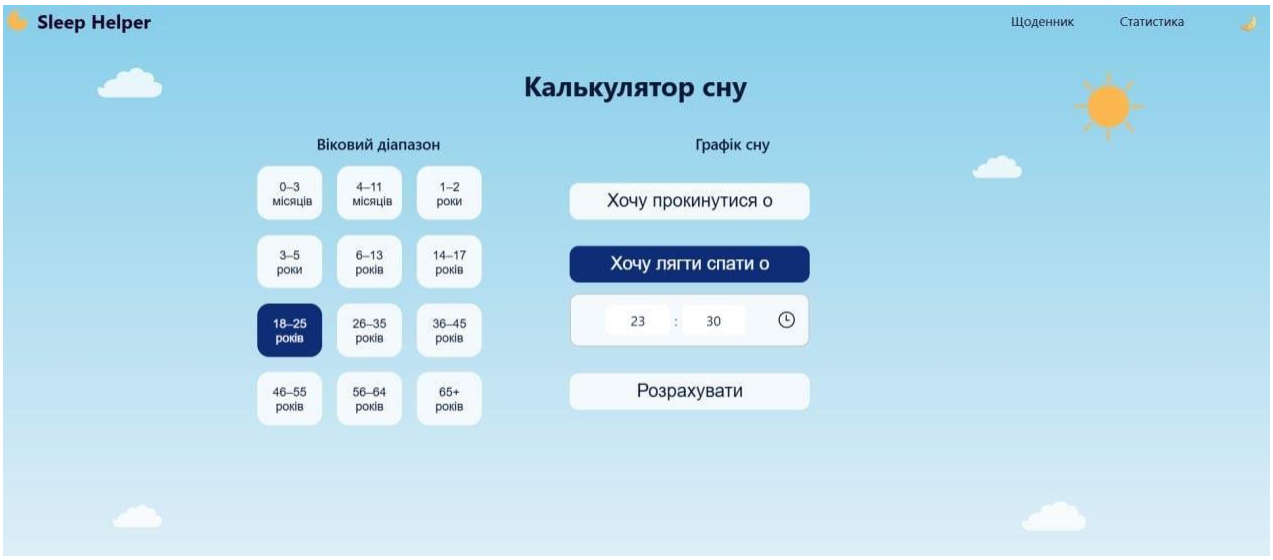

Рисунок 3.10 – Інтерфейс калькулятора сну вебсайту

На рисунку 3.10 показано головні елементи калькулятора. У лівій частині сторінки розташовано перелік вікових категорій, а в правій — елементи керування режимом роботи калькулятора та поле введення часу. Такий підхід дозволяє користувачу швидко виконати необхідний розрахунок без додаткових налаштувань.

Після натискання кнопки «Розрахувати» система виконує обробку введених даних та формує перелік рекомендованих часових проміжків. Результати групуються за рівнем рекомендованої тривалості сну та відображаються у вигляді окремих інформаційних карток. Приклад результатів роботи калькулятора наведено на рисунку 3.11.

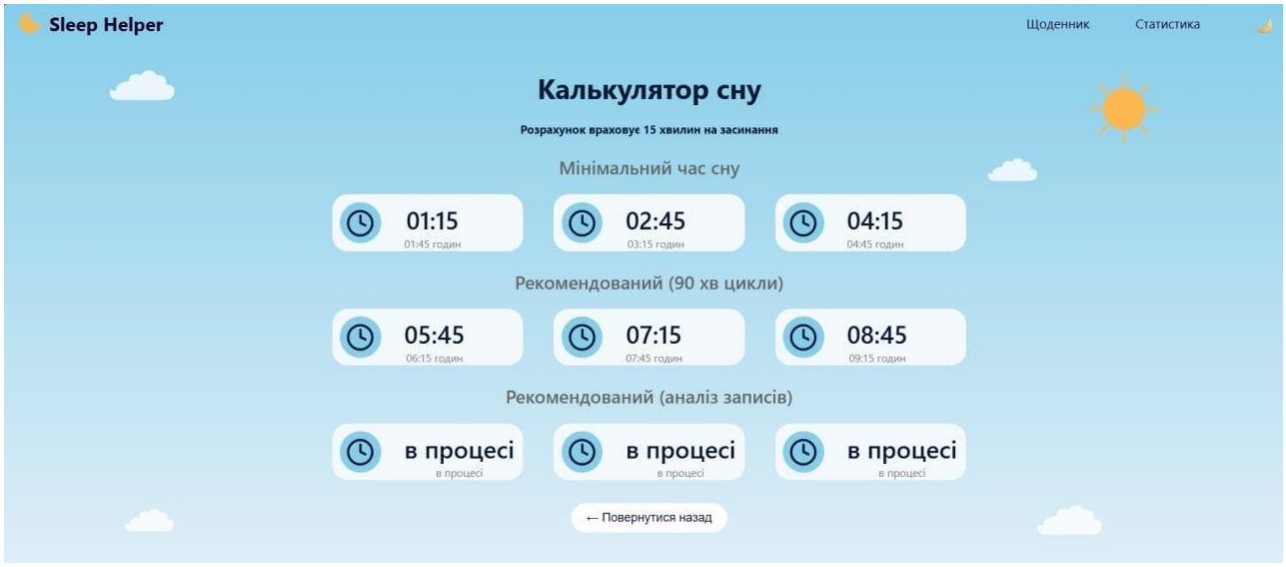

Рисунок 3.11 – Відображення результатів роботи калькулятора сну

Як видно з рисунка 3.11, користувач отримує декілька можливих варіантів часу засинання або пробудження. Для кожного варіанта відображається рекомендований час та орієнтовна тривалість сну. Додатково система повідомляє, що під час обчислень враховується приблизно 15 хвилин, необхідних для засинання.

Результати розрахунків згруповані за рівнем рекомендованої тривалості відпочинку, що дозволяє користувачу обрати найбільш зручний для себе варіант. Такий підхід враховує індивідуальні особливості режиму дня та дає можливість адаптувати рекомендації до різних життєвих ситуацій. Наприклад, користувач може обрати як мінімально допустиму тривалість сну, так і більш оптимальні варіанти, що відповідають рекомендованій кількості циклів сну.

Важливою особливістю калькулятора є використання концепції циклічності сну. Оскільки пробудження наприкінці повного циклу зазвичай є

комфортнішим для людини, система формує рекомендації таким чином, щоб момент пробудження збігався із завершенням чергового циклу. Це дозволяє зменшити відчуття втоми після сну та покращити загальне самопочуття протягом дня.

Крім того, під час формування рекомендацій враховується віковий діапазон користувача. Для різних вікових груп можуть використовуватися різні рекомендації щодо оптимальної тривалості відпочинку, що робить результати більш персоналізованими. Завдяки цьому калькулятор може використовуватися широким колом користувачів та забезпечувати більш коректні рекомендації залежно від їхніх потреб.

Таким чином, модуль калькулятора сну забезпечує швидке отримання рекомендацій щодо часу засинання та пробудження на основі вікових особливостей користувача та стандартної структури циклів сну. Використання такого інструменту допомагає користувачам планувати власний режим відпочинку, формувати корисні звички сну та більш усвідомлено ставитися до контролю його якості. Це сприяє підвищенню ефективності відпочинку та може позитивно впливати на загальний стан здоров’я й працездатність людини.

3.4.5 Зона релаксації та рекомендацій

Для підвищення практичної цінності вебсайту «SleepHelper» було реалізовано окремий розділ релаксації. Його основне призначення полягає у допомозі користувачу під час підготовки до сну та формуванні корисних звичок, які позитивно впливають на якість відпочинку. На відміну від щоденника сну та статистичного модуля, цей розділ орієнтований не на аналіз даних, а на безпосередню підтримку користувача в процесі покращення режиму сну.

Сторінка об’єднує декілька інструментів, серед яких дихальні вправи, релаксаційні аудіозаписи та корисні рекомендації щодо здорового сну. Усі елементи розміщені на одній сторінці, що дозволяє користувачу швидко перемикатися між різними способами релаксації без необхідності відкривати

додаткові розділи застосунку. Загальний вигляд сторінки релаксації наведено на рисунку 3.12.

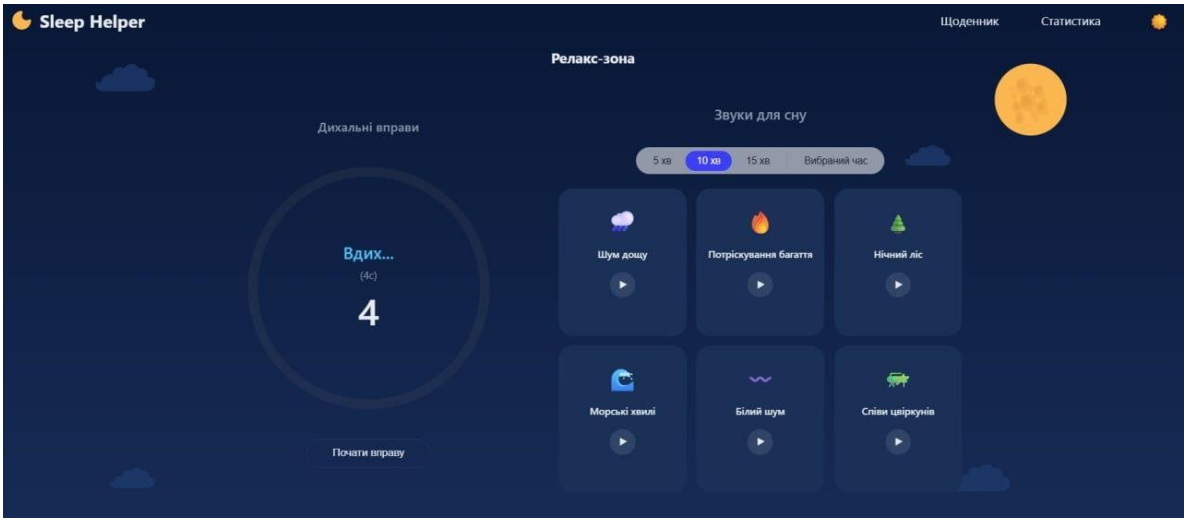

Рисунок 3.12 – Сторінка релаксації вебсайту

На рисунку 3.12 показано основні функціональні блоки розділу. Користувач може обирати різні способи підготовки до сну залежно від власних потреб та вподобань. Такий підхід дозволяє зробити застосунок не лише інструментом для збору статистики, а й засобом підтримки здорових звичок сну.

Розділ релаксації об'єднує декілька інструментів, спрямованих на зниження рівня стресу та покращення процесу засинання. До його складу входять дихальні вправи, корисні рекомендації щодо організації відпочинку та добірка заспокійливих звуків. Завдяки поєднанню цих елементів користувач отримує можливість не лише аналізувати власний сон, а й активно працювати над покращенням його якості.

Одним із ключових компонентів сторінки є модуль дихальних вправ. Його основне завдання полягає у допомозі користувачу під час виконання ритмічного дихання перед сном. Для цього використовується проста анімація та текстові підказки, які допомагають підтримувати правильний ритм вдиху та видиху. Інтерфейс дихальних вправ наведено на рисунку 3.13.

Виконання дихальних вправ сприяє зниженню рівня напруження, нормалізації серцевого ритму та підготовці організму до відпочинку. Анімовані елементи інтерфейсу дозволяють користувачу зосередитися на процесі дихання без необхідності самостійно відраховувати часові інтервали. Завдяки цьому використання модуля є простим та зрозумілим навіть для користувачів, які раніше не практикували подібні техніки релаксації.

Крім того, дихальні вправи можуть використовуватися не лише перед сном, а й протягом дня для зниження рівня стресу та покращення емоційного стану. Інтеграція такого функціоналу у вебзастосунок дозволяє розширити його можливості та зробити більш корисним для користувачів, які прагнуть покращити якість свого відпочинку та загальне самопочуття.

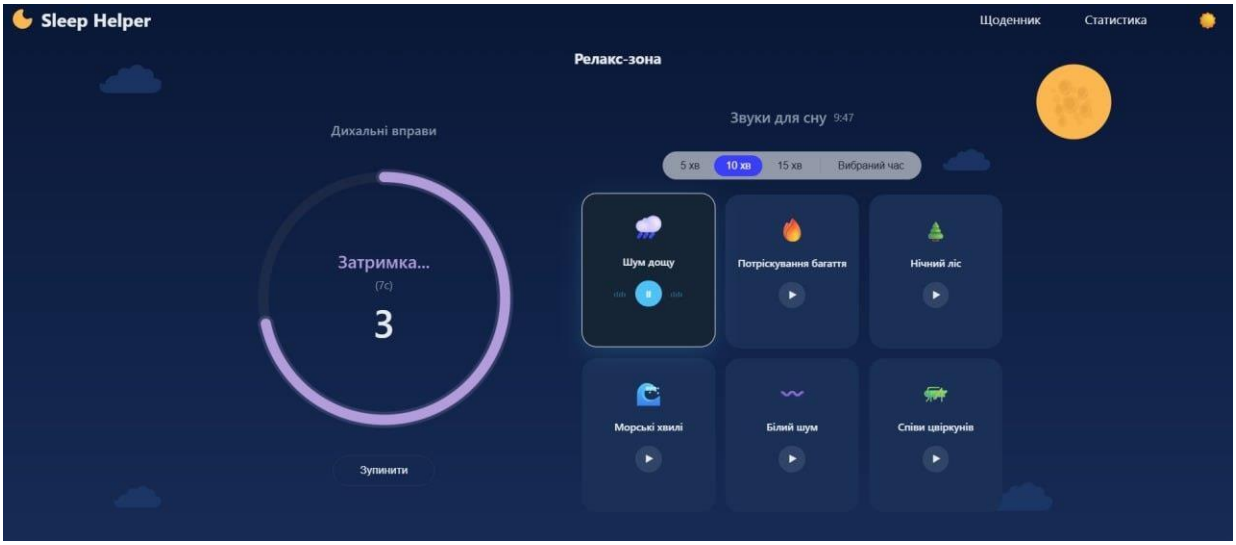

Рисунок 3.13 – Модуль дихальних вправ

Як показано на рисунку 3.13, користувач отримує візуальні підказки для виконання вправи. Регулярне використання таких вправ може сприяти зменшенню рівня стресу та покращенню процесу засинання.

Таким чином, розділ релаксації доповнює основний функціонал вебсайту «SleepHelper» та надає користувачам додаткові інструменти для підготовки до сну. Поєднання дихальних вправ, релаксаційних звуків і практичних

рекомендацій створює комплексний підхід до підтримки здорового способу життя та покращення якості відпочинку.

3.4.6 Модуль візуалізації статистики та графіків

Для аналізу накопичених даних про сон у вебсайті «SleepHelper» реалізовано окремий модуль статистики. Його основним призначенням є надання користувачеві наочного представлення інформації про тривалість та якість сну за різні часові проміжки. Використання графіків і діаграм дозволяє швидко виявляти закономірності, оцінювати стабільність режиму сну та контролювати зміни показників у часі.

Після відкриття сторінки статистики система автоматично завантажує записи користувача та виконує їх попередню обробку. Користувач може обрати необхідний часовий проміжок для аналізу за допомогою спеціального елемента керування. Після зміни періоду відображення статистичні показники автоматично оновлюються відповідно до вибраних даних. Загальний вигляд сторінки статистики наведено на рисунку 3.14.

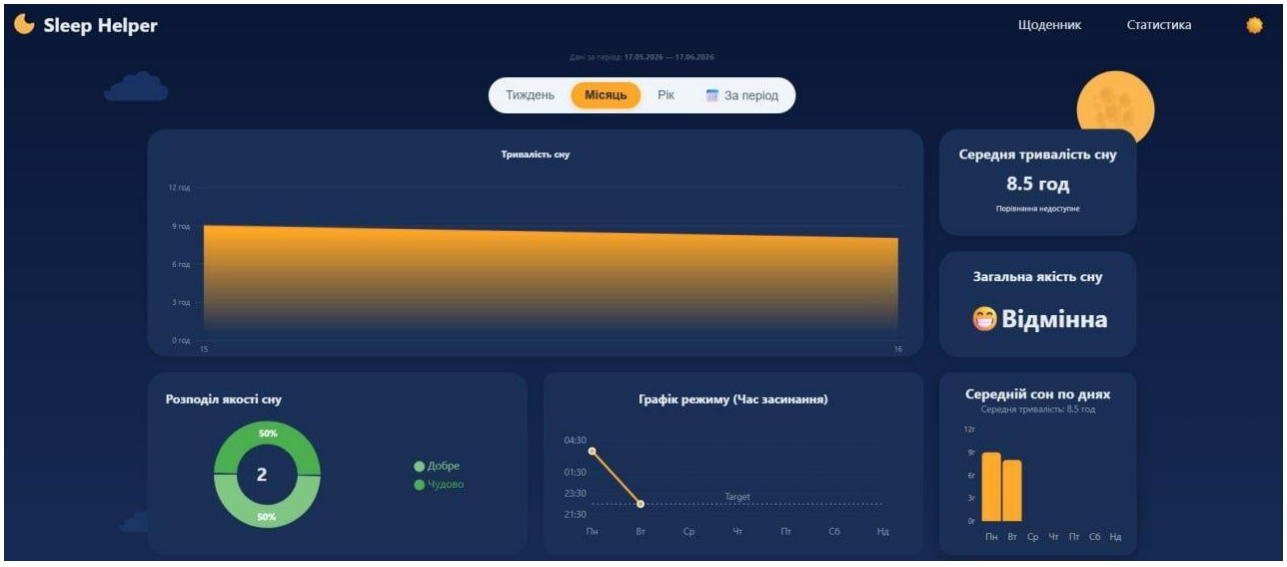

Рисунок 3.14 – Сторінка статистики вебсайту

Як показано на рисунку 3.14, сторінка містить декілька інформаційних блоків, які відображають різні аспекти режиму сну користувача. У верхній

частині знаходяться картки із середньою тривалістю сну та загальною оцінкою його якості. Нижче розташовано набір графіків і діаграм для детальнішого аналізу накопичених даних.

Таким чином, модуль статистики та графіків забезпечує наочне представлення накопичених даних про сон та дозволяє користувачу отримувати детальну інформацію про власний режим відпочинку. Використання графічних засобів візуалізації значно спрощує аналіз даних та підвищує практичну цінність вебсайту «SleepHelper».

3.5 Тестування функціональності вебсайту

Після завершення розробки вебсайту «SleepHelper» було проведено тестування його основних функціональних можливостей. Метою тестування була перевірка коректності роботи всіх реалізованих модулів, оцінка взаємодії між клієнтською та серверною частинами, а також виявлення можливих помилок під час роботи користувача із системою [35].

Тестування виконувалося шляхом послідовної перевірки кожної функціональної можливості вебсайту. Особлива увага приділялася роботі системи реєстрації та авторизації, коректності створення та редагування записів щоденника сну, функціонуванню калькулятора сну, роботі статистичного модуля та відображенню даних користувача. Результати проведеного тестування наведено в таблиці 3.2.

Таблиця 3.2 – Результати тестування функціональності вебсайту

Функціональна можливість	Очікуваний результат	Результат тестування
Реєстрація користувача	Створення нового облікового запису	Успішно
Авторизація користувача	Вхід до системи за логіном і паролем	Успішно

Продовження таблиці 3.2

Калькулятор сну	Формування рекомендованого часу сну	Успішно
Додавання запису про сон	Створення нового запису в базі даних	Успішно
Редагування запису	Оновлення інформації про сон	Успішно
Видалення запису	Видалення запису з бази даних	Успішно
Робота календаря	Відображення записів за обраною датою	Успішно
Відображення статистики	Побудова графіків та діаграм	Успішно
Розділ релаксації	Відображення рекомендацій та вправ	Успішно
Авторизація через JWT	Доступ лише до власних даних користувача	Успішно

Всі перевірені функціональні можливості працюють відповідно до поставлених вимог. Під час тестування не було виявлено критичних помилок, які б перешкоджали використанню вебсайту.

Особлива увага приділялася перевірці коректності взаємодії між клієнтською та серверною частинами системи. У процесі тестування було підтверджено правильність передачі даних через REST API, коректну обробку HTTP-запитів та успішне збереження інформації у базі даних MongoDB. Також було перевірено механізм авторизації на основі JWT-токенів, який забезпечує доступ користувачів лише до власних даних.

Окремо виконувалося тестування сценаріїв роботи користувача, що включали реєстрацію нового облікового запису, авторизацію в системі,

створення записів про сон, їх редагування та видалення [36]. Усі операції виконувалися коректно, а зміни своєчасно відображалися в інтерфейсі вебзастосунку без необхідності повторного завантаження сторінки.

Під час перевірки модуля статистики було підтверджено правильність формування графіків, діаграм та інформаційних карток на основі даних, внесених користувачем до щоденника сну. Також перевірялася робота калькулятора сну, який коректно формував рекомендації щодо часу засинання та пробудження залежно від заданих параметрів.

Додатково проводилося тестування інтерфейсу користувача з метою оцінки зручності навігації та коректності відображення інформації. Результати перевірки показали, що всі основні сторінки вебсайту функціонують стабільно, а користувач може безперешкодно отримувати доступ до необхідних можливостей системи.

Отримані результати свідчать про те, що розроблений вебсайт «SleepHelper» забезпечує стабільну роботу основних функціональних модулів та відповідає вимогам, сформульованим на етапі проєктування системи.

3.6 Аналіз результатів розробки та оцінка відповідності вимогам

Після завершення розробки вебсайту «SleepHelper» було проведено тестування його основних функціональних можливостей. Метою тестування була перевірка коректності роботи всіх програмних модулів, оцінка стабільності взаємодії між клієнтською та серверною частинами системи, а також підтвердження відповідності реалізованого функціоналу поставленим вимогам.

Тестування виконувалося поетапно. На першому етапі перевірялася робота модуля реєстрації та авторизації користувачів. Було протестовано створення нового облікового запису, перевірку правильності введених даних, вхід до системи та збереження авторизаційного токена. Результати перевірки

показали, що механізм аутентифікації працює коректно, а доступ до захищених сторінок надається лише авторизованим користувачам.

Наступним етапом стало тестування щоденника сну. Було перевірено створення нових записів, редагування раніше збережених даних, видалення записів та коректність їх відображення у календарі. Особлива увага приділялася перевірці автоматичного розрахунку тривалості сну та відображенню інформації за обраною датою. Під час тестування всі операції виконувалися без помилок, а зміни коректно зберігалися у базі даних MongoDB.

Окремо було перевірено роботу калькулятора сну. Тестування показало правильність формування рекомендованого часу засинання та пробудження залежно від введених параметрів користувача. Система коректно враховувала віковий діапазон, обраний режим роботи та час, введений користувачем.

Для статистичного модуля було проведено перевірку формування графіків та аналітичних показників на основі збережених записів. Під час тестування підтверджено правильність розрахунку середньої тривалості сну, визначення загальної оцінки якості сну, побудови діаграм та графіків. Усі статистичні дані відповідали інформації, що зберігалася у щоденнику користувача.

Також було протестовано роботу розділу релаксації. Перевірялася коректність відображення рекомендацій, дихальних вправ та релаксаційних матеріалів. Усі елементи сторінки працювали відповідно до проєктних вимог та забезпечували коректну взаємодію з користувачем.

Додатково виконувалася перевірка взаємодії між клієнтською та серверною частинами системи. Було протестовано передачу даних через REST API, обробку HTTP-запитів та отримання відповідей від серверної частини. У результаті тестування підтверджено коректну роботу механізму обміну даними та відсутність критичних помилок під час взаємодії компонентів системи.

Під час тестування вебсайт також перевірявся на різних розмірах вікна браузера. Результати показали, що інтерфейс коректно адаптується до зміни

роздільної здатності екрана, а всі основні функціональні елементи залишаються доступними для користувача.

3.7 Висновок до третього розділу

У третьому розділі було реалізовано програмну частину вебсайту «SleepHelper» з використанням технологій React, TypeScript, FastAPI та MongoDB. Розроблено серверну частину системи, яка забезпечує обробку запитів користувачів, реєстрацію та авторизацію, збереження даних про сон, а також взаємодію з базою даних. Також реалізовано клієнтську частину вебзастосунку у вигляді односторінкового застосунку (SPA), що забезпечує швидку та зручну навігацію між сторінками без перезавантаження браузера.

У ході роботи створено функціональні модулі для ведення щоденника сну, розрахунку рекомендованого часу засинання та пробудження, перегляду статистики, аналізу якості сну та отримання рекомендацій щодо покращення режиму відпочинку. Також було налаштовано взаємодію між клієнтською та серверною частинами за допомогою REST API та реалізовано механізм автентифікації на основі JWT-токенів для захисту персональних даних користувачів.

Проведене тестування підтвердило коректність роботи основних функцій вебсайту, зокрема реєстрації та авторизації користувачів, ведення щоденника сну, роботи калькулятора сну, формування статистики та відображення аналітичних даних. Таким чином, у результаті виконання третього розділу було створено повноцінний вебзастосунок «SleepHelper», який дозволяє користувачам вести облік власного режиму сну, аналізувати накопичені дані та отримувати інструменти для покращення якості відпочинку.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Соціальні та психологічні фактори ризику з комп’ютером

Сучасні інформаційні технології значно спрощують виконання професійних завдань, проте тривала робота за персональним комп'ютером супроводжується низкою соціальних та психологічних факторів ризику. Особливо актуальною ця проблема є для фахівців ІТ-галузі, програмістів, веброзробників, операторів комп'ютерного набору та інших працівників, діяльність яких пов'язана з постійною взаємодією з цифровими пристроями [37].

Соціальні фактори ризику виникають унаслідок особливостей організації праці та умов трудової діяльності. До них належать надмірне робоче навантаження, високий рівень відповідальності за результати роботи, дефіцит часу на виконання поставлених завдань, нерівномірний розподіл робочого часу та недостатня кількість перерв. Такі чинники можуть призводити до перевтоми, зниження працездатності та погіршення загального стану здоров'я працівників.

Психологічні фактори ризику пов'язані з емоційним та інтелектуальним навантаженням людини. Постійна концентрація уваги, необхідність прийняття рішень, обробка великих обсягів інформації та тривала робота в умовах обмеженого часу можуть викликати нервове напруження, стресові стани та емоційне виснаження. Особливо небезпечним є хронічний стрес, який здатний негативно впливати на функціонування нервової, серцево-судинної та ендокринної систем організму.

Одним із найпоширеніших наслідків психологічного перевантаження є порушення режиму сну. Недостатня тривалість або низька якість сну призводять до зниження концентрації уваги, погіршення пам'яті, зменшення продуктивності праці та підвищення ризику виникнення помилок під час виконання професійних обов'язків. Саме тому питання контролю та аналізу

якості сну є важливим елементом підтримки здоров'я працівників інформаційної сфери.

Для зниження негативного впливу соціальних та психологічних факторів ризику необхідно дотримуватися раціонального режиму праці та відпочинку, забезпечувати достатню тривалість сну, виконувати фізичні вправи, робити регулярні перерви під час роботи за комп'ютером та підтримувати сприятливий психологічний клімат у колективі [38]. Важливу роль також відіграє використання сучасних програмних засобів для моніторингу стану здоров'я та контролю режиму сну.

Розроблений у межах кваліфікаційної роботи вебзастосунок «SleepHelper» може бути використаний як інструмент для контролю режиму сну користувачів, накопичення статистики та формування корисних рекомендацій щодо покращення якості відпочинку. Використання таких систем сприяє профілактиці перевтоми, зниженню рівня стресу та підвищенню ефективності праці.

4.2 Загальні вимоги до безпеки з охорони праці для користувачів персональних комп’ютерів

Робота з персональними комп'ютерами належить до видів діяльності, які потребують дотримання встановлених вимог охорони праці. Незважаючи на відсутність значних фізичних навантажень, тривала робота за комп'ютером може негативно впливати на стан здоров'я працівника та спричиняти професійні захворювання.

Однією з основних вимог є правильна організація робочого місця користувача. Робочий стіл повинен забезпечувати достатню площу для розміщення обладнання та документів. Монітор необхідно встановлювати на відстані 50–70 см від очей користувача, а його верхній край повинен розташовуватися приблизно на рівні очей або трохи нижче [39]. Таке

розташування сприяє зменшенню навантаження на зоровий апарат та шийний відділ хребта.

Особливе значення має правильний вибір робочого крісла. Воно повинно мати регульовану висоту сидіння та спинки, забезпечувати підтримку поперекового відділу хребта та дозволяти працівнику зберігати правильну поставу протягом тривалого часу. Неправильне положення тіла може призводити до розвитку захворювань опорно-рухового апарату та виникнення хронічних болів у спині.

Важливим фактором безпечної роботи є належне освітлення робочого місця. Освітлення повинно бути достатнім для виконання робочих операцій, але не створювати відблисків на поверхні монітора. Рекомендується використовувати поєднання природного та штучного освітлення з рівномірним розподілом світлового потоку по всій площі приміщення.

Під час роботи з персональним комп'ютером необхідно дотримуватися встановленого режиму праці та відпочинку. Регулярні перерви дозволяють знизити навантаження на органи зору, м'язи шиї та спини, а також сприяють підтриманню високої працездатності. Під час перерв рекомендується виконувати вправи для очей та легку виробничу гімнастику.

Правильна організація робочого місця користувача є одним із найважливіших чинників профілактики професійних захворювань, пов'язаних із тривалою роботою за комп'ютером. Недотримання ергономічних вимог може призводити до перевтоми, погіршення зору, виникнення болю в шиї, спині та суглобах. Особливу увагу необхідно приділяти висоті розташування монітора, положенню рук під час роботи з клавіатурою та мишею, а також правильній посадці користувача. На рисунку 4.1 наведено порівняння неправильного та правильного розташування користувача за комп'ютером відповідно до вимог ергономіки.

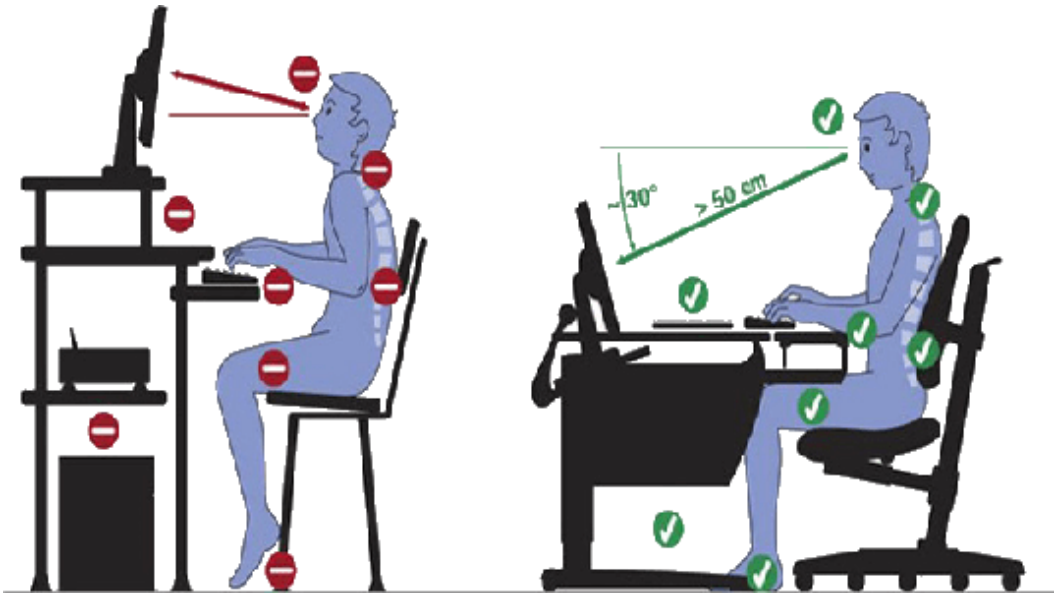

Рисунок 4.1 – Порівняння робочого місця користувача

Під час правильної організації робочого місця відстань від очей до екрана повинна становити приблизно 50–70 см, верхній край монітора має розташовуватися на рівні очей або трохи нижче, а кут згину рук і ніг повинен бути близьким до 90–100°. Спина повинна спиратися на спинку крісла, а стопи — повністю стояти на підлозі або спеціальній підставці. Дотримання цих вимог дозволяє зменшити навантаження на опорно-руховий апарат, органи зору та нервову систему, що сприяє підвищенню продуктивності праці та збереженню здоров'я користувача.

Не менш важливими є вимоги електробезпеки. Усе комп'ютерне обладнання повинно бути технічно справним, мати захисне заземлення та відповідати вимогам чинних нормативних документів. Забороняється експлуатація обладнання з пошкодженою ізоляцією проводів або несправними елементами електроживлення.

Для забезпечення комфортних умов праці необхідно також контролювати параметри мікроклімату приміщення. Температура повітря повинна підтримуватися в межах 20–24 °C, відносна вологість – 40–60 %, а швидкість руху повітря не повинна викликати дискомфорту у працівників. Дотримання

зазначених параметрів сприяє підвищенню працездатності та зменшенню втоми.

Таким чином, дотримання вимог охорони праці під час роботи з персональними комп'ютерами дозволяє мінімізувати вплив шкідливих виробничих факторів, зберегти здоров'я працівників та забезпечити високий рівень продуктивності праці.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи було розглянуто питання безпеки життєдіяльності та охорони праці, які є важливими складовими під час використання інформаційних систем та виконання професійної діяльності, пов’язаної з роботою за персональним комп’ютером. Проведений аналіз дозволив визначити основні фактори, що можуть негативно впливати на здоров’я та працездатність користувачів, а також сформувати рекомендації щодо забезпечення безпечних і комфортних умов праці.

У підрозділі, присвяченому безпеці життєдіяльності, було досліджено питання інформаційної безпеки під час використання вебзастосунків. Встановлено, що сучасні інформаційні системи працюють із великими обсягами персональних даних, тому особливо важливим є захист інформації від несанкціонованого доступу, втрати або пошкодження. Розглянуто основні загрози, які можуть виникати під час експлуатації вебзастосунку, зокрема витік конфіденційних даних, компрометацію облікових записів користувачів, помилки програмного забезпечення та ризики, пов’язані з використанням мережі Інтернет. Також проаналізовано способи мінімізації таких загроз шляхом застосування сучасних механізмів автентифікації, шифрування даних, резервного копіювання та контролю доступу до інформаційних ресурсів.

У межах підрозділу з охорони праці було розглянуто особливості організації робочого місця користувача персонального комп’ютера. Встановлено, що значна частина професійної діяльності фахівців у сфері

інформаційних технологій пов’язана з тривалою роботою за монітором, що може призводити до перевтоми, зниження концентрації уваги, погіршення зору та виникнення захворювань опорно-рухового апарату. Саме тому важливого значення набуває дотримання ергономічних вимог до робочого місця, правильне розташування обладнання, вибір якісних меблів та забезпечення оптимальних параметрів освітлення. Особливу увагу приділено рекомендаціям щодо правильного розміщення монітора, клавіатури та робочого крісла, що дозволяє зменшити навантаження на хребет, шию, руки та органи зору.

Проведений аналіз показав, що дотримання вимог охорони праці сприяє не лише збереженню здоров’я працівників, а й підвищенню ефективності виконання професійних обов’язків. Раціональна організація робочого місця, регулярні перерви під час роботи, достатній рівень освітленості приміщення та підтримання сприятливого мікроклімату дозволяють знизити рівень втоми та підвищити продуктивність праці. Крім того, виконання вимог безпеки праці допомагає запобігти розвитку професійних захворювань, які можуть виникати внаслідок тривалої роботи за комп’ютером.

У результаті виконання четвертого розділу було встановлено, що ефективне функціонування вебзастосунку «SleepHelper» повинно супроводжуватися не лише забезпеченням надійності програмного забезпечення та захисту даних користувачів, а й створенням безпечних умов праці для осіб, які працюють із системою. Комплексне врахування вимог безпеки життєдіяльності, інформаційної безпеки та охорони праці дозволяє підвищити рівень захищеності користувачів, забезпечити комфортність використання вебзастосунку та сприяти збереженню здоров’я під час роботи з інформаційними технологіями. Таким чином, розглянуті заходи є важливою складовою впровадження та подальшої експлуатації розробленої системи моніторингу та аналізу якості сну «SleepHelper».

ВИСНОВКИ

У кваліфікаційній роботі освітнього рівня «Бакалавр» виконано розробку вебзастосунку «SleepHelper», призначеного для моніторингу якості сну користувачів, ведення щоденника сну, аналізу статистичних показників та надання рекомендацій щодо покращення режиму відпочинку. Актуальність роботи обумовлена зростанням кількості людей, які стикаються з порушеннями сну, а також необхідністю використання сучасних інформаційних технологій для контролю стану здоров’я та формування корисних звичок.

У першому розділі кваліфікаційної роботи проведено аналіз предметної області та досліджено особливості організації здорового сну. Подано теоретичні відомості щодо значення сну для фізичного та психічного здоров’я людини, розглянуто основні стадії циклу сну та фактори, що впливають на його якість. Висвітлено сучасні підходи до моніторингу показників сну та використання інформаційних систем для їх аналізу. Проаналізовано існуючі програмні рішення, серед яких Sleep Cycle, BetterSleep, SleepScore та Sleep Monitor. Виконано порівняльний аналіз функціональних можливостей аналогів, визначено їх переваги та недоліки. На основі проведеного дослідження сформовано функціональні та нефункціональні вимоги до вебзастосунку «SleepHelper», а також поставлено задачу на його розробку.

У другому розділі виконано обґрунтування технологічного стеку та проєктування системи. Досліджено сучасні засоби розробки вебзастосунків та здійснено вибір програмних технологій для реалізації проєкту. Обґрунтовано використання бібліотеки React та мови TypeScript для створення клієнтської частини, а також фреймворку FastAPI для реалізації серверної логіки. Для зберігання даних обрано документоорієнтовану систему керування базами даних MongoDB. Сформовано архітектуру вебзастосунку, розроблено структуру бази даних та описано механізми взаємодії між клієнтською та серверною частинами. Також спроєктовано REST API, що забезпечує обмін

даними між компонентами системи та реалізує необхідний функціонал для роботи користувача.

У третьому розділі виконано програмну реалізацію вебзастосунку «SleepHelper». Розроблено серверну частину системи на базі FastAPI, реалізовано механізми реєстрації та авторизації користувачів із використанням JWT-токенів, а також модулі роботи з даними про сон. Створено клієнтську частину застосунку на React та TypeScript із використанням підходу Single Page Application. Реалізовано інтерактивний щоденник сну, який дозволяє створювати, редагувати та видаляти записи про сон користувача. Розроблено калькулятор часу засинання та пробудження, що формує рекомендації на основі циклів сну та вікових особливостей людини. Реалізовано модуль статистики з використанням графіків та діаграм для візуального аналізу накопичених даних. Додатково створено розділ релаксації, який містить дихальні вправи, рекомендації щодо покращення якості відпочинку та аудіоматеріали для розслаблення.

Після завершення розробки було проведено тестування функціональних можливостей вебзастосунку. Перевірено коректність роботи системи реєстрації та авторизації користувачів, функціонування щоденника сну, калькулятора сну, статистичного модуля та засобів релаксації. Результати тестування підтвердили відповідність реалізованого програмного забезпечення поставленим вимогам. Під час перевірки не було виявлено критичних помилок, які б перешкоджали використанню системи.

У розділі «Безпека життєдіяльності, основи охорони праці» розглянуто питання забезпечення безпечних умов праці під час роботи з комп’ютерною технікою. Висвітлено вимоги щодо організації робочого місця користувача, ергономічних параметрів обладнання, освітлення, мікроклімату приміщення та дотримання правил охорони праці під час роботи з персональним комп’ютером.

Таким чином, у процесі виконання кваліфікаційної роботи досягнуто поставленої мети та виконано всі визначені завдання.

ПЕРЕЛІК ДЖЕРЕЛ

1. Healthy Sleep Also called: Sleep Hygiene. URL: https://medlineplus.gov/healthysleep.html?utm_source(дата звернення: 27.01.2026).

2. The daily recommended hours of sleep you need changes as you age. URL: https://www.cdc.gov/sleep/about/index.html?utm_source(дата звернення: 28.01.2026).

3. Stages of Sleep: What Happens in a Normal Sleep Cycle. URL: https://www.sleepfoundation.org/stages-of-sleep?utm_source(дата звернення: 1.02.2026).

4. 10 Tips for a Better Night's Sleep. URL: https://www.thensf.org/sleep-tips/?utm_source (дата звернення: 3.02.2026).

5. Веб-додаток проти веб-сайту: розкриття переваг кожного рішення. URL: https://pnn.com.ua/ua/blog/detail/web-application-vs-website-uncovering-benefits-of-each-solution (дата звернення: 4.02.2026).

6. Мобільна версія: переваги, недоліки, особливості. URL: https://webakula.ua/uk/blog/mobilna-versiya-perevagi-nedoliki-osoblivosti(дата звернення: 6.02.2026).

7. Безпека. URL: https://fastapi.tiangolo.com/tutorial/security/(дата звернення: 16.02.2026).

8. Створення інтерфейсів користувача з компонентів REACT. URL https://react.dev/ (дата звернення: 18.02.2026).

9. Sleep Cycle. URL: https://sleepcycle.com/(дата звернення: 22.02.2026).

10. BetterSleep. URL:https://www.bettersleep.com/?utm_source(дата звернення: 24.02.2026).

11. SleepScore. URL: https://www.sleep.ai/sleepscore/?utm_source (дата звернення: 26.02.2026).

12. Sleep Monitor (посилання на завантаження додатку). URL: https://apps.apple.com/ru/app/sleep-monitor-sleep-tracker/id1510325113?l=uk (дата звернення: 28.02.2026).

13. What is Usability Evaluation. URL: https://ixdf.org/literature/topics/usability-evaluation?utm_source (дата звернення: 1.03.2026).

14. What is Software Requirement Specification. URL: https://www.browserstack.com/guide/software-requirement-specification?utm_source (дата звернення: 3.03.2026).

15. Розробка з боку Front end – що це таке і чим відрізняється від Back end. URL: https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaietsja-vid-back-end/ (дата звернення: 3.03.2026).

16. React. The library for web and native user interfaces . URL: https://react.dev/?utm_source (дата звернення: 5.03.2026).

17. Vue.js.The Progressive JavaScript Framework An approachable, performant and versatile framework for building web user interfaces. URL: https://vuejs.org/?utm_source (дата звернення: 7.03.2026).

18. Want to learn more about Angular. URL: https://angular.dev/?utm_source (дата звернення: 9.03.2026).

19. Svelte . Web development for the rest of us. URL: https://svelte.dev/?utm_source (дата звернення: 11.03.2026).

20. Що таке Typescript. URL: https://artjoker.ua/tech/web-stack/typescript/ (дата звернення: 13.03.2026).

21. Що таке Back-end розробка. URL: https://wezom.com.ua/ua/blog/chto-takoe-back-end-razrabotka (дата звернення: 17.03.2026).

22. FastAPI. URL: https://fastapi.tiangolo.com/?utm_source (дата звернення: 18.03.2026).

23. Django. URL: https://docs.djangoproject.com/en/6.0/?utm_source (дата звернення: 28.03.2026).

24. Flask. URL: https://flask.palletsprojects.com/en/stable/?utm_source (дата звернення: 30.03.2026).

25. MongoDB. URL: https://www.mongodb.com/docs/?utm_source (дата звернення: 1.04.2026).

26. Motor. URL: https://motor.readthedocs.io/en/stable/?utm_source (дата звернення: 1.04.2026).

27. PostgreSQL. URL: https://www.postgresql.org/?utm_source (дата звернення: 1.04.2026).

28. MySQL. URL: https://www.mysql.com/?utm_source (дата звернення: 1.04.2026).

29. MongoDB. URL: https://uk.wikipedia.org/wiki/MongoDB (дата звернення: 1.04.2026).

30. Rect API Tutorial. URL: https://restfulapi.net/?utm_source (дата звернення: 1.04.2026).

31. MDN Web Docs. HTTP request methods. URL: https://developer.mozilla.org/enUS/docs/Web/HTTP/Reference/Methods?utm_sourc (дата звернення: 1.04.2026).

32. MongoDB. URL: https://www.mongodb.com/docs/manual/tutorial/ (дата звернення: 1.04.2026).

33. Басс, Л., Клементс, П., & Казман, Р. (2018). Архітектура програмного забезпечення на практиці.

34. Uvicorn. URL: https://uvicorn.dev/ (дата звернення: 1.04.2026).

35. 35 Мельник, А., & дмитроца, Л. (2026). Методи та архітектурні підходи до автоматизації тестування мобільних і вебзастосунків. Вимірювальна та обчислювальна техніка в технологічних процесах, (2), 74-81.

36. Melnyk, A., Dmytrotsa, L., Palka, O., Vasylenko, Y., & Klymuk, N. (2025). Dynamic test case prioritisation for mobile applications based on real user behaviour data.

37. Андрейчук Н.І., Кіт Ю.В., Шибанов С.В., Шерстньова О.В. Охорона праці. Львів: Видавництво Львівська політехніка, 2021. 276 с.

38. Жидецький В.Ц. Охорона праці користувачів комп'ютерів. Львів: Афіша, 2020. 176 с.

39. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями.

ДОДАТКИ

Додаток А

Серверна частина FastAPI для авторизації користувачів та керування записами сну

```
from fastapi import FastAPI, HTTPException, Depends
from fastapi.middleware.cors import CORSMiddleware
from fastapi.security import HTTPBearer, HTTPAuthorizationCredentials
from pydantic import BaseModel
from typing import Optional, List
import motor.motor_asyncio
from dotenv import load_dotenv
import os
import bcrypt
import jwt
from datetime import datetime, timedelta

load_dotenv()


MONGO_URL  = os.getenv("MONGO_URL")
DB_NAME    = os.getenv("DB_NAME", "sleep_tracker")
JWT_SECRET = os.getenv("JWT_SECRET", "super_secret_key_change_in_production")
JWT_EXPIRE_DAYS = 7

client     = motor.motor_asyncio.AsyncIOMotorClient(MONGO_URL)
db         = client[DB_NAME]
collection = db["sleep_records"]
users_col  = db["users"]

bearer_scheme = HTTPBearer()

class SleepRecord(BaseModel):
    id: int
    date: str
    timeSleep: str
    timeWake: str
    duration: str
    mood: str
    notes: Optional[str] = None

class RegisterRequest(BaseModel):
    name: str
    email: str
    password: str

class LoginRequest(BaseModel):
    email: str
    password: str
```

```
class UserResponse(BaseModel):
    id: str
    name: str
    email: str

app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000",
"http://127.0.0.1:3000"],
    allow_methods=["*"],
    allow_headers=["*"],
    allow_credentials=True,
)

def create_token(user_id: str) -> str:
    payload = {
        "sub": user_id,
        "exp": datetime.utcnow() + timedelta(days=JWT_EXPIRE_DAYS)
    }
    return jwt.encode(payload, JWT_SECRET, algorithm="HS256")

def decode_token(token: str) -> str:
    try:
        payload = jwt.decode(token, JWT_SECRET,
algorithms=["HS256"])
        return payload["sub"]
    except jwt.ExpiredSignatureError:
        raise HTTPException(status_code=401, detail="Токен
прострочений")
    except jwt.InvalidTokenError:
        raise HTTPException(status_code=401, detail="Невалідний
токен")

async def get_current_user(credentials:
HTTPAuthorizationCredentials = Depends(bearer_scheme)) -> str:
    return decode_token(credentials.credentials)

@app.post("/api/auth/register")
async def register(data: RegisterRequest):
    existing = await users_col.find_one({"email": data.email})
    if existing:
        raise HTTPException(status_code=400, detail="Email вже
зареєстровано")

    hashed = bcrypt.hashpw(data.password.encode(),
bcrypt.gensalt()).decode()

    result = await users_col.insert_one({
        "name": data.name,
        "email": data.email,
        "password": hashed,
```

```
    })

    token = create_token(str(result.inserted_id))
    return {"token": token, "name": data.name, "email":
data.email}

@app.post("/api/auth/login")
async def login(data: LoginRequest):
    user = await users_col.find_one({"email": data.email})
    if not user:
        raise HTTPException(status_code=401, detail="Невірний
email або пароль")

    if not bcrypt.checkpw(data.password.encode(),
user["password"].encode()):
        raise HTTPException(status_code=401, detail="Невірний
email або пароль")

    token = create_token(str(user["_id"]))
    return {"token": token, "name": user["name"], "email":
user["email"]}

@app.get("/api/auth/me")
async def get_me(user_id: str = Depends(get_current_user)):
    from bson import ObjectId
    user = await users_col.find_one({"_id": ObjectId(user_id)})
    if not user:
        raise HTTPException(status_code=404, detail="Юзера не
знайдено")
    return {"id": str(user["_id"]), "name": user["name"], "email":
user["email"]}

@app.post("/api/sleep", response_model=SleepRecord)
async def create_record(record: SleepRecord, user_id: str =
Depends(get_current_user)):
    existing = await collection.find_one({"id": record.id,
"user_id": user_id})
    if existing:
        raise HTTPException(status_code=400, detail="Запис з таким
id вже існує")

    data = record.model_dump()
    data["user_id"] = user_id
    await collection.insert_one(data)
    return record

@app.get("/api/sleep", response_model=List[SleepRecord])
async def get_records(date: Optional[str] = None, user_id: str =
Depends(get_current_user)):
    query = {"user_id": user_id}
    if date:
        query["date"] = date
```

```
    records = await collection.find(query, {"_id": 0, "user_id": 0}).to_list(length=1000)
    return records

@app.put("/api/sleep/{record_id}", response_model=SleepRecord)
async def update_record(record_id: int, record: SleepRecord, user_id: str = Depends(get_current_user)):
    data = record.model_dump()
    data["user_id"] = user_id
    result = await collection.replace_one({"id": record_id, "user_id": user_id}, data)
    if result.matched_count == 0:
        raise HTTPException(status_code=404, detail="Запис не знайдено")
    return record

@app.delete("/api/sleep/{record_id}")
async def delete_record(record_id: int, user_id: str = Depends(get_current_user)):
    result = await collection.delete_one({"id": record_id, "user_id": user_id})
    if result.deleted_count == 0:
        raise HTTPException(status_code=404, detail="Запис не знайдено")
    return {"status": "deleted", "id": record_id}
```

Додаток Б

Компонент сторінки входу користувача в систему

```
import { useState } from "react";
import { useNavigate, Link } from "react-router-dom";
import { useAuth } from "../../context/AuthContext";
import styles from "./Login.module.css";
const Login = () => {
    const { login } = useAuth();
    const navigate = useNavigate();
    const [email, setEmail]       = useState("");
    const [password, setPassword] = useState("");
    const [error, setError]       = useState("");
    const [isLoading, setIsLoading] = useState(false);
    const handleSubmit = async () => {
        if (!email || !password) {
            setError("Заповніть всі поля");
            return;
        }

        setIsLoading(true);
        setError("");

        try {
            const res = await
fetch("http://127.0.0.1:8000/api/auth/login", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ email, password }),
            });

            const data = await res.json();
            if (!res.ok) {
                setError(data.detail || "Помилка входу");
                return;
            }


            login(data.token, { name: data.name, email: data.email
});
            navigate("/diary");
        } catch {
            setError("Сервер недоступний");
        } finally {
            setIsLoading(false);
        }
    };
    return (
        <div className={styles.page}>
            <div className={styles.card}>
```

```
                <h1 className={styles.title}>Вхід 🌙</h1>
                <p  className={styles.subtitle}>Раді  бачити  тебе
знову!</p>
                {error && <p className={styles.error}>{error}</p>}
                <div className={styles.field}>
                    <label className={styles.label}>Email</label>
                    <input
                        className={styles.input}
                        type="email"
                        placeholder="example@email.com"
                        value={email}
                        onChange={e => setEmail(e.target.value)}
                    />
                </div>
                <div className={styles.field}>
                    <label className={styles.label}>Пароль</label>
                    <input
                        className={styles.input}
                        type="password"
                        placeholder="••••••••"
                        value={password}
                        onChange={e                                  =>
setPassword(e.target.value)}
                    />
                </div>
                <button
                    className={styles.button}
                    onClick={handleSubmit}
                    disabled={isLoading}
                >
                    {isLoading ? "Входимо..." : "Увійти"}
                </button>
                <p className={styles.link}>
                    Ще немає акаунту?{" "}
                    <Link                              to="/register"
className={styles.linkText}>
                        Зареєструватись
                    </Link>
                </p>
            </div>
        </div>
    );
};

export default Login;
```

Додаток В

Компонент сторінки реєстрації користувача

```
import { useState } from "react";
import { useNavigate, Link } from "react-router-dom";
import { useAuth } from "../../context/AuthContext";
import styles from "./Register.module.css";
const Register = () => {
    const { login } = useAuth();
    const navigate = useNavigate();
    const [name, setName]         = useState("");
    const [email, setEmail]       = useState("");
    const [password, setPassword] = useState("");
    const [error, setError]       = useState("");
    const [isLoading, setIsLoading] = useState(false);
    const handleSubmit = async () => {
        if (!name || !email || !password) {
            setError("Заповніть всі поля");
            return;
        }
        if (password.length < 6) {
            setError("Пароль має бути мінімум 6 символів");
            return;
        }
        setIsLoading(true);
        setError("");
        try {
            const              res              =              await
fetch("http://127.0.0.1:8000/api/auth/register", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ name, email, password }),
            });
            const data = await res.json();
            if (!res.ok) {
                setError(data.detail || "Помилка реєстрації");
                return;
            }

            login(data.token, { name: data.name, email: data.email
});
            navigate("/diary");
        } catch {
            setError("Сервер недоступний");
        } finally {
            setIsLoading(false);
        }
    };
    return (
        <div className={styles.page}>
            <div className={styles.card}>
```

```
                <h1 className={styles.title}>Реєстрація 😊</h1>
                <p  className={styles.subtitle}>Почни  відстежувати
свій сон!</p>
                {error && <p className={styles.error}>{error}</p>}
                <div className={styles.field}>
                    <label className={styles.label}>Ім’я</label>
                    <input
                        className={styles.input}
                        type="text"
                        placeholder="Name"
                        value={name}
                        onChange={e => setName(e.target.value)}
                    />
                </div>
                <div className={styles.field}>
                    <label className={styles.label}>Email</label>
                    <input
                        className={styles.input}
                        type="email"
                        placeholder="example@email.com"
                        value={email}
                        onChange={e => setEmail(e.target.value)}
                    />
                </div>
                <div className={styles.field}>
                    <label className={styles.label}>Пароль</label>
                    <input
                        className={styles.input}
                        type="password"
                        placeholder="мінімум 6 символів"
                        value={password}
                        onChange={e                                  =>
setPassword(e.target.value)}
                    />
                </div>
                <button
                    className={styles.button}
                    onClick={handleSubmit}
                    disabled={isLoading}
                >
                    {isLoading      ?      "Реєструємось..."      :
"Зареєструватись"}
                </button>
                <p className={styles.link}>
                    Вже є акаунт?{" "}
                    <Link to="/login" className={styles.linkText}>
                        Увійти
                    </Link>
                </p>
            </div>
        </div>
    ); };
export default Register;
```

Додаток Г

Реалізація інтерактивного щоденника сну та роботи із записами

```
import React, { useState, useEffect, useMemo } from "react";
import { format } from 'date-fns';
import { uk } from 'date-fns/locale';
import { useNavigate } from 'react-router-dom';
import Layout from "../../component/layout/Layout";
import Header from "../../component/header/Header";
import Calendar from "../../component/Calendar/Calendar";
import WeeklyStats from "../../component/WeeklyStats/WeeklyStats";
import DreamJournal from
"../../component/DreamJournal/DreamJournal";
import Modal from "../../component/Modal/Modal";
import { useAuth } from '../../context/AuthContext';

import { SleepRecord, SleepFormData } from '../../types/sleep';
import styles from "./Diary.module.css";

const API_URL = "http://127.0.0.1:8000/api";

const calculateTotalDuration = (records: SleepRecord[]) => {
    if (records.length === 0) return "немає даних";
    let totalMinutes = 0;
    records.forEach(record => {
        if (record.duration) {
            const [hours, minutes] =
record.duration.split(':').map(Number);
            totalMinutes += (hours * 60) + minutes;
        }
    });
    const h = Math.floor(totalMinutes / 60);
    const m = totalMinutes % 60;
    return `${h}:${m < 10 ? '0' + m : m}`;
};

const getDurationInMinutes = (durationStr?: string) => {
    if (!durationStr) return 0;
    const [h, m] = durationStr.split(':').map(Number);
    return (h * 60) + m;
};

export default function Diary() {
    const navigate = useNavigate();
    const { token, logout } = useAuth();
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [selectedRecord, setSelectedRecord] =
useState<SleepRecord | null>(null);
    const [selectedDate, setSelectedDate] = useState<Date>(new
Date());
```

```
    const [dayRecords, setDayRecords] = useState<SleepRecord[]>([]);
    const [allRecords, setAllRecords] = useState<SleepRecord[]>([]);
    const [isLoading, setIsLoading] = useState(true);

    const authHeaders: HeadersInit | undefined = token ? { Authorization: `Bearer ${token}` } : undefined;

    const parseErrorMessage = async (res: Response) => {
        const text = await res.text();
        try {
            const data = JSON.parse(text);
            return data.detail || data.message || text || res.statusText;
        } catch {
            return text || res.statusText;
        }
    };

    const handleAuthFailure = (res: Response) => {
        if (res.status === 401) {
            alert("Токен недійсний. Будь ласка, увійдіть повторно.");
            logout();
            navigate('/login');
            return true;
        }
        return false;
    };


    useEffect(() => {
        if (!token) return;

        const fetchAll = async () => {
            setIsLoading(true);
            try {
                const res = await fetch(`${API_URL}/sleep`, {
                    headers: { Authorization: `Bearer ${token}` },
                });
                if (res.ok) {
                    const data: SleepRecord[] = await res.json();
                    setAllRecords(data);
                } else {
                    console.error("Помилка завантаження з сервера", res.status);
                }
            } catch (e) {
                console.error("Сервер недоступний:", e);
            } finally {
                setIsLoading(false);
            }
```

```
        };
        fetchAll();
    }, [token]);


    useEffect(() => {
        const formattedDate = format(selectedDate, 'yyyy-MM-dd');
        const found = allRecords.filter(r => r.date ===
formattedDate);
        setDayRecords(found);
    }, [selectedDate, allRecords]);


    const sleepMoodMap = useMemo(() => {
        const map: Record<string, { mood: string; duration: number
}> = {};
        allRecords.forEach(record => {
            const currentDuration =
getDurationInMinutes(record.duration);
            if (!map[record.date] || currentDuration >
map[record.date].duration) {
                map[record.date] = { mood: record.mood, duration:
currentDuration };
            }
        });
        const finalMap: Record<string, string> = {};
        for (const date in map) {
            finalMap[date] = map[date].mood;
        }
        return finalMap;
    }, [allRecords]);


    const handleOpenAddModal = () => {
        setSelectedRecord(null);
        setIsModalOpen(true);
    };
    const handleSave = async (formData: SleepFormData) => {
        if (!token) {
            alert("Будь ласка, увійдіть повторно, щоб зберегти
запис.");
            return;
        }

        const record: SleepRecord = {
            ...formData,
            id: Date.now(),
            date: format(selectedDate, 'yyyy-MM-dd'),
        };

        try {
            const res = await fetch(`${API_URL}/sleep`, {
                method: "POST",
```

```
                headers: {
                    "Content-Type": "application/json",
                    ...(authHeaders || {}),
                },
                body: JSON.stringify(record),
            });

            if (await handleAuthFailure(res)) return;

            const text = await res.text();
            if (res.ok) {
                const saved: SleepRecord = JSON.parse(text || '{}');
                setAllRecords(prev => [...prev, saved]);
                setIsModalOpen(false);
            } else {
                const message = await parseErrorMessage(res);
                alert(`Помилка сервера при збереженні: ${message}`);
                console.error("Save failed:", res.status, message);
            }
        } catch (e) {
            alert("Помилка мережі. Перевірте чи запущений сервер.");
            console.error(e);
        }
    };


    const handleUpdate = async (formData: SleepFormData) => {
        if (!selectedRecord) return;
        if (!token) {
            alert("Будь ласка, увійдіть повторно, щоб оновити запис.");
            return;
        }

        const updated: SleepRecord = {
            ...formData,
            id: selectedRecord.id,
            date: selectedRecord.date,
        };

        try {
            const res = await fetch(`${API_URL}/sleep/${updated.id}`, {
                method: "PUT",
                headers: {
                    "Content-Type": "application/json",
                    ...(authHeaders || {}),
                },
                body: JSON.stringify(updated),
```

```
            });

            if (await handleAuthFailure(res)) return;

            if (res.ok) {
                setAllRecords(prev => prev.map(r => r.id ===
updated.id ? updated : r));
                setIsModalOpen(false);
            } else {
                const message = await parseErrorMessage(res);
                alert(`Помилка сервера при оновленні:
${message}`);
                console.error("Update failed:", res.status,
message);
            }
        } catch (e) {
            alert("Помилка мережі. Перевірте чи запущений
сервер.");
            console.error(e);
        }
    };


    const handleDelete = async () => {
        if (!selectedRecord) return;
        if (!token) {
            alert("Будь ласка, увійдіть повторно, щоб видалити
запис.");
            return;
        }

        try {
            const res = await
fetch(`${API_URL}/sleep/${selectedRecord.id}`, {
                method: "DELETE",
                headers: authHeaders,
            });

            if (await handleAuthFailure(res)) return;

            if (res.ok) {
                setAllRecords(prev => prev.filter(r => r.id !==
selectedRecord.id));
                setIsModalOpen(false);
            } else {
                const message = await parseErrorMessage(res);
                alert(`Помилка сервера при видаленні:
${message}`);
                console.error("Delete failed:", res.status,
message);
            }
        } catch (e) {
```

```
            alert("Помилка мережі. Перевірте чи запущений
сервер.");
            console.error(e);
        }
    };

    return (
        <>
            <Header />
            <Layout>
                <section className={styles.diarySection}>
                    <section className={styles.calendarSection}>
                        <Calendar
                            selectedDate={selectedDate}
                            onDateSelect={setSelectedDate}
                            sleepMoodMap={sleepMoodMap}
                        />
                    </section>

                    <div className={styles.diaryEntries}>
                        <section
className={styles.dayOverviewSection}>
                            <h1 className={styles.day}>
                                {format(selectedDate, 'd MMMM,
yyyy', { locale: uk })}
                            </h1>
                            <div className={styles.sleepInfo}>
                                <p
className={styles.durationSleep}>
                                    Загальна тривалість сну:
{calculateTotalDuration(dayRecords)}
                                </p>
                            </div>
                        </section>

                        <section
className={styles.addButtonSection}>
                            <button
className={styles.addRecordBtn} onClick={handleOpenAddModal}>
                                Додати запис
                            </button>
                        </section>

                        <section
className={styles.recordsSection}>
                            {isLoading ? (
                                <p>Завантаження...</p>
                            ) : dayRecords.length > 0 ? (
                                dayRecords.map((record) => (
                                    <DreamJournal
                                        key={record.id}

startTime={record.timeSleep}
```

```
                                        endTime={record.timeWake}
                                        sleepQuality={record.mood}
                                        description={record.notes}
                                        onClick={() => {
setSelectedRecord(record);
                                            setIsModalOpen(true);
                                        }}
                                    />
                                ))
                            ) : (
                                <DreamJournal
                                    startTime=""
                                    endTime=""
                                    sleepQuality=""
                                    onClick={handleOpenAddModal}
                                />
                            )}
                        </section>
                    </div>

                    <section
className={styles.WeeklyStatsSection}>
                        <WeeklyStats records={allRecords} />
                    </section>

                    <Modal
                        isOpen={isModalOpen}
                        onClose={() => setIsModalOpen(false)}
                        title={selectedRecord ? "Редагувати запис
📝" : "Додати запис 🌙"}
                        initialData={selectedRecord}
                        onSave={handleSave}
                        onUpdate={handleUpdate}
                        onDelete={handleDelete}
                    />
                </section>
            </Layout>
        </>
    );
}
```

Додаток Ґ

Реалізація модуля калькулятора сну

```
import React from "react";
import style from "./CalculatorResult.module.css";
import FiledResult from "./FieldResult"
type Props = {
  ageRange: string;
  mode: "wake" | "sleep";
  time: string;
  onBack: () => void;
};
export default function SleepResult({ ageRange, mode, time, onBack
}: Props) {

type AgeRanges = {
  label: string;
  maxCycle: number;
};
  const ranges: AgeRanges[] = [
  { label: "0-3", maxCycle: 11 },
  { label: "4-11", maxCycle: 10 },
  { label: "1-2", maxCycle: 9 },
  { label: "3-5", maxCycle: 8 },
  { label: "6-13", maxCycle: 7 },
  { label: "14-17", maxCycle: 6 },
  { label: "18-25", maxCycle: 6 },
  { label: "26-35", maxCycle: 6 },
  { label: "36-45", maxCycle: 6 },
  { label: "46-55", maxCycle: 6},
  { label: "56-64", maxCycle: 6 },
  { label: "65+", maxCycle: 6 },
];
    const cycleDuration= 90; // minutes
    const fallAsleepTime= 15; // minutes
    const [hours, minutes] =  time.split(':').map(Number);
    const   maxCycle   =   ranges.find(range   =>   range.label   ===
ageRange)?.maxCycle || 6;
    const recomandTimes: string[] = [];
    const generalTimes: string[] = [];
    let totalMinutes: number;
    for (let i = maxCycle; i > (maxCycle-6); i--) {
        if (mode === "wake") {
           totalMinutes = hours * 60 + minutes - fallAsleepTime -
i * cycleDuration;
          if (totalMinutes < 0) {
            totalMinutes += 24 * 60;
          }
        } else {
           totalMinutes = hours * 60 + minutes + fallAsleepTime +
i * cycleDuration;
```

```
          if (totalMinutes >= 24 * 60) {
            totalMinutes -= 24 * 60;
          }
    }
          const recHours = Math.floor(totalMinutes / 60);
          const recMinutes = totalMinutes % 60;
          recomandTimes.push(
            `${recHours.toString().padStart(2,
"0")}:${recMinutes.toString().padStart(2, "0")}`
          );
         generalTimes.push(
            `${Math.floor(i * 1.5 + 0.25).toString().padStart(2,
"0")}:${((i *90 + 15)% 60).toString().padStart(2, "0")}`
          );
  }

  return (
    <>
      <h2>Розрахунок враховує 15 хвилин на засинання</h2>
     <p className={style.headline}>Мінімальний час сну</p>
      <section className={style.filedBox}>
          <FiledResult          recomandTime={`${recomandTimes[5]}`}
generalTime={`${generalTimes[5]} годин `}/>
          <FiledResult          recomandTime={`${recomandTimes[4]}`}
generalTime={`${generalTimes[4]} годин ` }/>
          <FiledResult          recomandTime={`${recomandTimes[3]}`}
generalTime={`${generalTimes[3]} годин `} />
      </section>
     <p     className={style.headline}>Рекомендований     (90     хв
цикли)</p>
       <section className={style.filedBox}>
          <FiledResult          recomandTime={`${recomandTimes[2]}`}
generalTime={`${generalTimes[2]} годин `}/>
          <FiledResult          recomandTime={`${recomandTimes[1]}`}
generalTime={`${generalTimes[1]} годин `} />
          <FiledResult          recomandTime={`${recomandTimes[0]}`}
generalTime={`${generalTimes[0]} годин `} />
       </section>
     <p      className={style.headline}>Рекомендований      (аналіз
записів)</p>
       <section className={style.filedBox}>
          <FiledResult   recomandTime="в   процесі"   generalTime="в
процесі" />
          <FiledResult   recomandTime="в   процесі"   generalTime="в
процесі" />
          <FiledResult   recomandTime="в   процесі"   generalTime="в
процесі" />
       </section>
       <section className={style.goBackSection}>
        <button className={style.goBackBtn} onClick={onBack}>
          ← Повернутися назад
        </button>
       </section> </> );}
```