ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розроблення системи миттєвої підтримки користувачів на основі PHP та Ajax»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Денис САВОНЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр. КНЗ-51
Денис САВОНЧУК

Керівник: Сергій ІЩЕРЯКОВ
науковий ступінь, вчене звання: д.ф.-м.н., професор

Рецензент:
науковий ступінь, вчене звання

_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Савончуку Денису Володимировичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розроблення системи миттєвої підтримки користувачів на основі PHP та Ajax

керівник кваліфікаційної роботи Сергій ІЩЕРЯКОВ к.т.н., доцент,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, мова програмування PHP, технологія Ajax, СУБД SQLite, фреймворки.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Огляд існуючих систем миттєвої підтримки.
Вибір напрямків для розробки системи на основі аналізу аналогів.
Розробка системи миттєвої підтримки.
Оптимізація та тестування ефективності системи миттєвої підтримки.
5. Перелік графічного матеріалу: *презентація*

1. Інтерфейси популярних систем підтримки користувачів.
2. Приклад архітектури спілкувального інтерфейсу.
3. Діаграма функціоналу обробки повідомлень.
4. Діаграма роботи системи миттєвої підтримки.
5. Файлова структура додатку.
6. Екранні форми системи миттєвої підтримки.
7. Фрагменти програмного коду.

6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	27.02-09.03.24	
2	Огляд та аналіз існуючих систем миттєвої підтримки	09.03-16.03.24	
3	Вибір напрямків для розробки системи на основі аналізу аналогів	17.03-23.03.24	
4	Вибір інструментвльних засобів для розробки системи	24.03-30.03.24	
5	Розробка системи миттєвої підтримки	01.04-14.04.24	
6	Оптимізація та тестування ефективності системи миттєвої підтримки	15.04-24.04.24	
7	Оформлення роботи: вступ, висновки, реферат	25.04-02.05.24	
8	Розробка демонстраційних матеріалів	03.05-08.05.24	

Здобувач вищої освіти _____ Денис САВОНЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій ІЩЕРЯКОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 5 табл., 19 рис., 18 джерел.

Мета роботи – розробка системи миттєвої підтримки користувачів з використанням сучасних веб-технологій, дослідження можливостей впровадження системи, її функціональних можливостей та переваг перед існуючими рішеннями.

Об'єкт дослідження – процеси функціонування та взаємодії користувачів з системою .

Предмет дослідження – розробка системи миттєвої підтримки користувачів на основі мови програмування PHP та технології Ajax.

Короткий зміст роботи:

Проведено огляд і аналіз існуючих систем миттєвої підтримки, аналіз потреб та вимог користувачів щодо миттєвої підтримки. В результаті аналізу були визначені ключові функції та можливості, які мали б бути доступні в системі.

Спроектована та розроблена система миттєвої підтримки, яка базується на мові програмування PHP та використанні технології Ajax для забезпечення швидкості та інтерактивності. Використання Ajax дозволяє реалізувати взаємодію з сервером без перезавантаження сторінки, що забезпечує плавність та швидкість роботи системи.

Крім того, розглядається питання тестування та оптимізації роботи системи для досягнення найвищої ефективності та задоволення потреб користувачів. Результати даної роботи будуть корисні для компаній, які надають онлайн-сервіси та продукти, оскільки ефективна система миттєвої підтримки дозволить підвищити задоволеність користувачів та зменшити час вирішення їхніх проблем.

КЛЮЧОВІ СЛОВА: СИСТЕМА МИТТЄВОЇ ПІДТРИМКИ, АНАЛІЗ ВИМОГ, РОЗРОБКА СИСТЕМИ, PHP, JAVASCRIPT, AJAX, СУБД SQLITE, ТЕСТУВАННЯ СИСТЕМИ.

ABSTRACT

Text part of the bachelor's qualification work: 53 pages, 5 tables, 19 pictures, 18 sources.

The purpose of the work is to develop a system of instant user support using modern web technologies, to study the possibilities of implementing the system, its functionality and advantages over existing solutions.

Object of research – the processes of functioning and interaction of users with the system.

Subject of the research – the development of an instant user support system based on the PHP programming language and Ajax technology.

Summary of the work: A review and analysis of existing instant support systems, analysis of users' needs and requirements for instant support was conducted. As a result of the analysis, the key functions and capabilities that should be available in the system were determined.

Designed and developed an instant support system based on the PHP programming language and the use of Ajax technology to ensure speed and interactivity. The use of Ajax allows you to interact with the server without reloading the page, which ensures the smoothness and speed of the system.

In addition, the issue of testing and optimizing the system's operation to achieve the highest efficiency and meet the needs of users is considered. The results of this work will be useful for companies that provide online services and products, as an effective instant support system will increase user satisfaction and reduce the time it takes to solve their problems.

KEYWORDS: INSTANT SUPPORT SYSTEM, REQUIREMENTS ANALYSIS, SYSTEM DEVELOPMENT, PHP, JAVASCRIPT, AJAX, SQLITE DBMS, SYSTEM TESTING.

ЗМІСТ

ВСТУП

З розвитком інтернет-технологій та зростанням конкуренції сучасні компанії акцентують увагу на забезпеченні високоякісної та оперативної підтримки своїх клієнтів в онлайн-середовищі.. В цьому контексті стає надзвичайно важливим та актуальним створення ефективних механізмів миттєвої підтримки, які спроможні негайно реагувати на запити користувачів та забезпечувати вирішення їхніх проблем.

Дипломна робота присвячена розробленню та впровадженню системи миттєвої підтримки користувачів, яка базується на мові програмування PHP та використанні технології Ajax для забезпечення швидкості та інтерактивності.

Перед початком розробки системи проведено огляд і аналіз існуючих систем миттєвої підтримки, аналіз потреб та вимог користувачів щодо миттєвої підтримки. Особлива увага була приділена функціоналу, що найбільше цікавить користувачів та спрощує їхню взаємодію з системою. В результаті аналізу були визначені ключові функції та можливості, які мали б бути доступні в системі.

Окрім того, робота обґрунтовує вибір мови програмування PHP та технології Ajax для реалізації проекту. PHP відомий своєю широкою популярністю та можливістю швидкого розгортання веб-додатків. Використання Ajax дозволяє реалізувати взаємодію з сервером без перезавантаження сторінки, що забезпечує плавність та швидкість роботи системи.

Ця робота детально описує процес проектування, розробки та впровадження системи миттєвої підтримки користувачів на основі вищезгаданих технологій. Крім того, розглядається питання тестування та оптимізації роботи системи для досягнення найвищої ефективності та задоволення потреб користувачів.

Предметом дослідження є розробка системи миттєвої підтримки користувачів на основі мови програмування PHP та технології Ajax. Об'єктом дослідження є процеси функціонування та взаємодії користувачів з даною системою.

Метою даної роботи є розробка системи миттєвої підтримки користувачів з використанням сучасних веб-технологій. Досліджуються можливості впровадження системи, її функціональні можливості та переваги перед існуючими рішеннями.

Для досягнення цієї мети передбачено наступні завдання:

- Огляд та аналіз існуючих систем миттєвої підтримки
- Аналіз вимог користувачів щодо системи миттєвої підтримки.
- Проектування архітектури системи та її компонентів з урахуванням потреб користувачів.
- Розроблення програмного забезпечення на базі PHP та Ajax з метою забезпечення ефективності та зручності використання.
- Тестування та впровадження системи для перевірки її працездатності та коректності функціонування.

Наукова новизна роботи полягає у використанні сучасних веб-технологій для розробки системи миттєвої підтримки користувачів, а також у вивченні та аналізі їхнього застосування в контексті вирішення практичних завдань бізнесу. Результати даної роботи будуть корисні для компаній, які надають онлайн-сервіси та продукти, оскільки ефективна система миттєвої підтримки дозволить підвищити задоволеність користувачів та зменшити час вирішення їхніх проблем.

1 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ ТА ВИМОГ ДЛЯ СИСТЕМИ

1.1 Дослідження існуючих систем миттєвої підтримки

У процесі розроблення системи миттєвої підтримки на основі PHP та Ajax важливим етапом є аналіз існуючих рішень на ринку. Це дозволить ідентифікувати сильні та слабкі сторони конкурентів та виявити потенційні можливості для інновацій. Огляд кількох популярних систем підтримки користувачів надасть змогу глибше зрозуміти вимоги до сучасних сервісів обслуговування клієнтів та визначити ключові напрямки для розвитку власної системи.

LiveChat – це одна з відомих платформ для надання живої підтримки, яка інтегрується з різними вебсайтами та дозволяє операторам ефективно комунікувати з користувачами у реальному часі. Її переваги полягають у швидкості відгуку, високому рівні налаштування інтерфейсу та можливості інтеграції з іншими сервісами, такими як CRM-системи. Однак, сервіс може бути досить дорогим для малого та середнього бізнесу, і не завжди гарантує повноцінне рішення для складних задач обслуговування (рис. 1.1).

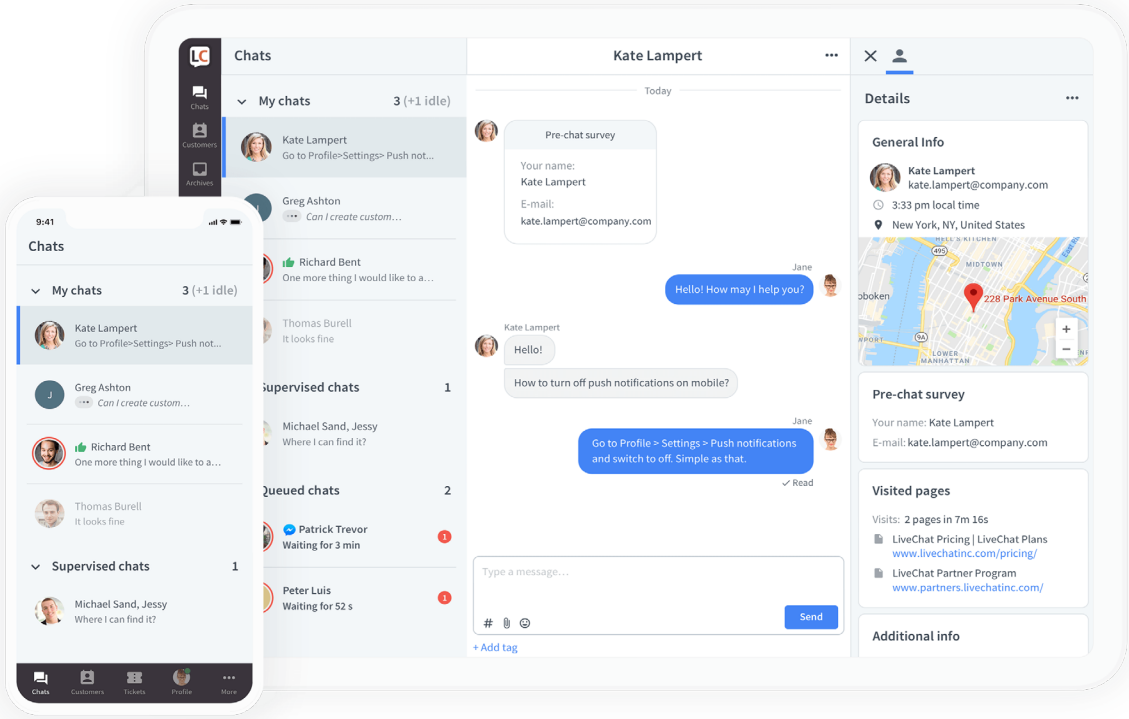


Рисунок 1.1 – Система Life Chat

Zendesk Chat – ще один популярний інструмент, який забезпечує широкий спектр функціональності для комунікації з користувачами. Особливостями Zendesk Chat є її гнучкість і масштабованість, яка дозволяє використовувати її як для невеликих так і для великих компаній. Також, сервіс надає розширену аналітику та звітність, що допомагає краще розуміти потреби клієнтів. Тим не менше, інтерфейс може здатися складним для нових користувачів, а ціна – високою для стартапів (рис. 1.2).

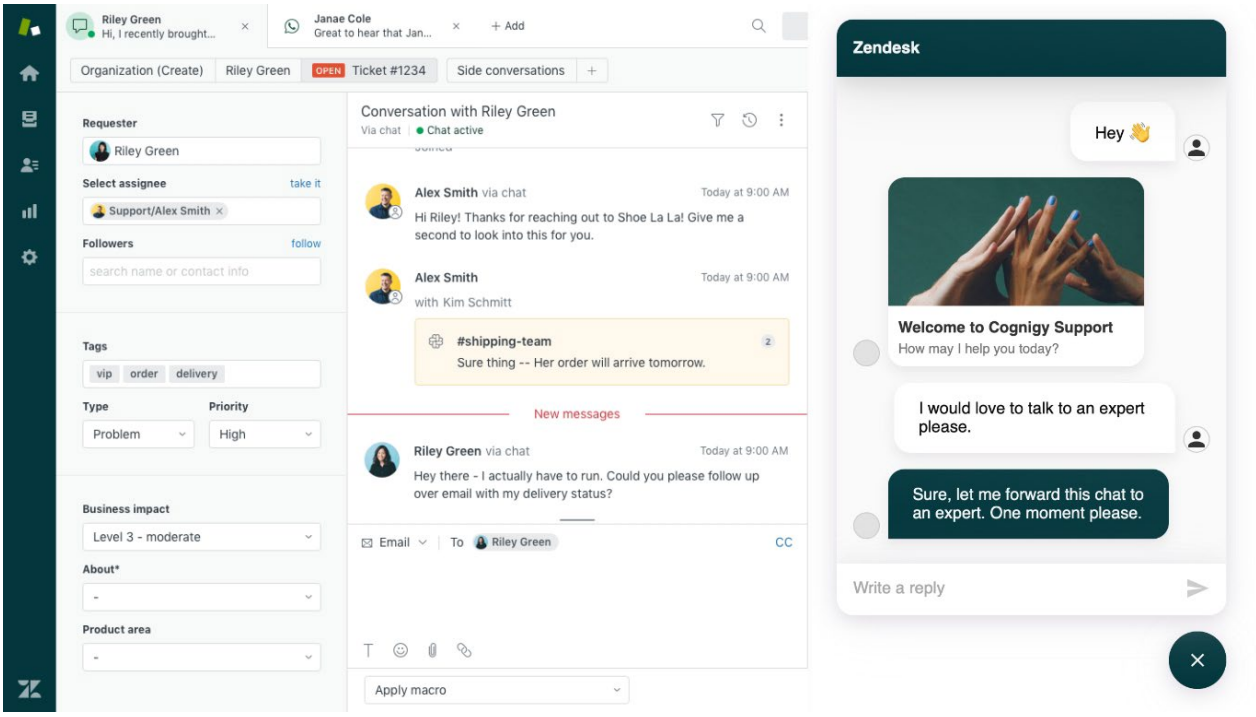


Рисунок 1.2 – Zendesk Chat

Intercom– інноваційна платформа, яка об'єднує в собі елементи живого чату, маркетингу та служби підтримки. Її унікальною особливістю є використання штучного інтелекту для автоматизації відповідей та сегментації користувачів, що дозволяє надавати більш персоналізований досвід. Однак, Intercom також може виявитися занадто складним у налаштуванні та вимагати значних інвестицій для повного розгортання та інтеграції з вже існуючими системами підприємства (рис. 1.3).

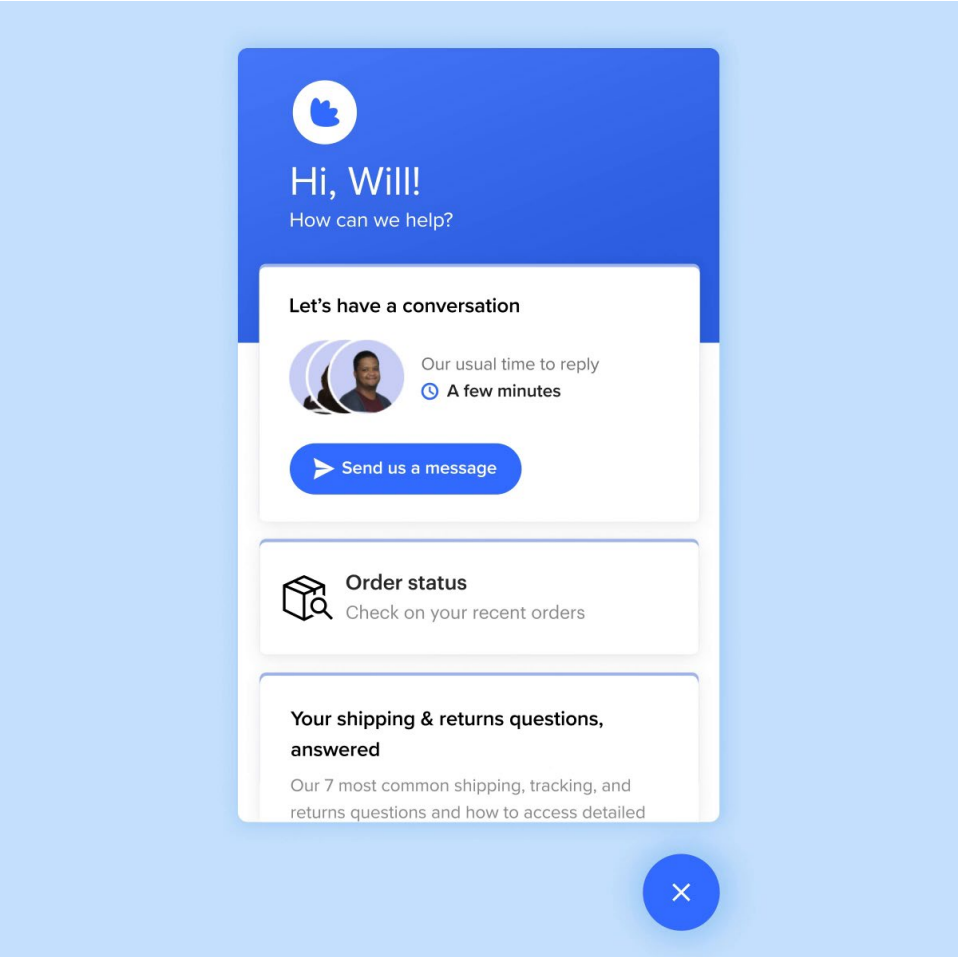


Рисунок 1.3 – Платформа Intercom

Drift — це платформа для залучення відвідувачів сайту, яка використовує "розумні" чат-боти, що можуть ініціювати розмову з потенційними клієнтами. Це допомагає збільшити конверсію та поліпшити досвід користувачів завдяки швидкому та ефективному спілкуванню. Drift пропонує також широкі можливості для персоналізації діалогів залежно від поведінки користувачів на сайті. Основними недоліками можуть бути висока вартість підписки та складність налаштування деталізованих сценаріїв для чат-ботів (рис. 1.4).

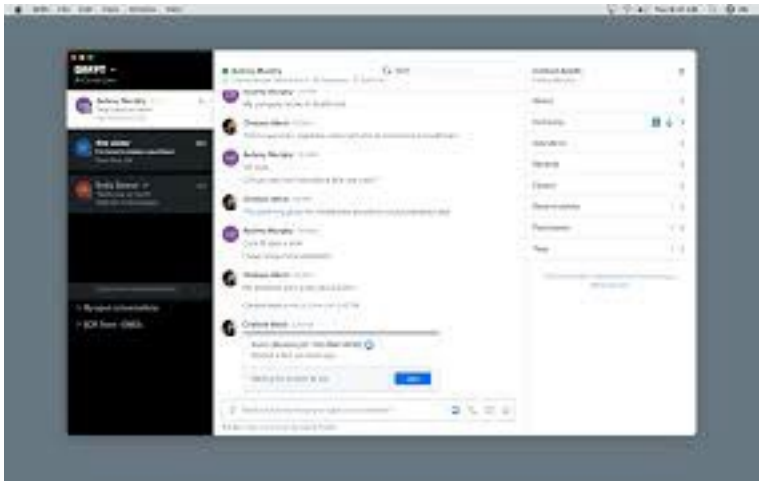


Рисунок 1.4 – Платформа Drift

Tidio Chat — інструмент, який дозволяє інтегрувати на сайті функцію живого чату з можливістю додавання чат-ботів. Завдяки гнучким налаштуванням і простоті інтеграції, Tidio став популярним серед малих та середніх підприємств. Він також пропонує базовий безкоштовний план, що робить його доступним для стартапів. Однак, розширені функції вимагають платної підписки, і можливості штучного інтелекту та автоматизації можуть бути обмежені в порівнянні з більш дорогими платформами (рис. 1.5).

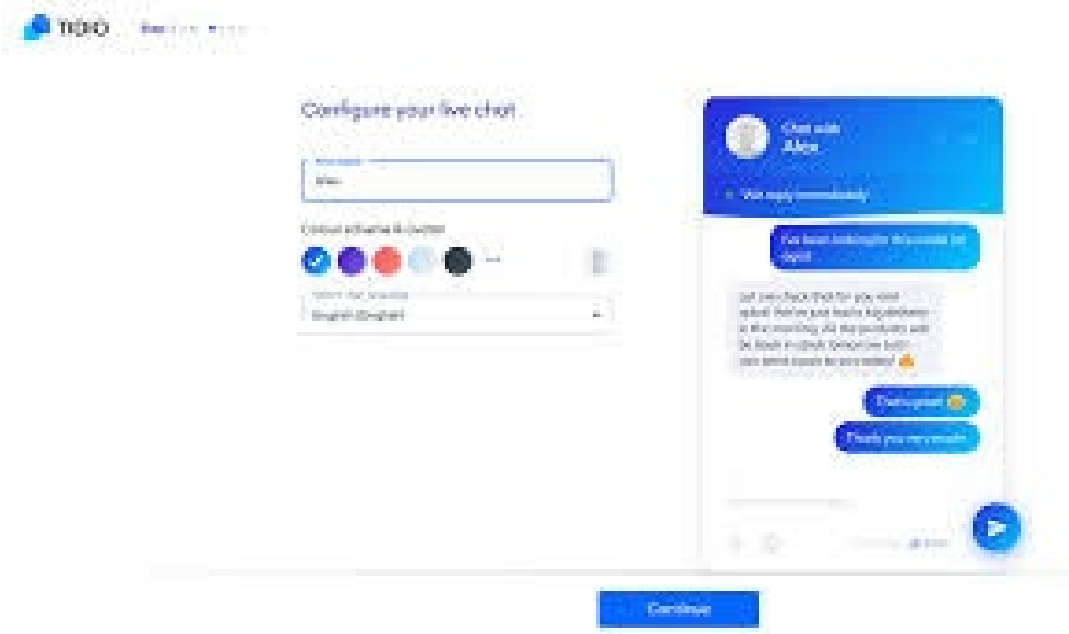


Рисунок 1.5 – Платформа Tidio

У таблиці 1.1 представлене порівняння основних критеріїв, за якими аналізувалися існуючі системи миттєвої підтримки. Ці критерії дозволяють зрозуміти сильні сторони кожного рішення та потенційні області для вдосконалення.

Таблиця 1.1 – Порівняння платформ

Критерій	**LiveChat**	**Zendesk Chat**	**Intercom**	**Drift**	**Tidio Chat**
Швидкість відгуку	Висока	Висока	Висока	Висока	Середня
Рівень налаштування	Високий	Високий	Високий	Високий	Середній
Інтеграція з CRM	Так	Так	Так	Так	Ні
Ціна	Висока для SMB	Висока	Висока	Дуже висока	Доступна
Складність налаштування	Низька	Складна для новачків	Висока	Складна	Низька
Інноваційні функції	Ні	Ні	Так	Так	Ні
Персоналізація діалогів	Так	Так	Так	Так	Обмежена
Автоматизація та AI	Обмежена	Обмежена	Висока	Висока	Обмежена
Гнучкість і масштабованість	Висока	Дуже висока	Висока	Висока	Середня
Простота інтеграції	Середня	Середня	Складна	Складна	Висока

На основі цієї таблиці можна зробити висновок, що системи миттєвої підтримки різняться за рядом параметрів, таких як швидкість відгуку, наявність інноваційних функцій, і можливостями інтеграції. На сьогоднішній день значення мають не лише базові функції, але й додаткові можливості, такі як персоналізація і автоматизація обслуговування, які покращують загальний досвід взаємодії з клієнтом.

І хоча деякі платформи пропонують розширені функції штучного інтелекту, вони можуть бути складними у налаштуванні та інтеграції, що створює бар'єри для впровадження, особливо для малих та середніх підприємств. Ціновий фактор також відіграє важливу роль, особливо для стартапів та SMB, для яких доступність рішень є критичною.

Розглядаючи ці існуючі рішення, можна зробити висновок, що ключ до успіху сучасної системи миттєвої підтримки полягає у збалансуванні між потужними функціональними можливостями та простотою використання. Інвестиції у

розвиток штучного інтелекту, машинного навчання та автоматизованих відповідей можуть значно покращити ефективність системи, але необхідно також звернути увагу на вартість інструментів та їхню доступність для різних сегментів ринку. У своєму розробленні слід враховувати ці аспекти, а також створити рішення, яке буде легко інтегрувати і масштабувати в залежності від потреб бізнесу.

1.2 Вибір напрямків для розробки системи на основі аналізу аналогів

Проведення детального аналізу існуючих систем миттєвої підтримки дозволяє визначити напрямки, на які слід зосередитися під час розробки нової системи. Основні аспекти, які необхідно врахувати, включають в себе інтеграційну сумісність, вартість впровадження, легкість використання, можливість масштабування та впровадження інноваційних технологій.

Інтеграційна сумісність – Система має легко інтегруватися з різними CRM-системами та іншими інструментами, які вже використовуються на підприємстві. Це дозволить забезпечити безперебійний потік інформації та уникнути дублювання даних.

Вартість впровадження – Важливо, щоб система була доступною для широкого спектру бізнесів, включаючи малі та середні підприємства. Передбачення гнучких тарифних планів та безкоштовного доступу до базових функцій може збільшити привабливість продукту на ринку.

Легкість використання – Інтуїтивно зрозумілий інтерфейс, що не вимагає складного навчання для операторів та кінцевих користувачів, є ключовим для швидкого впровадження та прийняття системи споживачами.

Можливість масштабування – Система повинна бути гнучкою, щоб витримувати збільшення обсягів запитів та користувачів без втрати продуктивності. Це дозволить бізнесу рости без необхідності частих оновлень системи.

Впровадження інновацій – Використання штучного інтелекту для автоматизації відповідей та аналітики даних дозволить не тільки знизити

навантаження на операторів, але й забезпечити більш персоналізований підхід до кожного клієнта.

Безпека та конфіденційність – З огляду на зростаючу увагу до питань захисту персональних даних, система має включати надійні засоби захисту інформації та відповідати вимогам регуляторів.

На основі цих напрямків можна скласти технічне завдання для розробки системи, яка відповідатиме сучасним вимогам ринку та запитам користувачів. Розробка має бути спрямована на створення продукту, що пропонує зручне і ефективне вирішення проблем клієнтів, при цьому забезпечуючи високу якість обслуговування та унікальність користувацького досвіду.

Таким чином, стратегія розробки включатиме створення модульної платформи з можливістю легкої інтеграції у вже існуючі системи бізнесу, фокусування на оптимізації вартості для забезпечення доступності для всіх категорій бізнесів, а також інтеграції передових технологій для поліпшення якості обслуговування та персоналізації спілкування.

Водночас, особлива увага має бути приділена впровадженню інструментів штучного інтелекту, що дозволять підвищити швидкість та якість відповідей системи, автоматизувати рутинні запити та надавати операторам більш складні випадки, що вимагають особистого втручання. Застосування машинного навчання може допомогти системі «навчатися» на основі попередніх запитів і відповідей, щоб з кожним разом надавати все більш точні та ефективні рішення.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

2.1 Основні складові системи миттєвої підтримки

Системи миттєвої підтримки є ключовим елементом сучасних веб-додатків та інтернет-сервісів, надаючи користувачам можливість отримувати оперативну допомогу та вирішувати проблеми в режимі реального часу [1]. Вони часто відомі як чат-системи або онлайн-підтримка і забезпечують механізм зв'язку між користувачами та командою підтримки за допомогою текстового чи мультимедійного обміну повідомленнями.

Основні складові системи миттєвої підтримки включають в себе:

Спілкувальний інтерфейс

Спілкувальний інтерфейс – це ключовий елемент системи миттєвої підтримки, який використовується для взаємодії між користувачем та підтримкою [17]. Його належне проектування є важливим для забезпечення максимального комфорту та зручності для кінцевого користувача (рис. 2.1).

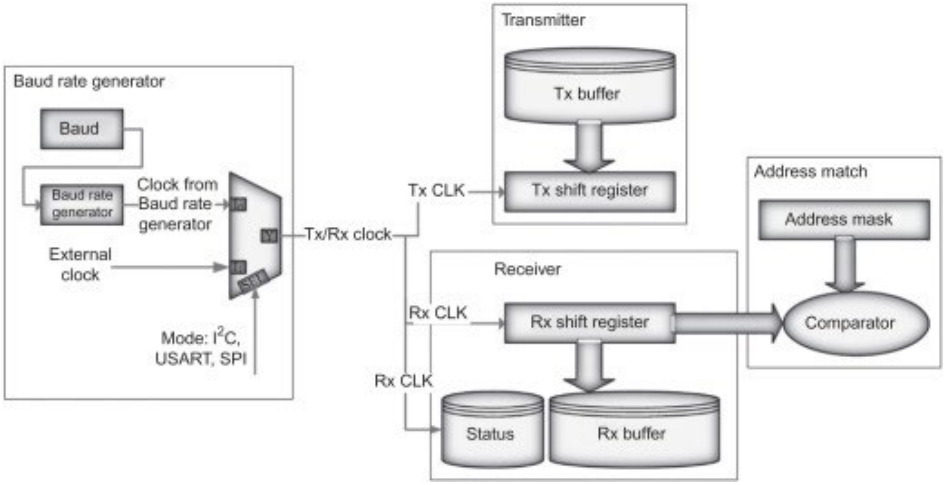


Рисунок 2.1 – Приклад архітектури спілкувального інтерфейсу

Спілкувальний інтерфейс може містити різноманітні функції, що полегшують комунікацію, такі як автозаповнення, яке допомагає користувачам швидше вибирати варіанти або заповнювати форми, емодзі для вираження емоцій,

можливість прикріплення файлів для обміну додатковою інформацією та функції швидкого доступу до основних опцій чату.

При проектуванні спілкувального інтерфейсу важливо враховувати потреби та особливості цільової аудиторії, щоб забезпечити максимальну ефективність та задоволення від використання.

Функціонал обробки повідомлень

Функціонал обробки повідомлень в системі миттєвої підтримки відіграє важливу роль у забезпеченні ефективної комунікації між користувачами та підтримкою [2]. Цей компонент відповідає за аналіз та опрацювання повідомлень, що надходять від користувачів, з метою забезпечення оперативної відповіді та вирішення їхніх проблем (рис. 2.2).

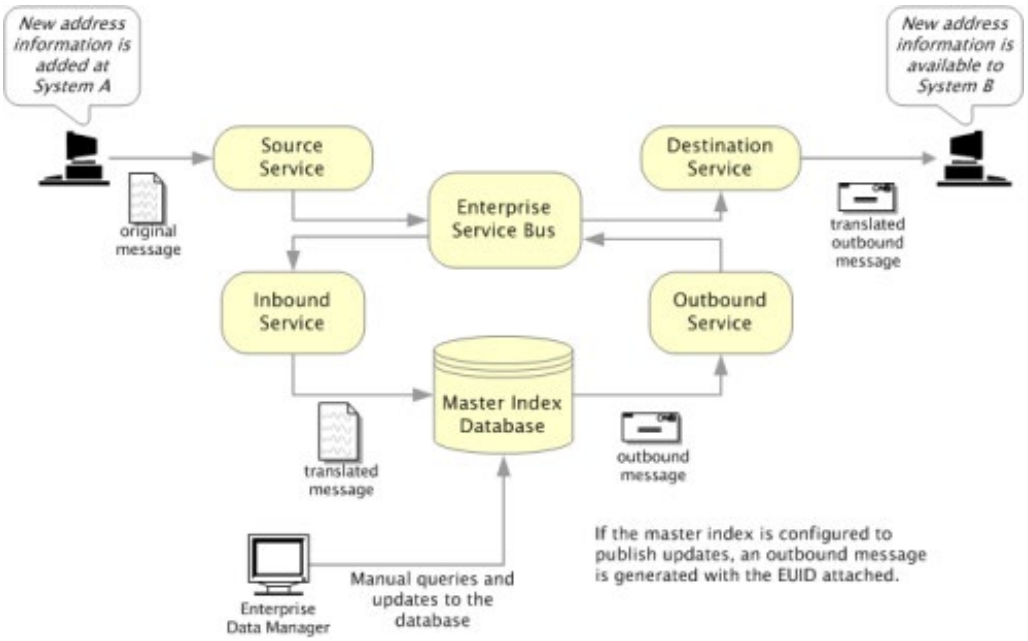


Рисунок 2.2 – Діаграма функціоналу обробки повідомлень

Першим етапом функціоналу обробки повідомлень є система розпізнавання тексту, яка аналізує вхідні повідомлення та перетворює текст у структурований формат для подальшої обробки. Це може включати в себе визначення ключових слів, фраз або шаблонів, що допомагають в ідентифікації сутностей або тем, що обговорюються.

Далі функціонал включає аналіз семантики повідомлень, що полягає у розумінні сутностей та контексту, що містяться в тексті. Це дозволяє системі відповідати на запити користувачів з урахуванням їхніх потреб та інтенцій.

Крім того, функціонал обробки повідомлень включає класифікацію проблем, що допомагає відокремити типові запитання та проблеми від більш складних або незвичайних ситуацій. Це дозволяє ефективно розподіляти ресурси підтримки та швидко реагувати на нагальні питання.

Нарешті, функціонал обробки повідомлень може включати автоматичні або напівавтоматичні системи відповідей на типові запитання або проблеми. Це допомагає знизити час очікування відповіді користувача та забезпечує швидке вирішення найпоширеніших питань.

Інтеграція з іншими системами

Інтеграція системи миттєвої підтримки з іншими внутрішніми системами компанії є важливим аспектом для забезпечення високоякісного обслуговування клієнтів та оптимізації бізнес-процесів. Ця інтеграція може включати такі системи, як бази даних клієнтів, системи управління відносинами з клієнтами (CRM), системи моніторингу та аналізу діяльності користувачів та інші [16].

Перш за все, інтеграція з базами даних клієнтів дозволяє системі миттєвої підтримки оперативно отримувати доступ до інформації про клієнтів, їхні попередні запити, історію взаємодії та інші важливі дані. Це дозволяє підтримці швидко оцінювати ситуацію та надавати персоналізовані відповіді.

Додатково, інтеграція з CRM-системами дозволяє автоматизувати процеси ведення бази клієнтів, управління зверненнями та аналізу результатів. Це сприяє покращенню взаємодії з клієнтами, виявленню нових можливостей для продажу та збільшенню задоволеності клієнтів [3].

Крім того, інтеграція з системами моніторингу та аналізу діяльності користувачів дозволяє збирати та аналізувати дані про поведінку користувачів, що допомагає виявляти та вирішувати проблеми з якістю обслуговування та покращувати досвід використання продукту чи сервісу.

Таким чином, інтеграція з іншими системами сприяє збільшенню ефективності роботи підтримки, покращенню обслуговування клієнтів та збільшенню задоволеності користувачів.

Забезпечення безпеки

Забезпечення безпеки є невід'ємною складовою системи миттєвої підтримки, оскільки вона може обробляти конфіденційну інформацію користувачів. Для забезпечення конфіденційності, цілісності та доступності даних необхідно застосовувати комплексний підхід до забезпечення безпеки.

Перш за все, використання методів шифрування даних є критично важливим. Шифрування даних дозволяє захистити інформацію від несанкціонованого доступу, забезпечуючи її зашифрований перехід по мережі та зберігання в зашифрованому вигляді на сервері.

Додатково, необхідно забезпечити захист доступу до системи шляхом використання автентифікації, паролів, біометричних методів або двофакторної аутентифікації. Це дозволить уникнути несанкціонованого входу в систему та зберігати дані в безпеці.

Моніторинг активності користувачів також є важливою складовою безпеки. Шляхом систематичного моніторингу можна вчасно виявляти підозрілу активність та вживати заходів для її нейтралізації, зменшуючи ризик злому чи несанкціонованого доступу [4].

Нарешті, для запобігання кібератакам необхідно вживати заходів забезпечення безпеки мережі, включаючи встановлення міжмережевих брандмауерів, застосування систем виявлення вторгнень та регулярні оновлення програмного забезпечення для закриття відомих вразливостей.

Подібний підхід до забезпечення безпеки допомагає зберегти конфіденційність даних користувачів, забезпечити надійний доступ до системи та запобігти можливим кібератакам, що можуть виникнути в мережі.

Цей функціонал допомагає створювати ефективні та зручні системи миттєвої підтримки, які дозволяють компаніям надавати високоякісний сервіс своїм користувачам та оперативно вирішувати їхні проблеми.

2.2 Технології розробки веб-додатків

Розробка веб-додатків є складним та динамічним процесом, який вимагає використання широкого спектру технологій та інструментів для створення функціональних та ефективних програм, призначених для роботи в онлайн-середовищі. Веб-додатки дозволяють користувачам взаємодіяти з інформацією та сервісами через веб-браузери, надаючи їм доступ до різноманітних функцій та можливостей безпосередньо через Інтернет.

Оскільки вимоги ринку та технологічні тренди постійно змінюються, розробники веб-додатків постійно шукають нові технології та підходи, щоб забезпечити високу якість та конкурентоспроможність своїх продуктів. Нижче наведено деякі ключові технології, які зараз широко використовуються в розробці сучасних веб-додатків:

Frontend технології

Фронтенд — це частина веб-додатка, що відповідає за інтерфейс, який бачить користувач та з яким він взаємодіє через браузер [5]. Використання правильних фронтенд технологій є важливим для створення зручного та привабливого інтерфейсу для користувача. Ось деякі ключові фронтенд технології (табл 2.1):

1. ***HTML (HyperText Markup Language)*** – HTML використовується для структурування веб-сторінок. Він визначає, як буде виглядати контент на сторінці, такий як текст, зображення, відео тощо.

2. ***CSS (Cascading Style Sheets)*** – CSS використовується для оформлення та стилізації веб-сторінок, які визначені HTML. Він дозволяє змінювати розміри, кольори, шрифти, відступи та інші візуальні аспекти сторінки.

3. ***JavaScript*** – JavaScript використовується для динамічних ефектів, взаємодії з користувачем та асинхронного обміну даними з сервером [15]. Він дозволяє створювати веб-сторінки з анімацією, валідацією форм, реактивними елементами і багато іншого.

4. ***React*** – Бібліотека JavaScript для створення інтерфейсів користувача, яка допомагає розбити UI на незалежні компоненти, що полегшує розробку складних веб-додатків.

5. ***Vue.js*** – Прогресивний JavaScript фреймворк для створення користувацьких інтерфейсів. Він використовується для розробки односторінкових додатків та додавання інтерактивності до існуючих веб-сайтів.

6. ***Angular*** – Фреймворк JavaScript, розроблений Google, який використовується для створення веб-додатків з високою швидкістю та масштабованістю.

Таблиця 2.1 – Порівняння фронтенд технологій

Технологія	**Використання**
React	Підходить для складних інтерфейсів з багатьма взаємодіючими елементами
Vue.js	Ідеальний для розробки невеликих та середніх проектів з інтерактивними інтерфейсами
Angular	Рекомендується для великих масштабних додатків зі складною логікою
SASS/SCSS	Зручно використовувати для стилізації складних та розширених веб-додатків
Bootstrap	Підходить для швидкої розробки адаптивних та стилізованих інтерфейсів
Tailwind CSS	Ідеальний для створення інтерфейсів за допомогою заранее визначених класів

Backend технології

Бекенд — це серверна частина веб-додатка, яка відповідає за обробку запитів користувачів, взаємодію з базою даних та відправку відповідей на клієнтську сторону. Для розробки бекенду використовуються різноманітні технології, які дозволяють створювати потужні та надійні веб-додатки. Ось деякі ключові бекенд технології (табл 2.2):

1. ***PHP***. PHP є однією з найпопулярніших мов програмування для веб-розробки бекенду. Вона має широкі можливості та велику кількість різноманітних фреймворків, таких як Laravel, Symfony, Yii та інші[6].

2. ***Node.js***. Node.js — це середовище виконання JavaScript, яке дозволяє створювати швидкі та масштабовані серверні додатки. Він популярний для розробки веб-додатків в реальному часі та API.

3. ***Python***. Python є добре відомим вибором для розробки бекенду завдяки своїй простоті та чистоті коду. Фреймворки, такі як Django та Flask, допомагають швидко створювати потужні веб-додатки.

4. ***Ruby***. Ruby разом з фреймворком Ruby on Rails є чудовим вибором для швидкого розвитку веб-додатків. Rails пропонує конвенцію над конфігурацією, що сприяє швидкості розробки.

5. ***Java***. Java використовується для створення масштабних та надійних серверних додатків. Фреймворки, такі як Spring та Hibernate, допомагають розробникам побудувати складні системи.

6. ***Go (Golang)***. Go є мовою програмування, яка набирає популярність у розробці веб-додатків. Вона пропонує швидкість виконання та простоту синтаксису.

Таблиця 2.2 – Порівняння бекенд технологій

Технологія	**Використання**
PHP	Відмінно підходить для швидкої розробки веб-сайтів та веб-додатків.
Node.js	Ідеальний для створення серверів в реальному часі, асинхронних додатків та API.
Python	Зручний використовувати для розробки складних веб-додатків та наукових проектів.
Ruby	Чудово підходить для швидкої розробки веб-сайтів з використанням Ruby on Rails фреймворку.
Java	Рекомендується для масштабних та надійних веб-додатків, особливо в корпоративному середовищі.
Go	Підходить для швидкої розробки високонавантажених серверних додатків.

Бази даних

База даних є фундаментальною складовою будь-якого веб-додатка, оскільки вона відповідає за зберігання та управління інформацією, яка використовується в додатку. Розробники веб-додатків можуть вибирати з різних систем керування базами даних (СКБД), кожна з яких має свої особливості та переваги, що враховуються при виборі відповідно до вимог конкретного проекту.

Наприклад, MySQL є однією з найпопулярніших відкритих СКБД та зазвичай використовується для різноманітних веб-додатків, включаючи малий бізнесовий рівень та великі корпоративні системи [7]. PostgreSQL, з іншого боку, відомий своєю надійністю та розширюваністю, що робить його популярним в середовищах, де вимагається великий обсяг даних та високі стандарти безпеки.

MongoDB, у свою чергу, відноситься до NoSQL баз даних та забезпечує гнучку схему даних, що особливо корисно для розробки додатків, де потрібна швидка ітерація та розвиток.

SQLite, хоча і зазвичай використовується в менших проектах або для розробки прототипів, є вбудованою СКБД, яка не вимагає окремого сервера баз даних, що робить її популярною для додатків, що працюють в обмежених умовах, таких як мобільні додатки [18].

Крім вибору конкретної СКБД, важливо також враховувати аспекти, такі як вимоги до продуктивності, масштабованість, доступність та безпеку, щоб забезпечити оптимальну роботу веб-додатка в цілому.

2.2.1 Розробка веб-додатків на основі AJAX

Розробка веб-додатків на основі AJAX (Asynchronous JavaScript and XML) є ключовою складовою для створення сучасних та інтерактивних інтерфейсів веб-додатків. AJAX використовується для забезпечення асинхронної взаємодії між користувачем та сервером без необхідності повного перезавантаження веб-сторінки [8].

Ця технологія дозволяє веб-додаткам взаємодіяти з сервером у фоновому режимі, що забезпечує швидку та плавну реакцію на дії користувачів. Наприклад, веб-сторінка може виконувати запити до сервера для отримання або оновлення даних без перезавантаження всієї сторінки. Це дозволяє створювати більш динамічні та ефективні веб-додатки, які відповідають вимогам сучасних користувачів.

Основні переваги використання AJAX включають підвищену ефективність, більшу інтерактивність, меншу потребу у пропускній здатності та підтримку багатьох форматів даних. Взаємодія без повного перезавантаження сторінки дозволяє зменшити час очікування користувача та навантаження на сервер. Крім того, AJAX надає зручну можливість реагувати на дії користувачів миттєво, створюючи більш зручний та інтуїтивно зрозумілий інтерфейс (табл 2.3).

Таблиця 2.3 – Порівняння AJAX з іншими методами взаємодії з сервером

Метод взаємодії	**Переваги**	**Недоліки**	**Використання**
AJAX	Взаємодія без повного перезавантаження сторінки, ефективність, більша інтерактивність	Потребує додаткової обробки помилок, може викликати проблеми з SEO, може збільшувати складність коду	Веб-додатки, які потребують динамічного оновлення вмісту без повного перезавантаження сторінки
Синхронний запит	Простота в реалізації, працює незалежно від JavaScript	Повільний, блокує інтерфейс користувача під час очікування відповіді від сервера, перезавантажує всю сторінку	Веб-сторінки з невеликою кількістю даних, які не потребують частого оновлення
Long Polling	Зменшення кількості запитів до сервера, простота реалізації	Збільшена кількість HTTP-з'єднань, може затримувати відправку даних на клієнт, можливість втрати підключення	Додатки, де необхідно миттєве оновлення даних, але рідше, ніж при використанні WebSocket
WebSockets	Реальний час, низька затримка, ефективне використання ресурсів, підтримка двонаправленого зв'язку	Складніше в налаштуванні та реалізації, може вимагати спеціальної інфраструктури на сервері	Додатки, які потребують постійного обміну даними у реальному часі, такі як чати та стрімінгові додатки

Під час розробки системи миттєвої підтримки користувачів на основі PHP та AJAX, важливо обрати належний метод взаємодії з сервером. AJAX може бути чудовим варіантом для багатьох типів веб-додатків, оскільки він дозволяє забезпечити асинхронну взаємодію без перезавантаження сторінки, зменшуючи затримки та поліпшуючи відгук додатка. Однак, для ситуацій, де необхідна більша швидкість або постійний потік даних у реальному часі, WebSockets може бути більш підходящим варіантом, незважаючи на його складність в реалізації [9]. Крім того, при розгляді методу взаємодії з сервером слід враховувати потенційні обмеження та вимоги до інфраструктури сервера.

2.2.2 Розробка веб-додатків з використанням фреймворків

Розробка веб-додатків з використанням фреймворків є одним із найпоширеніших підходів у сучасній веб-розробці. Фреймворки представляють собою набори готового коду, які надають стандартизовані рішення для певних завдань або проблем у розробці програмного забезпечення.

Основна перевага використання фреймворків полягає в тому, що вони дозволяють розробникам ефективно використовувати готові компоненти та шаблони. Це значно спрощує процес розробки, адже розробнику не потрібно писати весь код з нуля, а можна використовувати готові рішення для типових задач.

Наприклад, Laravel для мови програмування PHP, Express.js для Node.js, Django для Python та Ruby on Rails для Ruby - це лише кілька прикладів популярних веб-фреймворків, які знаходять широке застосування в індустрії [10].

Використання фреймворків дозволяє розробникам зосередитися на бізнес-логіці додатку, не втрачаючи час на написання базових компонентів та вирішення типових завдань. Крім того, фреймворки часто надають ряд допоміжних інструментів, таких як механізми автентифікації, маршрутизації та валідації даних, що спрощує структуру додатку та підвищує його безпеку.

Розробка веб-додатків вимагає вивчення та використання різноманітних технологій, інструментів та підходів. Обрані технології повинні відповідати вимогам проекту та забезпечувати якісний та ефективний результат. Використання фреймворків стає важливим етапом у цьому процесі, оскільки вони дозволяють розробникам створювати веб-додатки швидше, ефективніше та з меншими витратами ресурсів.

2.3 Принципи проектування користувацьких інтерфейсів

Проектування користувацьких інтерфейсів (UI) є невід'ємною та критично важливою складовою процесу розробки веб-додатків. Основна мета цього етапу - створення такого інтерфейсу, який забезпечить зручну та ефективну взаємодію між користувачем та програмним забезпеченням. Дотримання ключових принципів проектування є невід'ємною частиною цього процесу та визначає якість та ефективність результуючого інтерфейсу.

Один з основних принципів - це простота та зрозумілість. Це означає, що інтерфейс повинен бути максимально легким для сприйняття та використання, не створювати заплутаності та не потребувати великої кількості кроків для досягнення поставленої мети. Користувач повинен інтуїтивно розуміти, як взаємодіяти з додатком та як досягти бажаного результату, не потребуючи додаткового навчання чи інструкцій.

Простота і зрозумілість інтерфейсу сприяє позитивному досвіду користувача, роблячи процес взаємодії приємним та ефективним. Крім того, цей принцип сприяє зменшенню кількості помилок та підвищенню продуктивності користувачів, оскільки вони швидше та легше досягають своїх цілей за допомогою інтуїтивно зрозумілого інтерфейсу. Такий підхід дозволяє створювати додатки, які задовольняють потреби користувачів та забезпечують позитивний досвід використання.

Принцип консистентності є одним із ключових аспектів при проектуванні користувацьких інтерфейсів, оскільки він забезпечує єдність та послідовність у структурі та вигляді інтерфейсу на всіх його етапах. Цей принцип допомагає забезпечити користувачам однаковий досвід взаємодії з програмою чи веб-сайтом, що сприяє зручності навігації та полегшує їм розуміння, як користуватися додатком [11].

Консистентність дозволяє користувачам швидше розібратися з інтерфейсом, оскільки вони можуть передбачати, які елементи будуть знаходитися на різних сторінках чи в різних частинах програми. Наприклад, збереження однакового розташування меню або кнопок на всіх сторінках допомагає користувачам швидше зорієнтуватися та знайти необхідні функції.

Крім того, для досягнення консистентності важливо також дотримуватися принципу відповідності завданням. Цей принцип передбачає, що інтерфейс повинен відображати лише необхідну інформацію та функціонал, який дійсно сприяє виконанню завдань користувача. Відсутність зайвих або непотрібних елементів дозволяє зробити інтерфейс більш зрозумілим та ефективним для користувачів, забезпечуючи їм зручний та ефективний досвід взаємодії з програмою чи веб-сайтом.

Ще одним важливим аспектом є принцип забезпечення зручності навігації, що передбачає інтуїтивне та легке переміщення користувача по програмі чи веб-сторінці [12]. Чітке визначення шляхів переходу та функціоналу спрощує взаємодію та зменшує можливість збентеження користувачів. Принцип візуальної привабливості також не може бути недооціненим. Інтерфейс повинен бути не лише функціональним, а й привабливим для користувача, використання гармонійних кольорів, правильного шрифту та ефектів може поліпшити загальне враження від використання додатку.

Ці принципи є основою успішного проектування користувацьких інтерфейсів і важливі для створення досвіду, що задовольняє потреби користувачів та сприяє успішності програми чи веб-додатка.

2.4 Огляд та аналіз вимог до системи

У розвитку системи миттєвої підтримки користувачів, базованої на PHP та Ajax, ключовими аспектами є швидкість реакції на запити користувачів та гнучкість у взаємодії. Це дозволяє не тільки підвищити задоволеність користувачів, а й забезпечити їм негайну підтримку в разі потреби. Застосування технологій PHP та Ajax відкриває можливість для створення більш динамічних і інтерактивних веб-сайтів, де зміст може оновлюватися на льоту, не вимагаючи повного перезавантаження сторінки [13]. Це особливо важливо для систем миттєвої підтримки, де швидкість відповіді та здатність до негайного реагування на запити користувачів є критичними факторами.

Аналіз конкурентних рішень показує, що багато сучасних систем підтримки надають широкий спектр функціональних можливостей, включаючи автоматизовану обробку запитів за допомогою штучного інтелекту та машинного навчання, а також можливість взаємодії з живими операторами. Однак, існуючий потенціал для інновацій полягає у впровадженні адаптивних алгоритмів штучного інтелекту, які б могли не лише аналізувати і обробляти запити в режимі реального часу, але й навчатися на попередніх взаємодіях, тим самим покращуючи якість обслуговування та роблячи відповіді більш персоналізованими.

Інноваційний компонент, який може вирізняти розроблювану систему на тлі конкурентів, може включати розширений аналіз емоційних звернень користувачів з метою адаптації відповідей до їхнього емоційного стану. Такий підхід дозволить не тільки надавати рішення або відповіді на запити, а й зробити взаємодію з системою більш гуманною та емпатійною, підвищуючи загальне задоволення користувачів від використання сервісу.

Для забезпечення ефективності розроблюваної системи необхідно зосередитися на таких технічних аспектах, як оптимізація бази даних для швидкого доступу до і нформації про користувачів та їхні запити. Використання MySQL як системи управління базами даних дозволяє реалізувати надійне зберігання даних та їх швидкий доступ. Ключовою перевагою є можливість розробки комплексної

структури бази даних, що підтримує ефективну обробку великих обсягів даних і водночас забезпечує високу швидкість запитів.

Розширений аналіз емоційних звернень, як зазначалось, вимагає інтеграції алгоритмів машинного навчання та обробки природної мови, що дозволить системі не просто відповідати на запити, а робити це в контексті емоційного стану користувача. Такий підхід передбачає використання передових технологій та бібліотек, наприклад, TensorFlow або PyTorch для розробки моделей машинного навчання, та NLTK або SpaCy для обробки природної мови. Це дозволить системі адаптуватися та навчатися на основі зібраних даних, неперервно покращуючи якість та точність відповідей.

Оскільки розробка здійснюється на основі PHP та Ajax, важливо забезпечити високий рівень безпеки взаємодії між клієнтом та сервером. Використання сучасних практик захисту даних, таких як HTTPS, SSL-шифрування, токени доступу, а також реалізація заходів проти атак SQL-ін'єкцій та Cross-Site Scripting (XSS), є обов'язковими для захисту конфіденційності користувачів та інтегритету даних [14].

Завершуючи аналітичний огляд, можна сказати, що успіх системи миттєвої підтримки користувачів залежить не тільки від технічної реалізації, а й від глибокого розуміння потреб користувачів та здатності до інновацій. Інтеграція передових технологій та створення інтуїтивно зрозумілого, ефективного інтерфейсу можуть визначити нові стандарти в області миттєвої підтримки, пропонуючи користувачам не тільки вирішення

3 ПРОЕКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

3.1 Проектування архітектури системи

Архітектура системи миттєвої підтримки втілює в собі гармонійну взаємодію між людиною та машиною, орієнтовану на забезпечення високоякісної та ефективної служби підтримки. Узявши за основу надану схему, можна побачити, як звичайний запит користувача перетворюється в діалог, у якому штучний інтелект та оператор працюють разом для досягнення кінцевої мети – задоволення потреб клієнта (рис. 3.1).

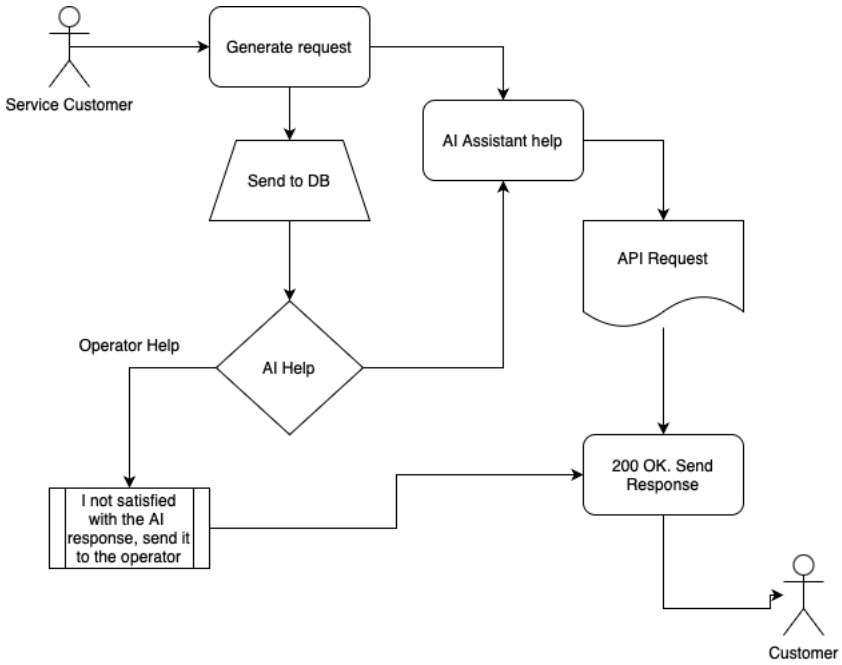


Рисунок 3.1 – Діаграма роботи застосунку

На першому етапі клієнт ініціює взаємодію з системою, висловлюючи своє запитання або заяву. Ця інформація збирається та структурується через інтерфейс користувача, який розроблено таким чином, щоб бути інтуїтивно зрозумілим та простим у використанні. Потім, дані надсилаються до сервера, де вони зберігаються у базі даних. Це дозволяє системі швидко отримувати доступ до історії користувача та до поточного контексту запиту.

Наступний етап передбачає залучення модуля штучного інтелекту, який обробляє отримані дані та намагається надати найкраще можливе рішення. Штучний інтелект використовує попередньо накопичені дані, аналітику поведінки та алгоритми машинного навчання для того, щоб зрозуміти запит користувача і запропонувати відповіді.

У разі, якщо AI не знаходить відповіді, або вона не влаштовує користувача, система перенаправляє запит до живого оператора, який вже має всю історію діалогу та аналіз AI перед очима, що дає йому змогу більш ефективно вирішити проблему. У процесі обробки запиту може знадобитися інтеграція з внутрішніми та зовнішніми API для отримання додаткових даних або для виконання певних дій. Це вимагає створення безпечних та надійних каналів зв'язку між системою підтримки та іншими сервісами. Завершальний етап полягає в тому, що система надає відповідь користувачу. Це може бути просте повідомлення про успішне виконання запиту або, у випадку більш складних сценаріїв, детальна інструкція або рекомендація. За кожною відповіддю стоїть аналітика даних, що дозволяє системі не тільки відповісти на запит, а й збагатити внутрішню базу знань для покращення подальшого обслуговування.

Враховуючи необхідність гнучкості у взаємодії з клієнтом, система також забезпечує функціонал переходу від штучного інтелекту до живого оператора. Це означає, що в архітектурі системи передбачено не тільки алгоритми AI, а й засоби для управління сесіями в чаті, розподілу навантаження між операторами та ведення статистики ефективності роботи служби підтримки.

При проектуванні архітектури також слід звернути увагу на забезпечення високого рівня безпеки. Компонент безпеки включає шифрування даних, регулярне оновлення сертифікатів, захист від зовнішніх загроз і зловмисних атак, а також забезпечення відповідності вимогам законодавства щодо захисту даних.

Збалансоване поєднання цих компонентів і принципів в архітектурі створює міцну основу для системи миттєвої підтримки, яка здатна адаптуватися до різних сценаріїв використання та розширюватися разом зі зростанням потреб бізнесу.

3.2 Огляд технологічного стеку

Вибір технологічного стеку для дипломної роботи, яка фокусується на розробці системи миттєвої підтримки, базується на сучасних практиках розробки програмного забезпечення. Нижче наведений огляд основних технологій та бібліотек, що були обрані для реалізації проекту.

JavaScript: Як основна мова для розробки клієнтської частини, JavaScript використовується для створення інтерактивних елементів інтерфейсу та забезпечення асинхронної взаємодії з сервером за допомогою Ajax. Динамічне змістове наповнення, анімації, валідація форм — усе це реалізовано на JavaScript.

PHP: Серверна мова програмування PHP задіяна для обробки запитів від користувачів, взаємодії з базою даних і виконання логіки бізнес-операцій. PHP було обрано через його широку підтримку хостинговими провайдерами та гнучкість у розробці веб-додатків.

CSS: Стильова мова CSS застосовується для оформлення веб-сторінок, забезпечуючи їх привабливий та сучасний вигляд. Використання передпроцесорів, таких як Sass або LESS, дозволяє підвищити ефективність стилізації компонентів.

Hack: Мова програмування Hack використовується як модернізований діалект PHP, який надає додаткові можливості для типізації та обробки асинхронного коду, покращуючи продуктивність та безпеку застосунку.

Dockerfile: Конфігурація Docker дозволяє створювати ізольоване середовище для додатку, що спрощує розгортання та забезпечує однакові умови роботи додатку на різних машинах.

Бібліотека orhanerday/open-ai: Це клієнт API для OpenAI GPT-3, написаний на PHP, який інтегрується у систему, дозволяючи використовувати потужні алгоритми машинного навчання для автоматизації відповідей системи підтримки та обробки запитів користувачів на природній мові.

Завдяки цим технологіям та інструментам, розроблювана система отримує ряд переваг, зякі включають швидкість, безпеку, адаптивність і можливість інтеграції. Цей комплексний підхід забезпечує високу продуктивність та гнучке

налаштування системи відповідно до змінних вимог користувачів та технологічних трендів.

Застосування бібліотеки orhanerday/open-ai в проекті дозволяє використовувати переваги однієї з найбільш передових технологій в галузі штучного інтелекту, GPT-3 від OpenAI, що надає можливість обробляти природну мову на високому рівні. Використання цієї бібліотеки забезпечує розроблюваній системі здатність глибоко розуміти запити користувачів, пропонувати релевантні відповіді та вивчати взаємодії для покращення подальшого сервісу.

Використання Docker у проекті гарантує, що програмне забезпечення може бути легко розгорнуте в будь-якому середовищі, що забезпечує консистентність між розробкою та виробництвом та відповідність сучасним практикам DevOps.

Також, важливо відзначити, що **безпека є основоположним елементом** у всіх аспектах розробки. Використання HTTPS, SSL-шифрування, сучасних алгоритмів аутентифікації та авторизації, а також систем захисту від SQL-ін'єкцій та XSS-атак, гарантує захист персональних даних користувачів та інтегритет системи.

Разом, ці технологічні рішення складають надійний фундамент для розробки системи миттєвої підтримки, здатної задовольнити найсуворіші вимоги користувачів та бізнесу.

3.3 Розробка інтерфейсу та серверної логіки системи миттєвої підтримки

Створення системи миттєвої підтримки включає в себе ретельне планування структури проекту, вибір адекватних технічних рішень для кожного з компонентів, та реалізацію цих компонентів в коді. Структура проекту показана на наданому зображенні відображає організацію файлів і папок, яка забезпечує зручне розташування всіх частин системи та їх легку доступність для розробників.

У корені проекту **CustomerAssistant** розташовуються основні файлі виконання, а також необхідні конфігураційні файли (рис. 3.2).

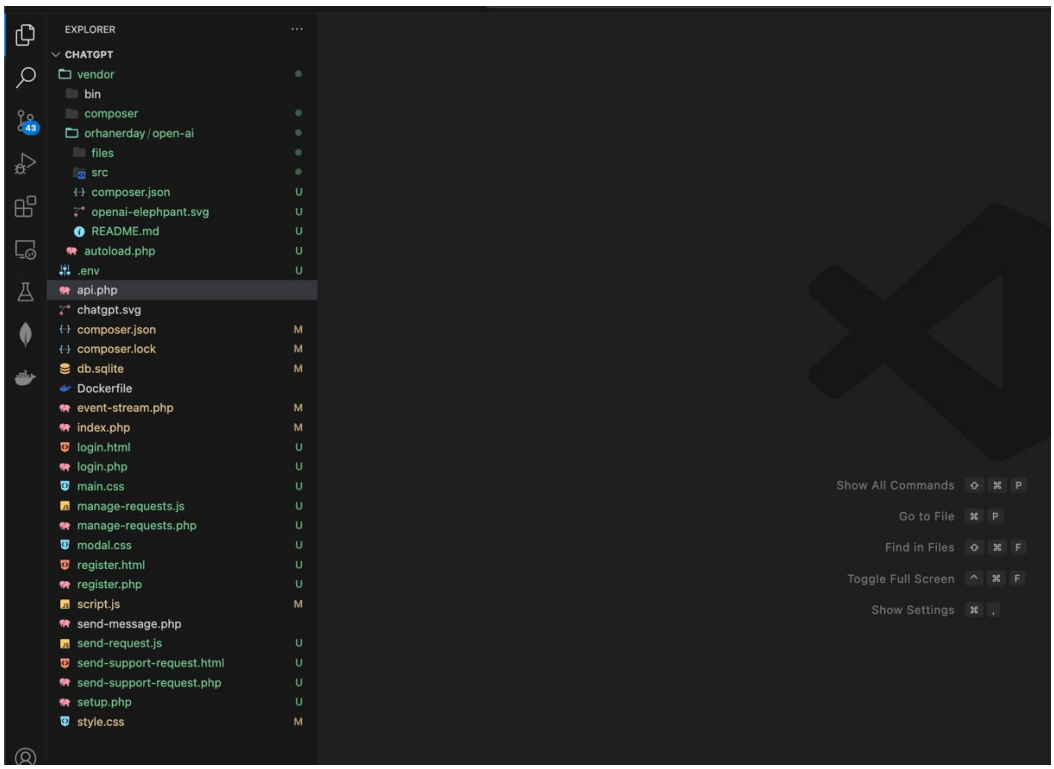


Рисунок 3.2 – Файлова структура додатку

Це створює чіткий і логічний розподіл відповідальності між різними сегментами системи:

- **api.php:** Цей файл відіграє роль вхідної точки API, через яку обробляються запити до системи. Це може включати обробку користувацьких запитів, інтеграцію з іншими системами та логіку API для штучного інтелекту.
- **composer.json та composer.lock:** Файли, пов'язані з менеджером залежностей Composer для PHP, забезпечують визначення та блокування залежностей проекту відповідно.
- **Dockerfile:** Визначає конфігурацію Docker контейнера для проекту, дозволяючи створювати ізольоване середовище для однакового розгортання на різних платформах.
- **db.sqlite:** Файл бази даних SQLite, який використовується для зберігання даних про користувачів, їх запити та історію взаємодій.
- **index.php:** Головний файл, який може слугувати вхідною точкою веб-додатку, об'єднуючи всі модулі системи.

- **script.js:** Файл JavaScript, що містить клієнтську логіку для інтерактивності веб-інтерфейсу, включаючи обробку подій користувача та асинхронні запити до сервера.
- **send-message.php:** Скрипт для обробки відправлення повідомлень користувачів через систему, може включати логіку валідації та збереження повідомлень.
- **style.css:** Файл стилів CSS відповідає за візуальну привабливість веб-сайту та забезпечує однорідність користувацького інтерфейсу на різних пристроях і платформах. Оформлення елементів, адаптивний дизайн та реагування на зміни розмірів вікон — усі ці аспекти регулюються за допомогою цього файлу.

Така структура проекту сприяє чіткому розділенню задач між фронтендом та бекендом, дозволяючи команді розробників ефективно працювати над різними аспектами системи миттєвої підтримки. Інтеграція різних компонентів та забезпечення їх спільної роботи вимагає глибокого розуміння бізнес-логіки проекту, а також технічних можливостей використовуваних технологій.

Файл **README.md** є документацією проекту, яка забезпечує опис роботи системи, інструкції для розгортання та використання, а також інформацію про ліцензію і авторські права. Це стає особливо важливим, коли проект розвивається і до нього долучаються нові учасники або він передається клієнту.

Розробка системи миттєвої підтримки — це не лише про написання коду, але й про створення логічної та інтуїтивно зрозумілої структури проекту, що дозволяє легко масштабувати та підтримувати додаток. Кожен файл та папка у проекті має певне призначення, і ефективне їх розташування сприяє кращій організації та подальшому розвитку системи.

3.4 Основні компоненти та ключові аспекти реалізації програмного забезпечення

Система миттєвої підтримки користувачів, розроблена у цій дипломній роботі, представляє собою витончене поєднання веб-технологій, яке надає користувачам миттєву взаємодію зі службою підтримки. Починаючи з моменту, коли користувач вводить свій запит у веб-інтерфейсі, система вступає в дію, обробляючи цей запит за допомогою сучасних веб-технологій та алгоритмів штучного інтелекту.

На веб-сторінці користувач зустрічає інтерактивну чат-форму, створену з використанням JavaScript, яка забезпечує динамічне спілкування без необхідності перезавантаження сторінки. Введене повідомлення відправляється на сервер через асинхронний запит API, виконаний в **api.php**.

Серверна частина, написана на PHP, отримує цей запит і проводить його первинну обробку. У випадку необхідності отримання додаткової інформації або виконання конкретних дій, сервер може звернутися до внутрішньої бази даних **db.sqlite** для отримання історії взаємодій користувача або до зовнішніх API.

Оглянемо ключові модулі проекту:

Модуль Api.php

Модуль **api.php** відіграє роль серця системи миттєвої підтримки користувачів, керуючи основними операціями з даними та взаємодією з клієнтом. Він приймає запити, обробляє їх і виконує відповідні дії на стороні сервера (рис. 3.3).

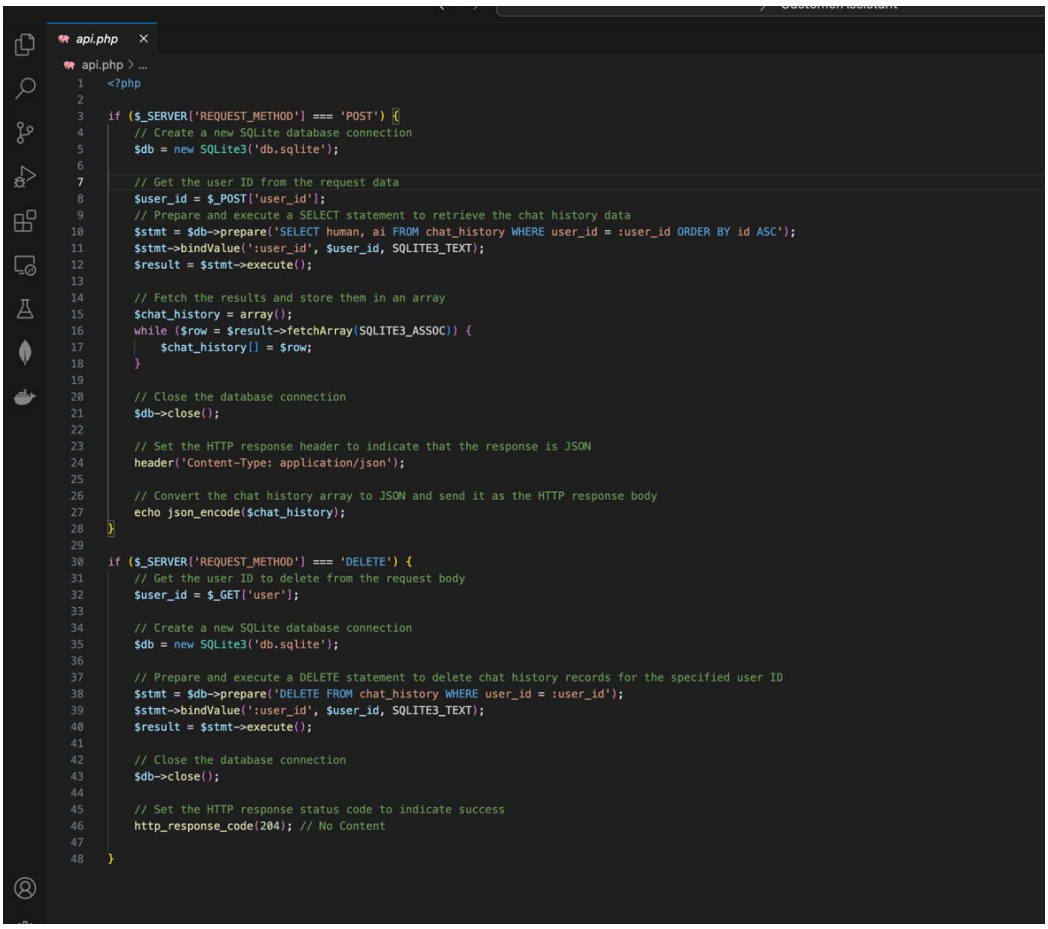


Рисунок 3.3 – Модуль API.php

При отриманні POST запиту, **api.php** встановлює з'єднання з локальною базою даних SQLite, що дозволяє системі ефективно зберігати та отримувати історію чатів користувача. Ця операція забезпечує ізоляцію та швидкий доступ до історичних даних, не навантажуючи зовнішні системи баз даних.

Код у **api.php** налаштовано на роботу з вхідними даними користувача – в даному випадку із значенням **user_id**, яке передається методом POST. Після отримання **user_id**, скрипт готує SQL-запит для вибірки відповідної історії чату з бази даних. Запит виконується, і результати – повертаються користувачу у форматі JSON, що дозволяє легко інтегрувати дані у клієнтську частину системи.

Для запитів DELETE, модуль **api.php** знову ініціює з'єднання з базою даних та обробляє запит на видалення, використовуючи отриманий **user_id**. Ця частина

логіки забезпечує можливість видалення історії чату для конкретного користувача, що є важливою функцією з погляду конфіденційності та управління даними.

Весь цикл взаємодії — від отримання запиту від користувача до повернення відповіді – відбувається швидко та ефективно, гарантуючи низький час відгуку і високу якість сервісу миттєвої підтримки.

Модуль Event Stream

Модуль **event-stream.php** відіграє вирішальну роль у підтримці безперервної взаємодії користувача з системою миттєвої підтримки, виступаючи в якості моста між клієнтським інтерфейсом та серверними обчислювальними процесами (рис. 3.4).

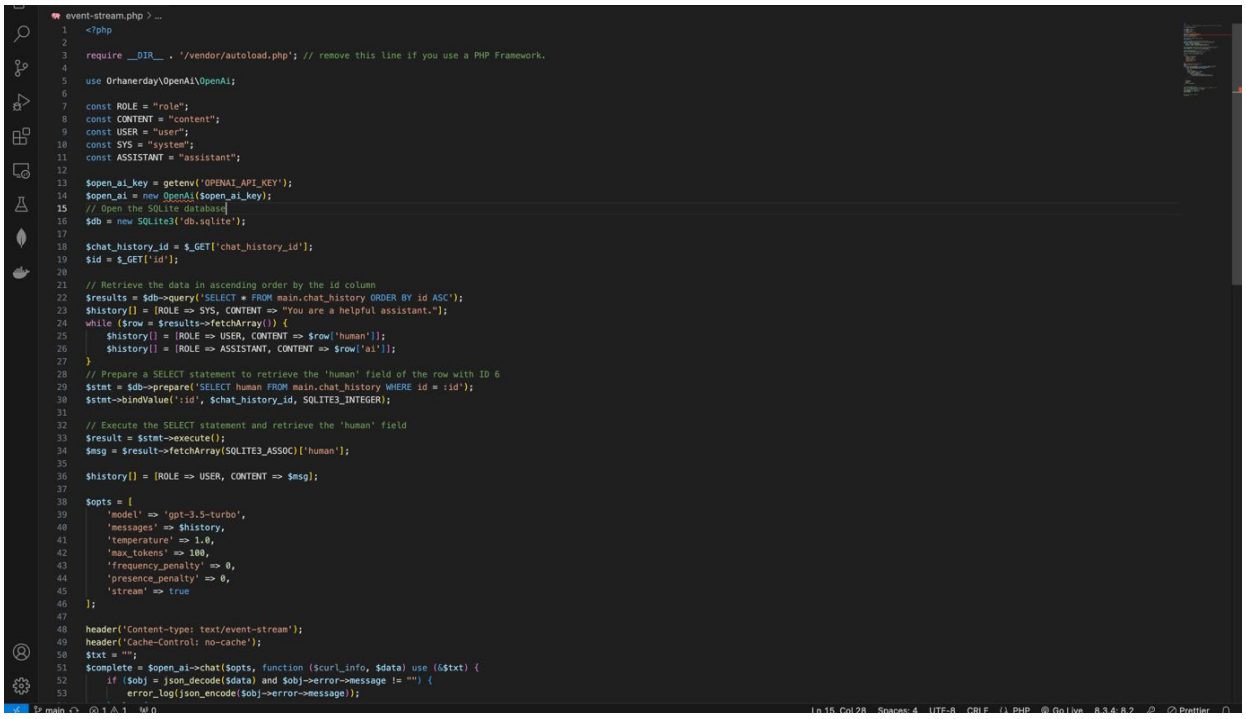


Рисунок 3.4 – Модуль Event-Stream

Процес роботи цього модуля розпочинається з підключення необхідних бібліотек для подальшої роботи з OpenAI, зокрема, використовуючи бібліотеку

Orhanerday\OpenAi\OpenAi, що дає можливість взаємодіяти з потужними алгоритмами штучного інтелекту.

З використанням SQLite бази даних, **event-stream.php** запитує та збирає історію взаємодії користувачів зі штучним інтелектом, формуючи контекст для наступної взаємодії. Він відновлює весь діалог, включаючи запити користувача та відповіді штучного інтелекту, створюючи цілісну картину бесіди.

Основна функція модуля полягає в реалізації механізму подій у реальному часі (Server-Sent Events), що дозволяє серверу активно надсилати до клієнта оновлення, які виникають у відповідь на запити користувачів. Це сприяє створенню ілюзії безпосередньої взаємодії з живим оператором.

Використання **event-stream.php** забезпечує здатність системи миттєво реагувати на повідомлення користувачів, генерувати відповіді в реальному часі та оновлювати інтерфейс користувача без необхідності в ручному оновленні сторінки. Крім того, цей модуль вносить важливий вклад у забезпечення неперервної взаємодії в додатку, що є вирішальним для підтримки високого рівня задоволеності користувачів.

Завершуючи процес, модуль виконує оновлення бази даних з новою інформацією, отриманою від штучного інтелекту, тим самим забезпечуючи актуальність даних для подальшого аналізу та використання в системі.

Модуль index.php

Модуль **index.php** є основою користувацького інтерфейсу для системи миттєвої підтримки, створюючи веб-сторінку, яка дозволяє користувачам взаємодіяти з чат-ботом. Ця веб-сторінка становить фронтенд додатку і визначає структуру та вигляд інтерфейсу, через який користувачі вводять свої запити та отримують відповіді (рис. 3.5).

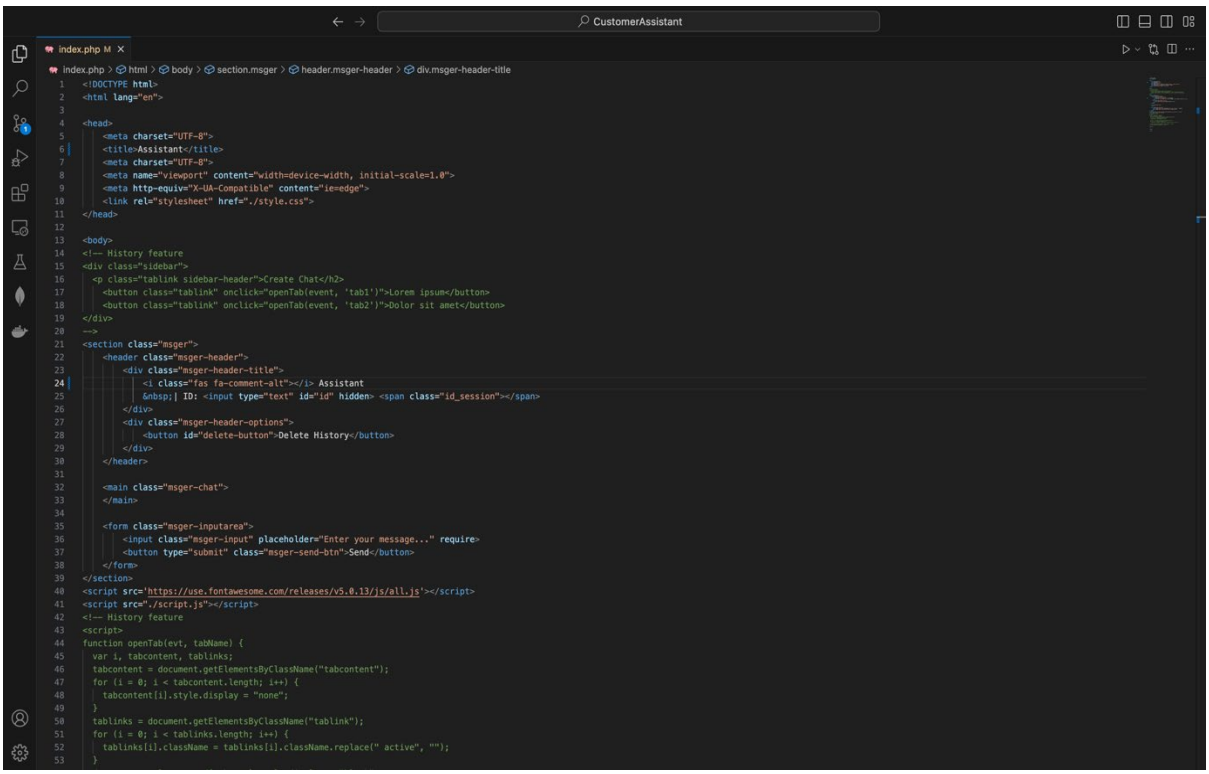


Рисунок 3.5 – Модуль Index.php

Структура HTML документа в **index.php** організована за допомогою семантичних тегів, що забезпечує ясність та доступність контенту. Метатеги, які вказують на кодування UTF-8 та сумісність з різними веб-оглядачами, гарантують правильне відображення сторінки на різноманітних пристроях. Використання відкликаного CSS файлу **style.css** забезпечує стилізацію інтерфейсу, роблячи його візуально привабливим та зручним для користувача.

У розділі **<header>** розміщено назву чату "Assistanse" та елементи для відображення ідентифікатора сесії, що можуть бути використані для відстеження чи управління історією чату. Кнопка "Delete History" дає користувачам можливість видаляти історію спілкування, що сприяє забезпеченню конфіденційності.

Основний вміст сторінки розміщений у розділі **<main>**, який представляє собою область чату, де відображатимуться повідомлення користувачів та відповіді штучного інтелекту. Форма вводу **<form>** дозволяє користувачам відправляти свої повідомлення, які після натискання кнопки "Send" передаються на сервер для обробки.

Скрипти JavaScript, підключені до сторінки, зокрема **script.js**, відповідають за обробку подій на сторінці, взаємодію з сервером через асинхронні запити та оновлення інтерфейсу без необхідності перезавантаження сторінки. Вони забезпечують динамічну взаємодію та підтримку користувацьких сценаріїв у реальному часі.

В цілому, **index.php** є ключовим елементом користувацького інтерфейсу системи миттєвої підтримки, створюючи міст між користувачем та сервером, і забезпечуючи зручне та інтуїтивно зрозуміле середовище для спілкування

Модуль Script.js

Модуль **script.js** є важливою частиною фронтенду в системі миттєвої підтримки, відповідаючи за взаємодію користувача з веб-інтерфейсом і забезпечуючи асинхронну комунікацію з сервером. Давайте розглянемо його основні компоненти та логіку роботи (рис. 3.6).

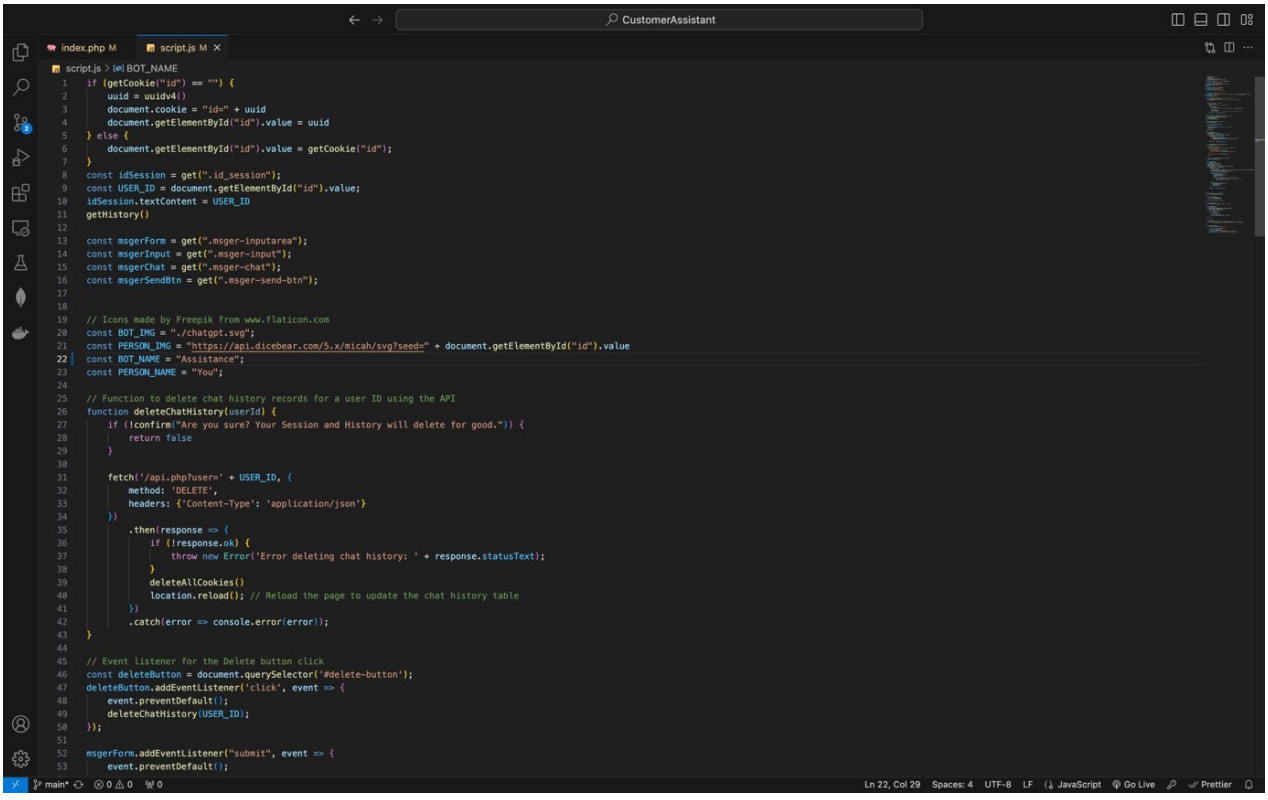


Рисунок 3.6 – Модуль script.js

Початок роботи скрипта визначається перевіркою наявності унікального ідентифікатора користувача (ID сесії) в cookies. Якщо ID відсутній, скрипт генерує новий за допомогою функції **uuidv4** і зберігає його в cookies, а також відображає у відповідному полі інтерфейсу. Цей ID використовується для ідентифікації користувача та його історії чату на сервері.

Основна логіка скрипта організована навколо взаємодії з формою вводу повідомлень. При надсиланні форми (**submit**) користувачем, текст повідомлення відправляється на сервер через асинхронний запит, а у чат додається нове повідомлення від імені користувача.

Для видалення історії чату реалізована функція **deleteChatHistory**, яка викликається при натисканні на кнопку "Delete History". Ця функція виконує асинхронний запит до **api.php** з методом DELETE, що призводить до видалення історії чату з бази даних.

Окрім цього, скрипт забезпечує отримання та відображення історії чату з сервера за допомогою функції **getHistory**. Історія чату завантажується на початку сесії, дозволяючи користувачам переглядати попередні повідомлення.

Ключовою особливістю модуля є реалізація механізму отримання відповідей від штучного інтелекту через EventSource. При кожному надсиланні повідомлення ініціюється з'єднання з **event-stream.php**, що дозволяє отримувати відповіді в режимі реального часу та динамічно відображати їх у чаті.

У цілому, **script.js** є фундаментальним для інтерактивності користувацького інтерфейсу системи миттєвої підтримки, забезпечуючи зручність та ефективність взаємодії користувачів зі службою підтримки.

У процесі розгляду реалізації системи миттєвої підтримки виявлено, що вона ефективно інтегрує різноманітні технології та підходи, включаючи серверну логіку на PHP, взаємодію з базою даних SQLite, інтеграцію з OpenAI для обробки запитів з використанням ШІ, та динамічний фронтенд на JavaScript. Модульна структура і асинхронна взаємодія забезпечують високу гнучкість і швидкість роботи системи, що робить її здатною задовольняти запити користувачів в реальному часі та підтримувати високий рівень взаємодії.

3.5 Оптимізація та тестування ефективності системи

Оптимізація та тестування ефективності системи миттєвої підтримки користувачів передбачає систематичний підхід до вимірювання та аналізу ключових показників продуктивності (KPIs). Цей процес дозволяє оцінити поточний стан системи, визначити потреби в оптимізації та оцінити ефективність внесених змін. Ось детальний опис процесу на основі трьох основних KPI:

3.5.1 Визначення ключових показників ефективності (KPI)

Час відгуку системи (RT - Response Time)

Час відгуку – це час, який проходить між надсиланням запиту користувачем та отриманням відповіді від системи. Цей показник критично важливий для користувацького досвіду, адже користувачі віддають перевагу системам, які відповідають швидко.

Формула для розрахунку:

$$RT = \frac{Total\ time\ for\ N\ Requests}{N}$$

де N – кількість запитів.

Пропускна спроможність системи (TP - Throughput)

Пропускна спроможність вимірює кількість запитів, які система може обробити за певний час. Вища пропускна спроможність означає, що система здатна ефективно обслуговувати більшу кількість користувачів.

Формула для розрахунку:

$$TP = \frac{Total\ Number\ of\ Requests}{Total\ Time}$$

Використання ресурсів сервера (RU - Resource Utilization):

Використання ресурсів сервера включає аналіз використання ЦП, пам'яті, дискового простору та мережевих ресурсів. Ефективне використання ресурсів забезпечує стабільність системи та її масштабованість.

Формула для оцінки загального використання ресурсів:

$$RU = \frac{Total\ Resource\ Used}{Resource\ Capacity}\ 100\%$$

3.5.2 Збір та аналіз даних

Для кожного з цих KPI важливо зібрати вихідні дані до впровадження оптимізацій та порівняти їх із даними після оптимізації. Це може бути зроблено через логування системи, спеціалізоване програмне забезпечення для моніторингу продуктивності, а також через навантажувальне тестування.

3.5.3 Впровадження оптимізації та подальше тестування

Після ідентифікації "вузьких місць" і визначення потенційних напрямків для покращення, розробляються і впроваджуються конкретні заходи оптимізації. Це може включати зміни в архітектурі системи, оптимізацію коду, перегляд стратегій кешування, а також реорганізацію бази даних.

Етапи оптимізації:

- *Оптимізація коду:* Перевірка коду на наявність неефективних циклів, зайвих обчислень та вдосконалення алгоритмів.
- *Оптимізація бази даних:* Індексація таблиць, ревізія запитів до бази даних, нормалізація або денормалізація даних для покращення швидкості виконання запитів.
- *Масштабування:* Впровадження горизонтального масштабування шляхом додавання додаткових серверів або використання обчислювальної хмари для розподілу навантаження.

- *Кешування:* Використання кешування для зменшення кількості звернень до бази даних або зовнішніх API, зокрема, впровадження Redis або Memcached.

3.5.4 Аналіз результатів тестування до та після оптимізації

Для ілюстрації ефективності проведених оптимізацій, ми представляємо порівняльні дані ключових показників ефективності (KPI) системи миттєвої підтримки користувачів до та після оптимізаційних заходів. Це дозволяє візуально оцінити зміни в продуктивності системи.

Таблиця 3.1 – Порівняння результатів тестування

Показник	**До оптимізації**	**Після оптимізації**
Час відгуку (RT)	1500 мс	800 мс
Пропускна спроможність (TP)	100 запитів/хв	200 запитів/хв
Використання ЦП сервера (RU)	75%	60%

Порівняльний аналіз результатів тестування до та після оптимізації системи миттєвої підтримки користувачів виявляє значні покращення за ключовими показниками ефективності. Скорочення часу відгуку з 1500 мілісекунд до 800 мілісекунд відображає важливе підвищення швидкості реагування системи на запити користувачів. Це підтверджує, що впроваджені оптимізації сприяли зменшенню затримок у взаємодії, що є критичним для забезпечення задоволення та високого рівня досвіду користувачів, оскільки швидка відповідь є ключовим очікуванням від цифрових сервісів.

Збільшення пропускної спроможності системи з 100 до 200 запитів за хвилину вказує на ефективність оптимізацій у здатності системи обробляти більшу кількість взаємодій одночасно. Це забезпечує системі кращу стабільність та

надійність під час пікових навантажень, що є особливо важливим для підтримання безперервності сервісу без зниження якості обслуговування користувачів.

Покращення використання ЦП сервера, яке знизилося з 75% до 60%, свідчить про більш ефективне розподілення та використання обчислювальних ресурсів. Це покращення сприяє не лише зниженню операційних витрат на утримання системи, але й забезпечує додатковий простір для масштабування сервісу, дозволяючи системі обслуговувати більшу кількість запитів без додаткового навантаження на сервери.

В цілому, проведені оптимізації мали суттєвий позитивний вплив на продуктивність системи миттєвої підтримки, що демонструється відчутними покращеннями у часі відгуку, пропускній спроможності та ефективності використання ресурсів сервера. Ці результати підкреслюють важливість постійного моніторингу, тестування та оптимізації систем для підтримки високої якості сервісу та задоволення користувачів.

3.6 Екранні форми системи

На рисунках 3.7 – 3.12 наведені екранні форми роботи системи:

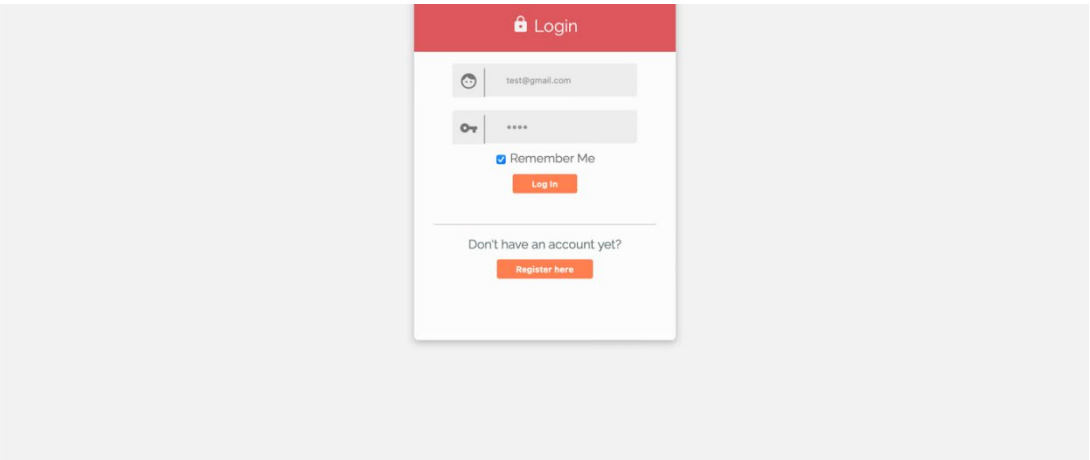


Рисунок 3.7 – Вікно ідентифікації

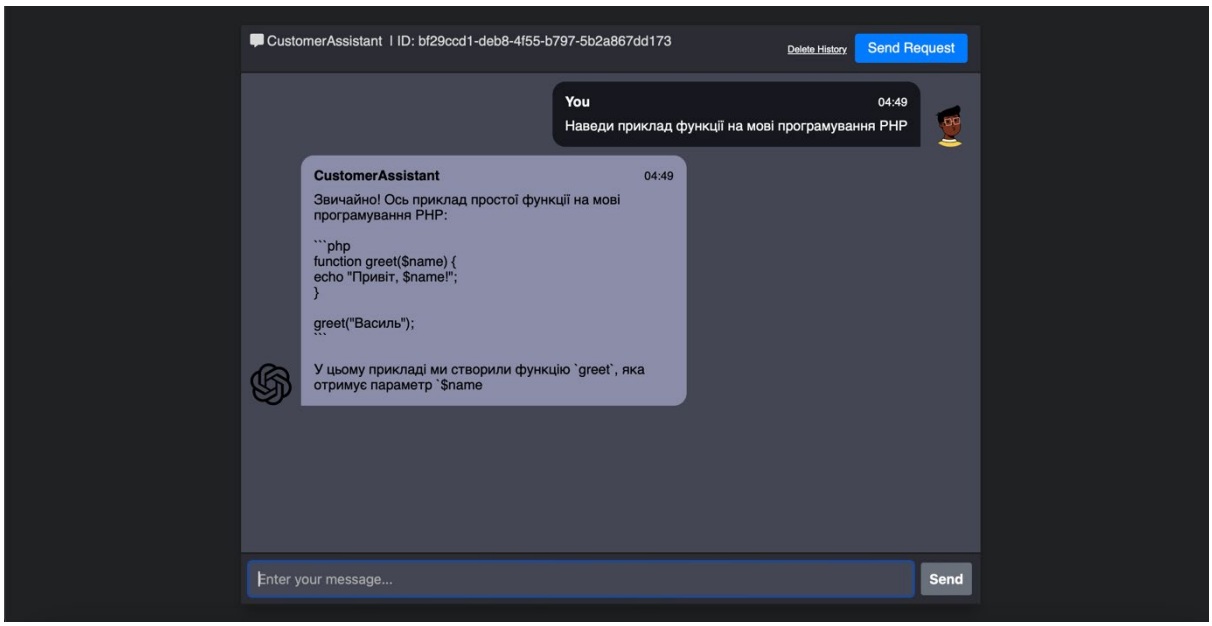


Рисунок 3.8 – Вікно із запитом та відповідью

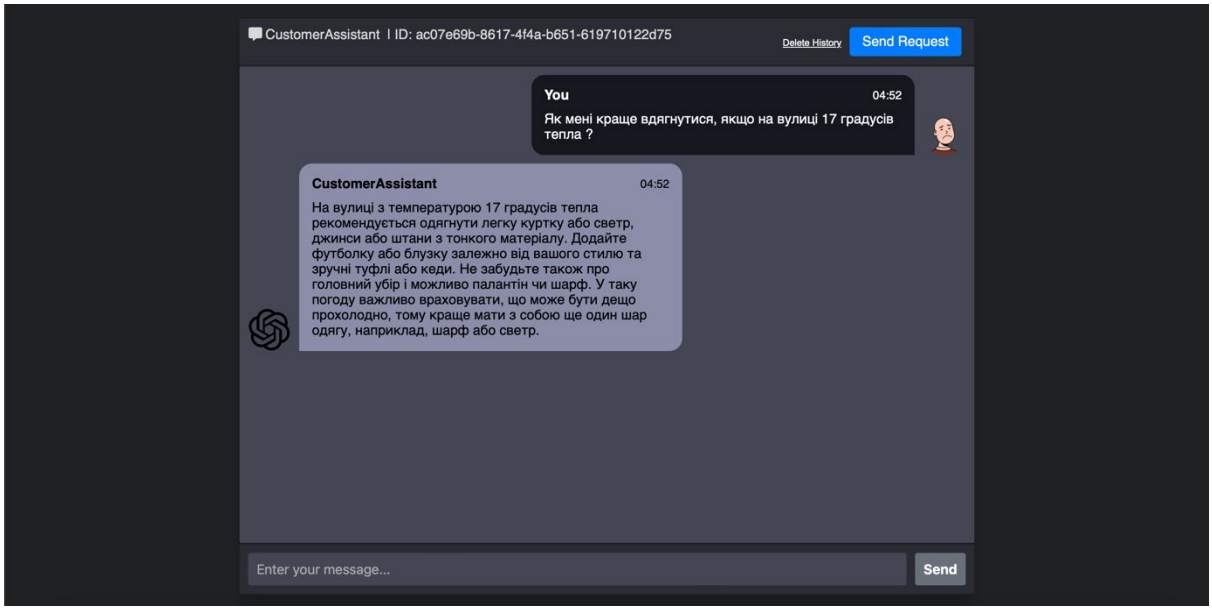


Рисунок 3.9 – Вікно із запитом та відповідью

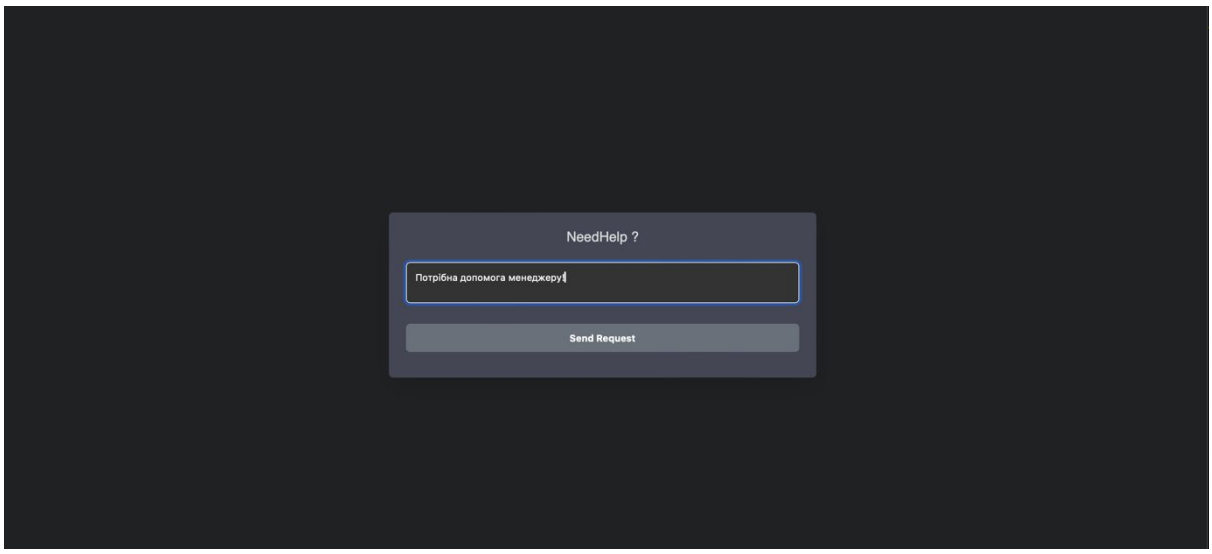


Рисунок 3.10 – Вікно підтримки

Support Requests

User ID	Message	Creation Date	Actions
1	23	2024-04-09 22:35:53	Reply
1	1231232	2024-04-09 22:39:20	Reply
1	Test message	2024-04-09 22:46:12	Reply
1	I want to speak with real person!	2024-04-09 22:51:44	Reply
1	Потрібна допомога менеджеру!	2024-04-10 01:53:36	Reply

Рисунок 3.11 – Вікно запитів користувача

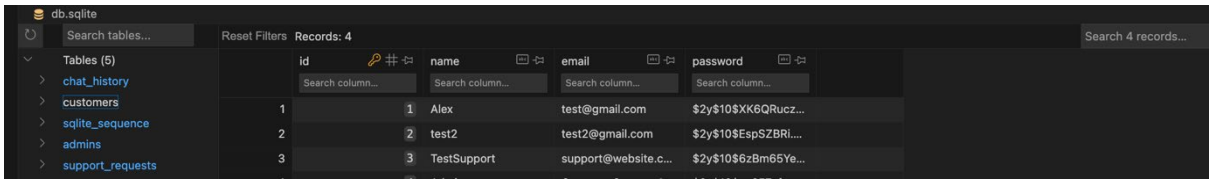


Рисунок 3.12 – Підсумкове вікно запитів

ВИСНОВКИ

Проведено огляд і аналіз існуючих систем миттєвої підтримки, аналіз потреб та вимог користувачів щодо миттєвої підтримки. В результаті аналізу були визначені ключові функції та можливості, які мали б бути доступні в системі.

Спроектована та розроблена система миттєвої підтримки, яка базується на мові програмування PHP та використанні технології Ajax для забезпечення швидкості та інтерактивності. Використання цих технологій дозволило забезпечити високу швидкість обробки запитів та надання відповідей без потреби у перезавантаженні сторінки, що суттєво підвищує зручність користування системою.

Особливу увагу було приділено процесам оптимізації та тестування системи, завдяки чому вдалося значно підвищити її продуктивність. Покращення ключових показників ефективності, таких як час відгуку, пропускна спроможність та використання ресурсів сервера, підтвердили високу ефективність реалізованих оптимізаційних заходів.

Розроблена система миттєвої підтримки має великий потенціал для використання у різних галузях, де необхідно забезпечити швидке та якісне обслуговування користувачів. Можливості подальшого розвитку та удосконалення системи, зокрема через інтеграцію з розширеними алгоритмами штучного інтелекту та розробку адаптивного інтерфейсу, відкривають нові перспективи для покращення якості обслуговування та задоволення потреб користувачів.

Результати даної роботи будуть корисні для компаній, які надають онлайн-сервіси та продукти, оскільки ефективна система миттєвої підтримки дозволить підвищити задоволеність користувачів та зменшити час вирішення їхніх проблем.

В цілому, результати виконаної роботи свідчать про важливість постійного пошуку нових технологічних рішень і методів оптимізації для підвищення ефективності систем миттєвої підтримки. Вони також підкреслюють значення комплексного підходу до розробки та впровадження інновацій, що включає не

тільки застосування новітніх технологій, але й увагу до потреб користувачів та особливостей їхньої взаємодії з системою.

ПЕРЕЛІК ПОСИЛАНЬ

1. Stuttard D., Pinto M. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws. NY: Wiley, 2020. 912 c. ISBN-13: 978-1119687359.

2. Snyder C., Southwell M., Owad T. PHP Security. CA: O'ReillyMedia, 2005. 428 c. ISBN-13: 978-0596006563.

3. Ballad T., Confer W. Securing PHP Web Applications. MA: Syngress, 2008. 304 c. ISBN-13: 978-1597495081.

4. Barnett R.C. Preventing Web Attacks with Apache. NJ: Pearson Education, 2006. 448 c. ISBN-13: 978-0321321289.

5. Schlossnagle G. Advanced PHP Programming. NJ: Pearson Education, 2004. 224 c. ISBN-13: 978-0672325614.

6. McDonald M., McGovern J., et al. Web Security for Developers: Real Threats, Practical Defense. CA: O'Reilly Media, 2022. 542 c. ISBN-13: 978-1492079515.

7. Snyder C., Myer T., Southwell M. Pro PHP Security: FromApplication Security Principles to the Implementation of XSS Defenses. NY: Apress, 2006. 704 c. ISBN-13: 978-1590595084.

8. Bergmann S., Priebsch S. Foundations of PHP for Web Developers. NY: Apress, 2015. 372 c. ISBN-13: 978-1484210438.

9. Zandstra M. PHP Objects, Patterns, and Practice. NY: Apress, 2013. 536 c. ISBN-13: 978-1430259278.

10. Shiflett C. Essential PHP Security. CA: O'Reilly Media, 2005. 100 c. ISBN-13: 978-0596006563.

11. Bergmann S., Priebsch S. PHPUnit Pocket Guide. CA: O'ReillyMedia, 2005. 120 c. ISBN-13: 978-0596101039.

12. Ullman L. PHP and MySQL for Dynamic Web Sites. CA: PeachpitPress, 2017. 696 c. ISBN-13: 978-0134301846.

13. Nixon R. Learning PHP, MySQL & JavaScript. CA: O'Reilly Media, 2018. 832 c. ISBN-13: 978-1491978917.

14. Sklar D., Trachtenberg A. PHP Cookbook. CA: O'Reilly Media, 2014. 820 c. ISBN-13: 978-1449363758.

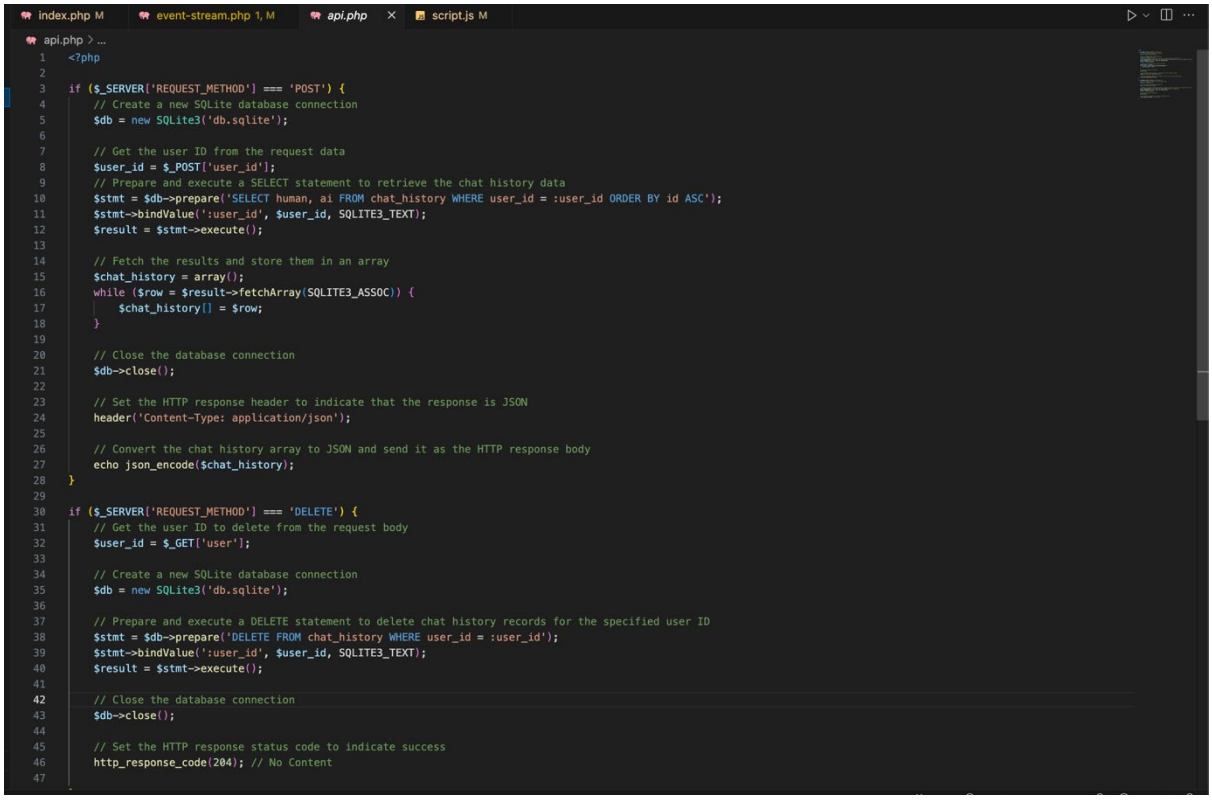
15. Welling L., Thomson L. PHP and MySQL Web Development. CA: Addison-Wesley, 2017. 672 c. ISBN-13: 978-0134804075.

16. Duckett J. PHP & MySQL: Novice to Ninja. UK: SitePoint, 2017. 690 c. ISBN-13: 978-0994346988.

17. Wenz C. PHP Phrasebook. IN: Addison-Wesley, 2005. 480 c. ISBN-13: 978-0672328707.

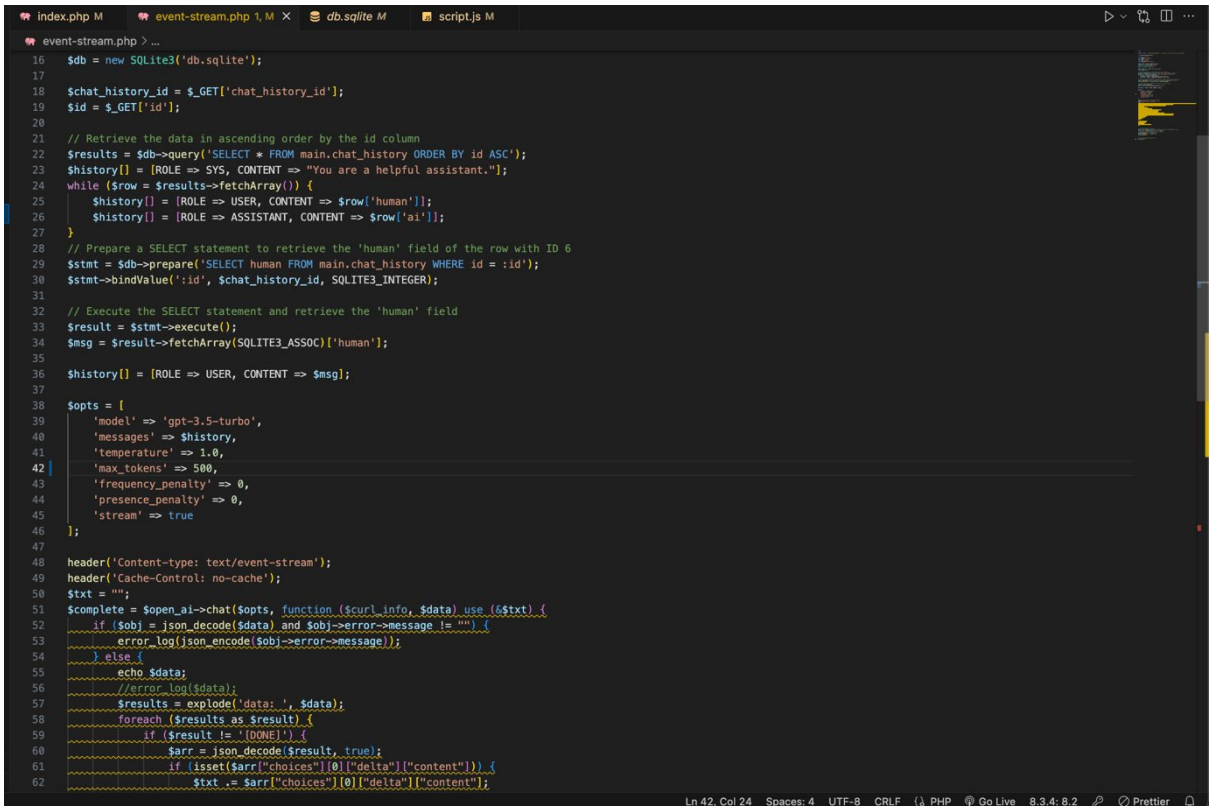
18. Nixon R. Modern PHP: New Features and Good Practices. CA: O'ReillyMedia, 2015. 268 c. ISBN-13: 978-1491905012.

ДОДАТОК А ФРАГМЕНТИ ПРОГРАМНОГО КОДУ



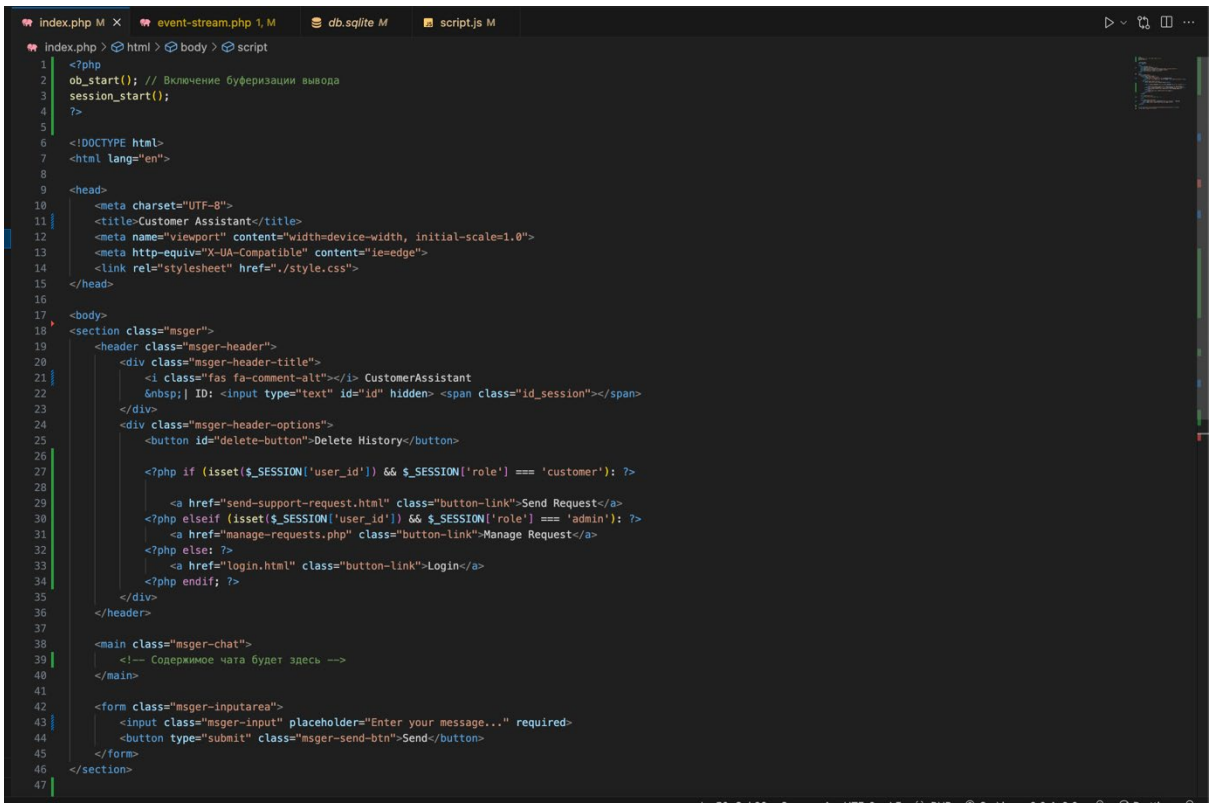
```php
<?php

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Create a new SQLite database connection
    $db = new SQLite3('db.sqlite');

    // Get the user ID from the request data
    $user_id = $_POST['user_id'];
    // Prepare and execute a SELECT statement to retrieve the chat history data
    $stmt = $db->prepare('SELECT human, ai FROM chat_history WHERE user_id = :user_id ORDER BY id ASC');
    $stmt->bindValue(':user_id', $user_id, SQLITE3_TEXT);
    $result = $stmt->execute();

    // Fetch the results and store them in an array
    $chat_history = array();
    while ($row = $result->fetchArray(SQLITE3_ASSOC)) {
        $chat_history[] = $row;
    }

    // Close the database connection
    $db->close();

    // Set the HTTP response header to indicate that the response is JSON
    header('Content-Type: application/json');

    // Convert the chat history array to JSON and send it as the HTTP response body
    echo json_encode($chat_history);
}

if ($_SERVER['REQUEST_METHOD'] === 'DELETE') {
    // Get the user ID to delete from the request body
    $user_id = $_GET['user'];

    // Create a new SQLite database connection
    $db = new SQLite3('db.sqlite');

    // Prepare and execute a DELETE statement to delete chat history records for the specified user ID
    $stmt = $db->prepare('DELETE FROM chat_history WHERE user_id = :user_id');
    $stmt->bindValue(':user_id', $user_id, SQLITE3_TEXT);
    $result = $stmt->execute();

    // Close the database connection
    $db->close();

    // Set the HTTP response status code to indicate success
    http_response_code(204); // No Content
```

Рисунок А.1 – Модуль API.php



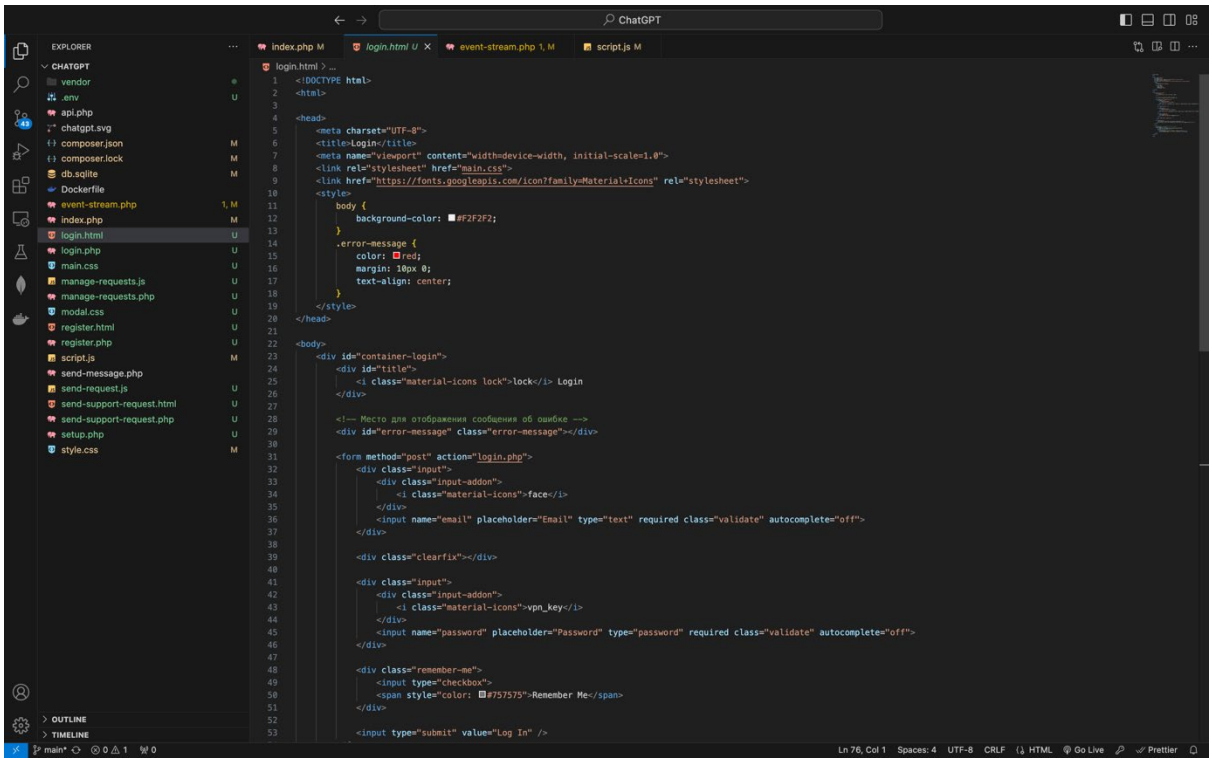
```php
$db = new SQLite3('db.sqlite');

$chat_history_id = $_GET['chat_history_id'];
$id = $_GET['id'];

// Retrieve the data in ascending order by the id column
$results = $db->query('SELECT * FROM main.chat_history ORDER BY id ASC');
$history[] = [ROLE => SYS, CONTENT => "You are a helpful assistant."];
while ($row = $results->fetchArray()) {
    $history[] = [ROLE => USER, CONTENT => $row['human']];
    $history[] = [ROLE => ASSISTANT, CONTENT => $row['ai']];
}
// Prepare a SELECT statement to retrieve the 'human' field of the row with ID 6
$stmt = $db->prepare('SELECT human FROM main.chat_history WHERE id = :id');
$stmt->bindValue(':id', $chat_history_id, SQLITE3_INTEGER);

// Execute the SELECT statement and retrieve the 'human' field
$result = $stmt->execute();
$msg = $result->fetchArray(SQLITE3_ASSOC)['human'];

$history[] = [ROLE => USER, CONTENT => $msg];

$opts = [
    'model' => 'gpt-3.5-turbo',
    'messages' => $history,
    'temperature' => 1.0,
    'max_tokens' => 500,
    'frequency_penalty' => 0,
    'presence_penalty' => 0,
    'stream' => true
];

header('Content-type: text/event-stream');
header('Cache-Control: no-cache');
$txt = "";
$complete = $open_ai->chat($opts, function ($curl_info, $data) use (&$txt) {
    if ($obj = json_decode($data) and $obj->error->message != "") {
        error_log(json_encode($obj->error->message));
    } else {
        echo $data;
        //error_log($data);
        $results = explode('data: ', $data);
        foreach ($results as $result) {
            if ($result != '[DONE]') {
                $arr = json_decode($result, true);
                if (isset($arr["choices"][0]["delta"]["content"])) {
                    $txt .= $arr["choices"][0]["delta"]["content"];
```

Рисунок А.2 – Модуль EVENT-STREAM.php



```php
<?php
ob_start(); // Включение буферизации вывода
session_start();
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Customer Assistant</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="./style.css">
</head>

<body>
<section class="msger">
    <header class="msger-header">
        <div class="msger-header-title">
            <i class="fas fa-comment-alt"></i> CustomerAssistant
             | ID: <input type="text" id="id" hidden> <span class="id_session"></span>
        </div>
        <div class="msger-header-options">
            <button id="delete-button">Delete History</button>

            <?php if (isset($_SESSION['user_id']) && $_SESSION['role'] === 'customer'): ?>

                <a href="send-support-request.html" class="button-link">Send Request</a>
            <?php elseif (isset($_SESSION['user_id']) && $_SESSION['role'] === 'admin'): ?>
                <a href="manage-requests.php" class="button-link">Manage Request</a>
            <?php else: ?>
                <a href="login.html" class="button-link">Login</a>
            <?php endif; ?>
        </div>
    </header>

    <main class="msger-chat">
        <!-- Содержимое чата будет здесь -->
    </main>

    <form class="msger-inputarea">
        <input class="msger-input" placeholder="Enter your message..." required>
        <button type="submit" class="msger-send-btn">Send</button>
    </form>
</section>
```

Рисунок А.3 – Модуль INDEX.php



```html
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="main.css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <style>
        body {
            background-color: #F2F2F2;
        }
        .error-message {
            color: red;
            margin: 10px 0;
            text-align: center;
        }
    </style>
</head>

<body>
    <div id="container-login">
        <div id="title">
            <i class="material-icons lock">lock</i> Login
        </div>

        <!-- Место для отображения сообщения об ошибке -->
        <div id="error-message" class="error-message"></div>

        <form method="post" action="login.php">
            <div class="input">
                <div class="input-addon">
                    <i class="material-icons">face</i>
                </div>
                <input name="email" placeholder="Email" type="text" required class="validate" autocomplete="off">
            </div>

            <div class="clearfix"></div>

            <div class="input">
                <div class="input-addon">
                    <i class="material-icons">vpn_key</i>
                </div>
                <input name="password" placeholder="Password" type="password" required class="validate" autocomplete="off">
            </div>

            <div class="remember-me">
                <input type="checkbox">
                <span style="color: #757575">Remember Me</span>
            </div>

            <input type="submit" value="Log In" />
```

Рисунок А.4 – Модуль LOGIN.html

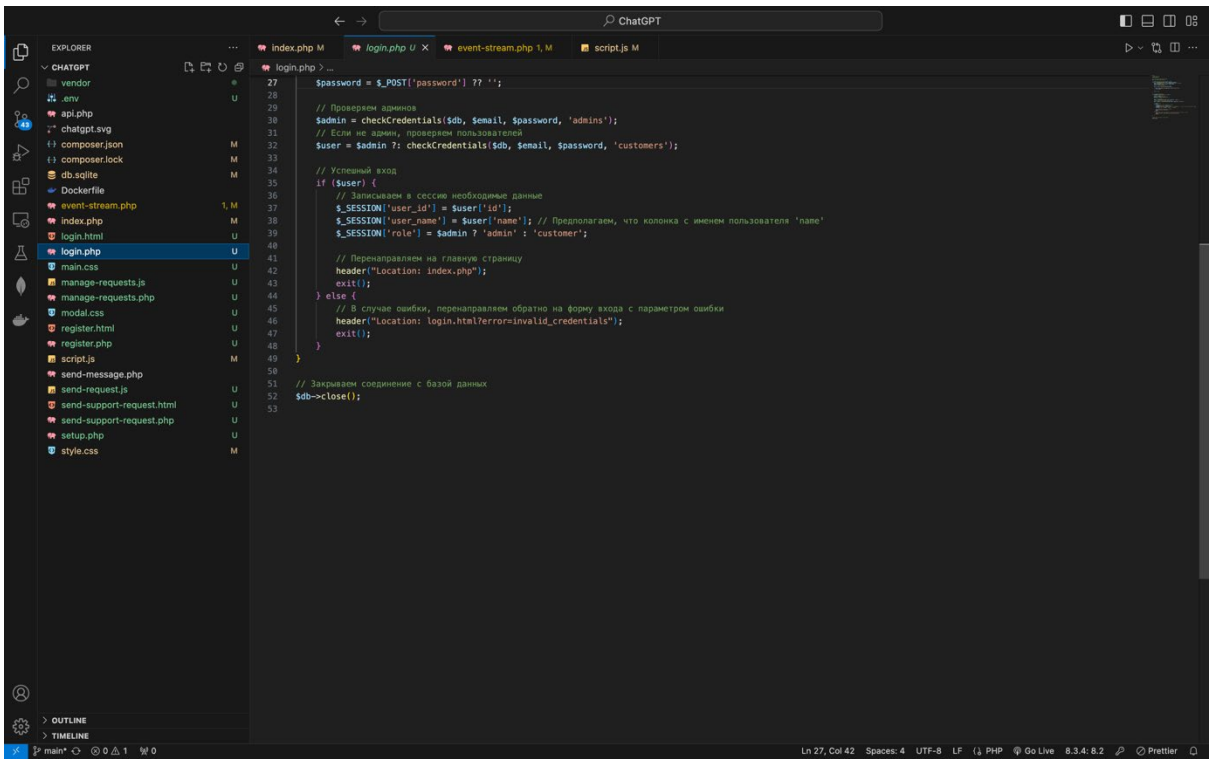


Рисунок А.5 – Модуль LOGIN.php

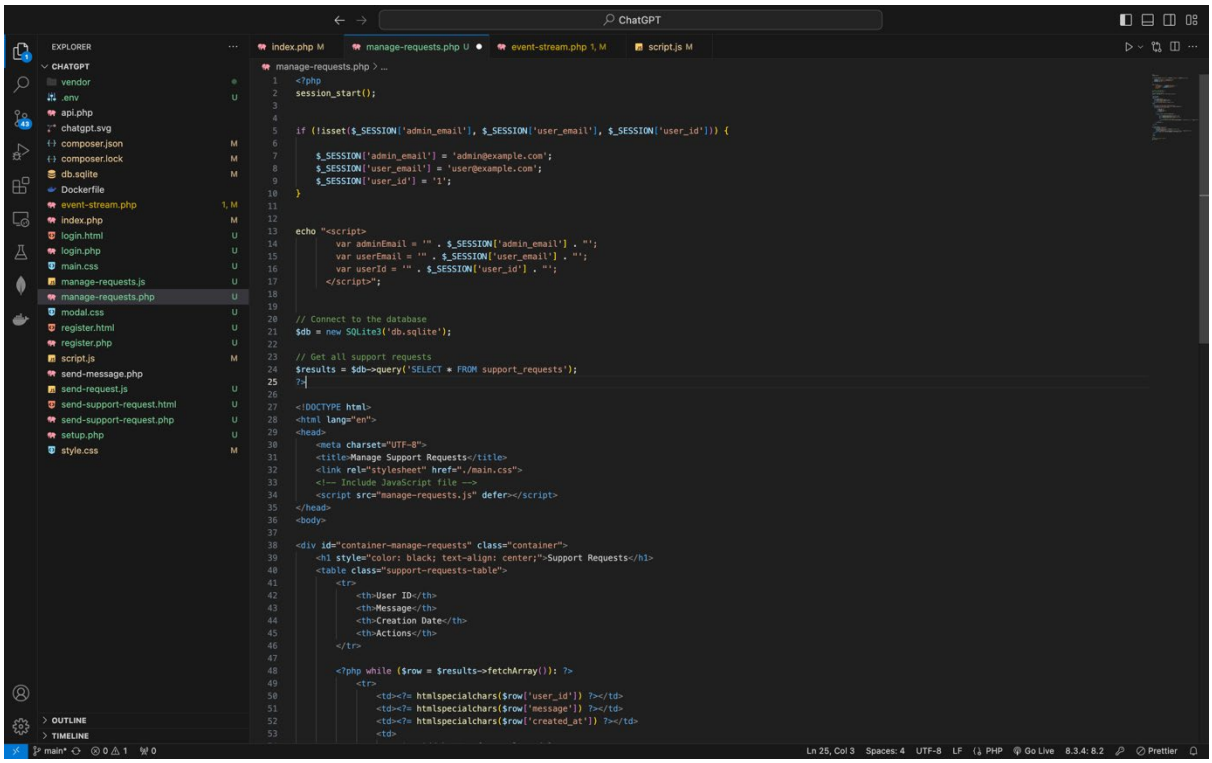


Рисунок А.6 – Модуль MANAGE-REQUESTS.php

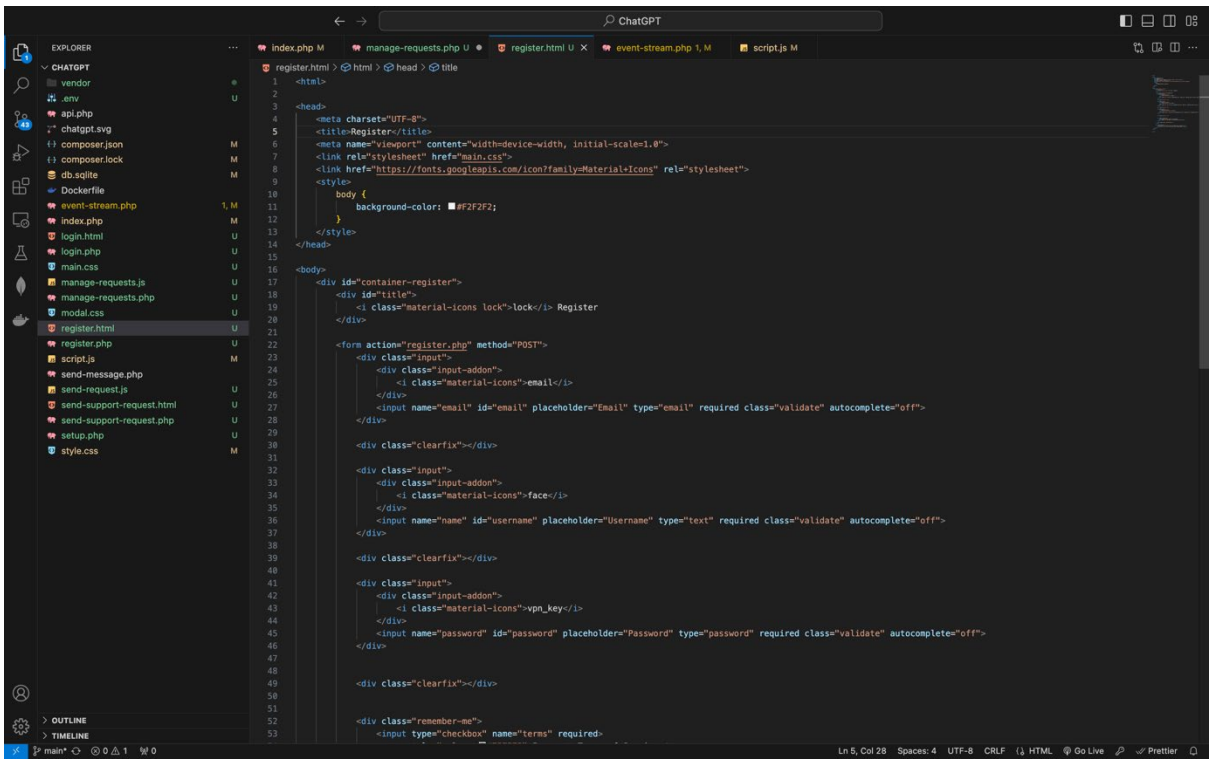


Рисунок А.7 – Модуль REGISTER.html

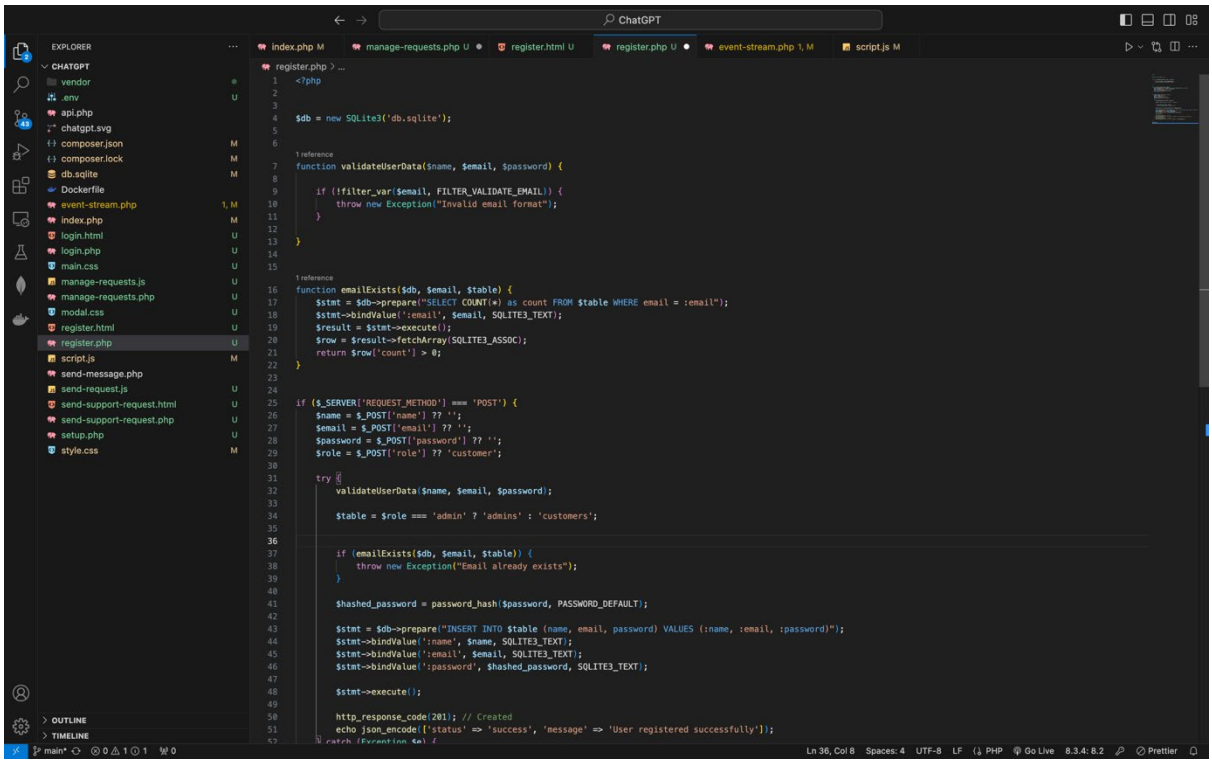


Рисунок А.8 – Модуль REGISTER.php

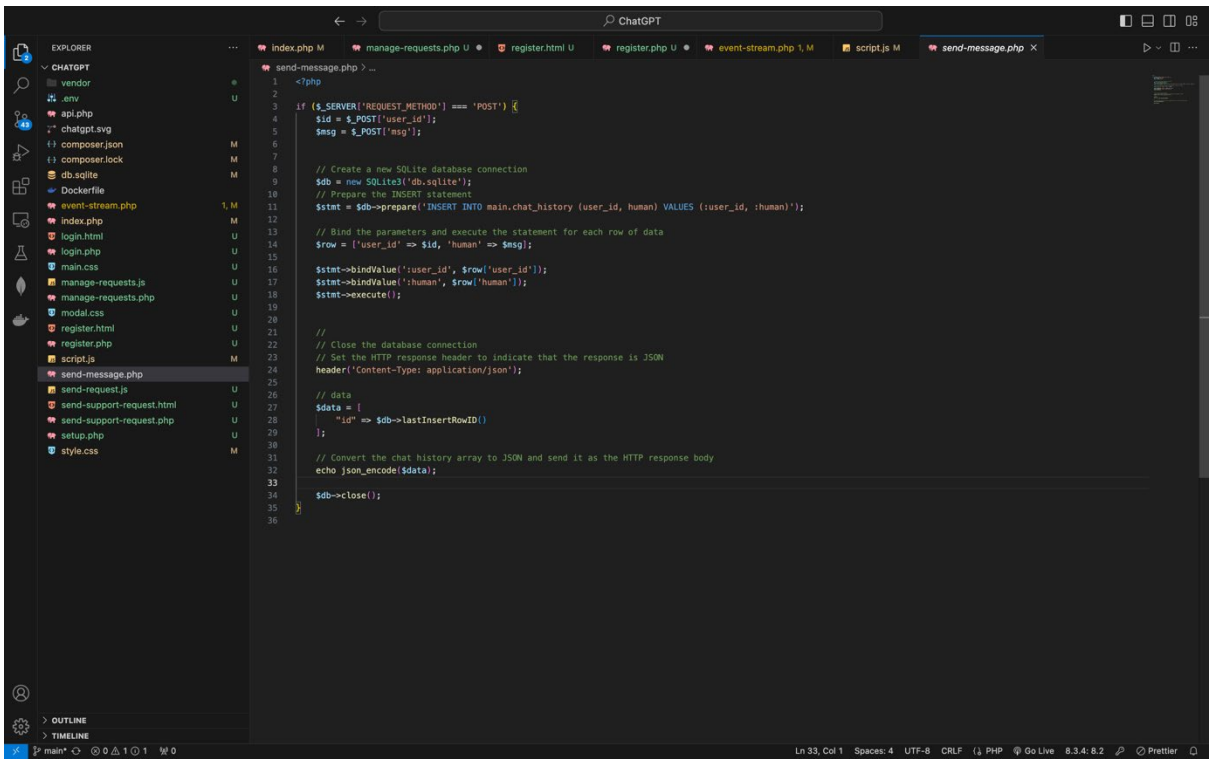


Рисунок А.9 – Модуль SEND-MESSAGE.php

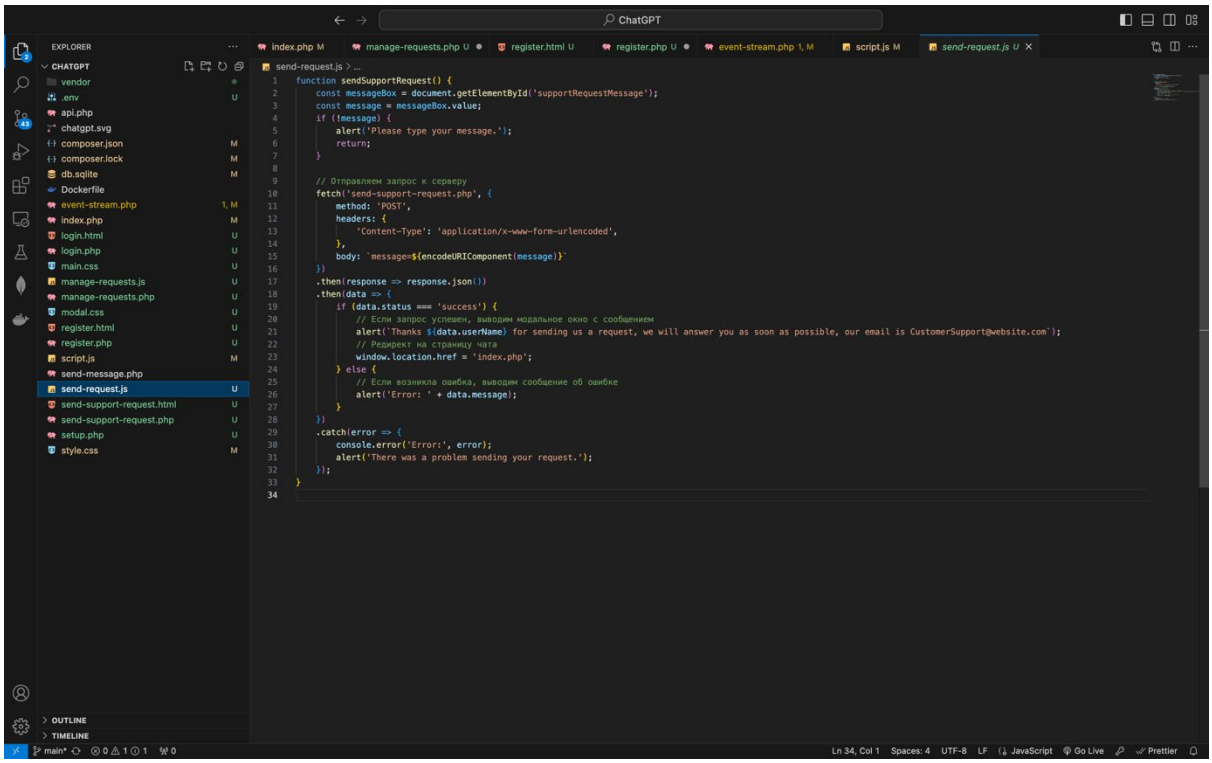


Рисунок А.10 – Модуль SEND-REQUEST.js

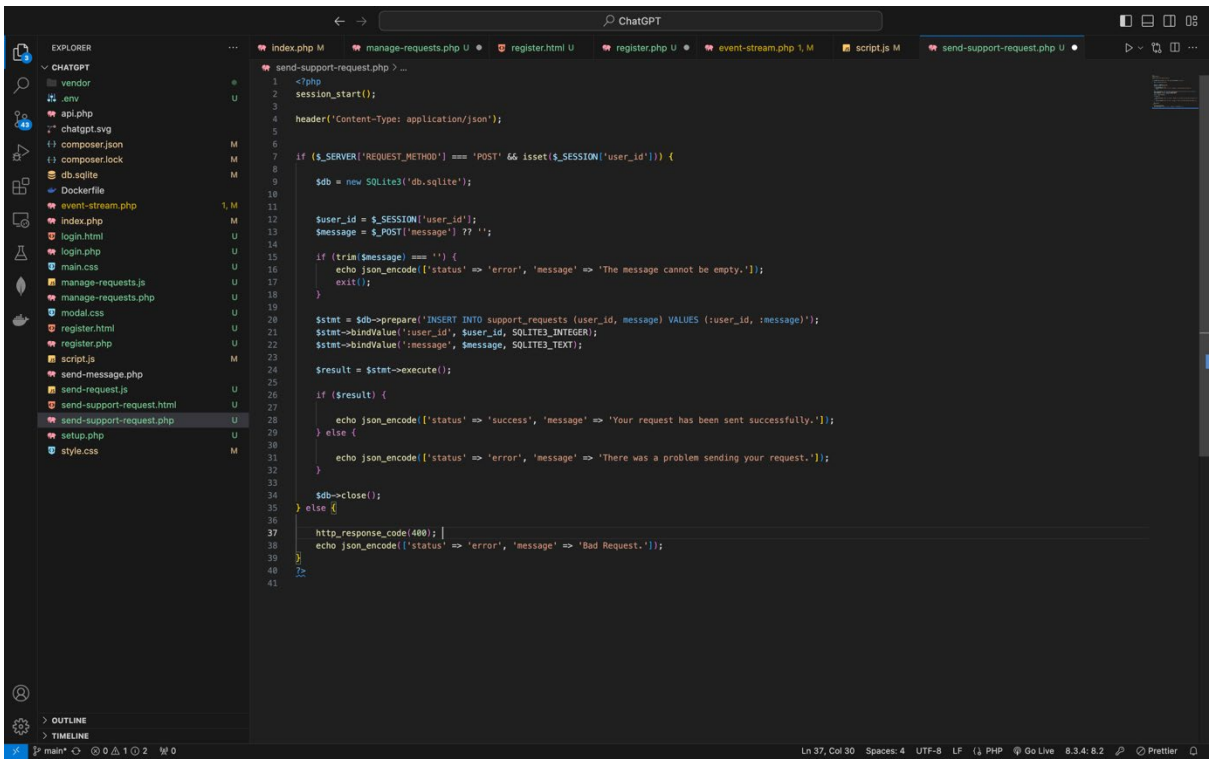


Рисунок А.11 – Модуль SEND-SUPPORT-REQUEST.php

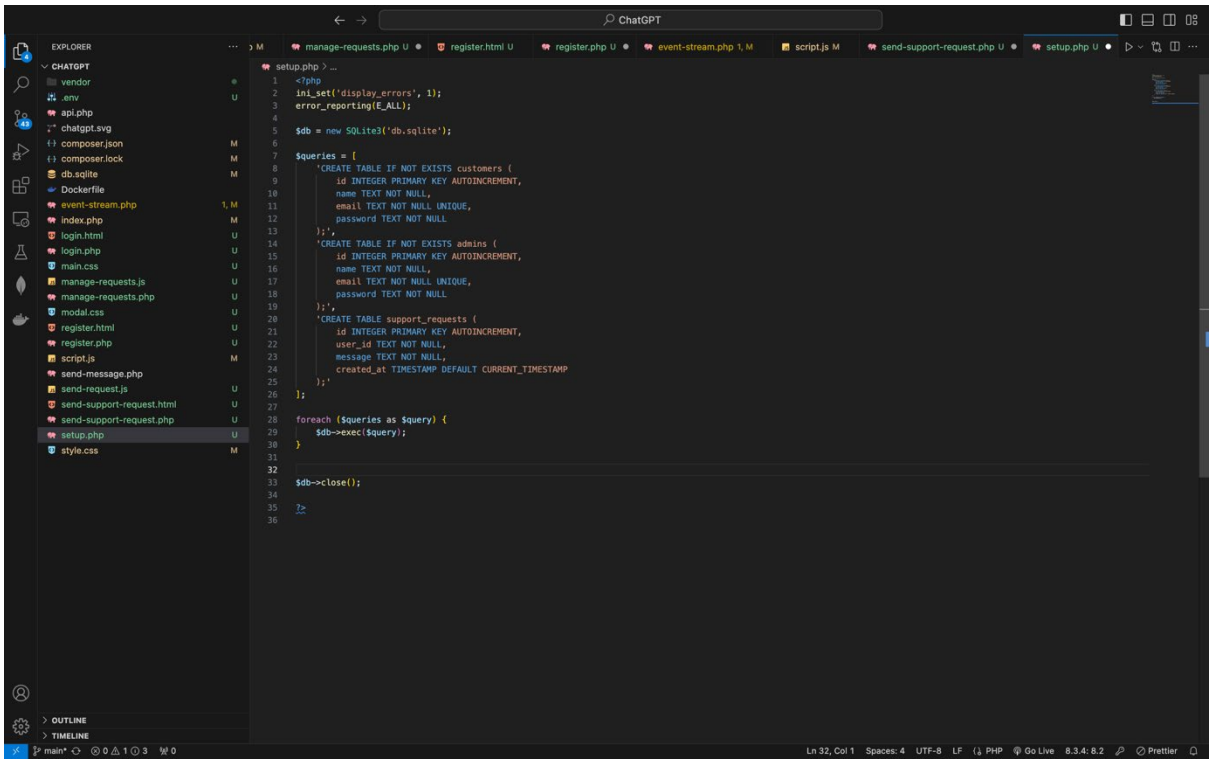


Рисунок А.12 – Модуль SETUP.php

ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерних наук

ДИПЛОМНА РОБОТА
на ступінь вищої освіти бакалавр
із спеціальності 122 Комп'ютерні науки

Розроблення системи миттєвої підтримки користувачів на основі PHP та Ajax

Виконав: студент 5 курсу, групи КНЗ-51
Савончук Денис Володимирович
Керівник: доцент кафедри Комп'ютерних наук
к.т.н., доцент Іщеряков С.М.

Київ - 2024

1

АКТУАЛЬНІСТЬ РОБОТИ

2

В теперішній час в умовах розвитку інтернет-технологій та зростання конкуренції сучасні компанії намагаються забезпечити високоякісну та оперативну підтримку своїх клієнтів.

Тому надзвичайно важливо створити ефективні механізми миттєвої підтримки, які спроможні негайно реагувати на запити користувачів та забезпечувати вирішення їхніх проблем. У існуючих аналогів існує ряд суттєвих недоліків. В зв'язку с цим тема розробленої бакалаврської роботи є *актуальною*.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Об`єкт дослідження – процеси функціонування та взаємодії користувачів з системою миттєвої підтримки.

Предмет дослідження –розробка системи миттєвої підтримки користувачів на основі мови програмування PHP та технології Ajax.

Мета роботи – розробка системи миттєвої підтримки користувачів з використанням сучасних веб-технологій, дослідження можливостей впровадження системи, її функціональних можливостей та переваг перед існуючими рішеннями.

Наукове завдання – огляд та аналіз існуючих систем миттєвої підтримки; аналіз вимог користувачів щодо системи миттєвої підтримки; проектування архітектури системи та її компонентів з урахуванням потреб користувачів; розроблення програмного забезпечення на базі PHP та Ajax з метою забезпечення ефективності та зручності використання; тестування та впровадження системи для перевірки її працездатності та коректності функціонування.

.**Методи дослідження** – мови програмування PHP, JavaScript, технологія AJAX, СУБД SQLite .

1 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ ТА ВИМОГ ДЛЯ СИСТЕМИ

Дослідження існуючих систем миттєвої підтримки

Рисунок 1.1 – Система Life Chat

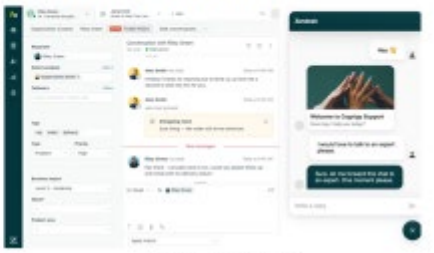


Рисунок 1.2 – Zendesk Chat

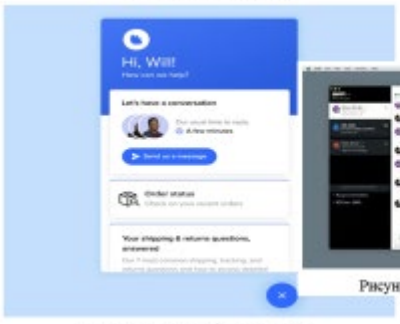


Рисунок 1.3 – Платформа Intercom

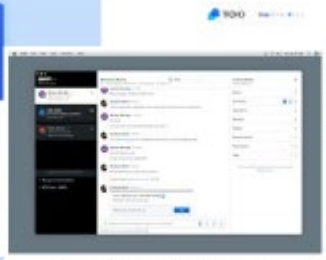


Рисунок 1.4 – Платформа Drift

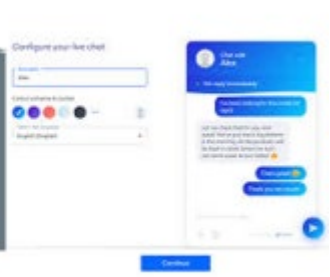


Рисунок 1.5 – Платформа Tidio

1 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ ТА ВИМОГ ДЛЯ СИСТЕМИ

Таблиця 1.1 – Порівняння платформ

Критерій	LiveChat	Zendesk Chat	Intercom	Drift	Tidio Chat
Швидкість відгуку	Висока	Висока	Висока	Висока	Середня
Рівень налаштування	Високий	Високий	Високий	Високий	Середній
Інтеграція з CRM	Так	Так	Так	Так	Ні
Ціна	Висока для SMB	Висока	Висока	Дуже висока	Доступна
Складність налаштування	Низька	Складна для новачків	Висока	Складна	Низька
Інноваційні функції	Ні	Ні	Так	Так	Ні
Персоналізація діалогів	Так	Так	Так	Так	Обмежена
Автоматизація та AI	Обмежена	Обмежена	Висока	Висока	Обмежена
Гнучкість і масштабованість	Висока	Дуже висока	Висока	Висока	Середня
Простота інтеграції	Середня	Середня	Складна	Складна	Висока

1 ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ ТА ВИМОГ ДЛЯ СИСТЕМИ

Вибір напрямків для розробки системи на основі аналізу аналогів

- ***Інтеграційна сумісність*** – Система має легко інтегруватися з різними CRM-системами та іншими інструментами, які вже використовуються на підприємстві.
- ***Вартість впровадження*** – Система повинна бути доступною для широкого спектру бізнесів, передбачати гнучкі тарифні плани та безкоштовний доступ до базових функцій.
- ***Легкість використання*** – Інтуїтивно зрозумілий інтерфейс для операторів та кінцевих користувачів.
- ***Можливість масштабування*** – Система повинна бути гнучкою, витримувати збільшення обсягів запитів та користувачів без втрати продуктивності.
- ***Впровадження інновацій*** – Використання штучного інтелекту для автоматизації відповідей та аналітики даних дозволить знизити навантаження на операторів та забезпечити більш персоналізований підхід до клієнтів.
- ***Безпека та конфіденційність*** – Система має включати надійні засоби захисту інформації.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

Основні складові системи миттєвої підтримки

Спілкувальний інтерфейс – це ключовий елемент системи миттєвої підтримки, який використовується для взаємодії між користувачем та підтримкою

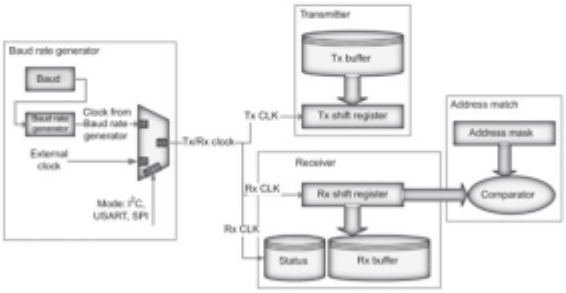


Рисунок 2.1 – Приклад архітектури спілкувального інтерфейсу

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

Функціонал обробки повідомлень в системі миттєвої підтримки відіграє важливу роль у забезпеченні ефективної комунікації між користувачами та підтримкою. Цей компонент відповідає за аналіз та опрацювання повідомлень, що надходять від користувачів, з метою забезпечення оперативної відповіді та вирішення їхніх проблем.

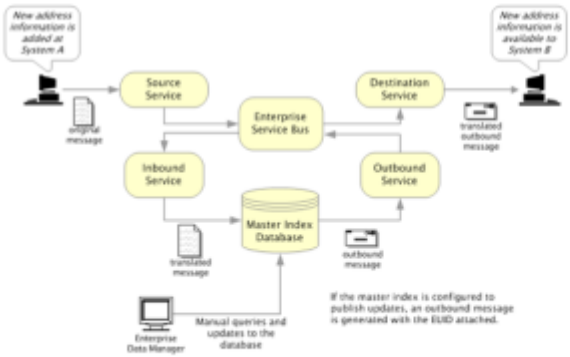


Рисунок 2.2 – Діаграма функціоналу обробки повідомлень

- розпізнавання тексту.
- аналіз семантики повідомлень.
- класифікацію проблем: типові запитання, більш складні проблеми, незвичайних ситуації.
- автоматичні або напівавтоматичні системи відповідей на типові запитання або проблеми.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

Інтеграція системи миттєвої підтримки ***з іншими внутрішніми системами*** компанії може включати такі системи, як бази даних клієнтів, системи управління відносинами з клієнтами (CRM), системи моніторингу та аналізу діяльності користувачів та інші .

Забезпечення безпеки. Для забезпечення конфіденційності, цілісності та доступності даних необхідно застосовувати комплексний підхід до забезпечення безпеки.

- Шифрування даних.
- Захист доступу до системи шляхом використання автентифікації, паролів, біометричних методів або двофакторної аутентифікації.
- Моніторинг активності користувачів.
- Для запобігання кібератакам необхідно вживати заходи забезпечення безпеки мережі.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

Технології розробки системи

JavaScript: Основна мова для розробки клієнтської частини, JavaScript використовується для створення інтерактивних елементів інтерфейсу та забезпечення асинхронної взаємодії з сервером за допомогою Ajax. Динамічне змістове наповнення, анімації, валідація форм — усе це реалізовано на JavaScript.

PHP: Серверна мова програмування PHP задіяна для обробки запитів від користувачів, взаємодії з базою даних і виконання логіки бізнес-операцій. PHP було обрано через його широку підтримку хостинговими провайдерами та гнучкість у розробці веб-додатків.

CSS: Стильова мова CSS застосовується для оформлення веб-сторінок, забезпечуючи їх привабливий та сучасний вигляд.

Dockerfile: Конфігурація Docker дозволяє створювати ізольоване середовище для додатку, що спрощує розгортання та забезпечує однакові умови роботи додатку на різних машинах.

Використання HTTPS, SSL-шифрування, сучасних алгоритмів аутентифікації та авторизації.

Проектування архітектури системи

Архітектура системи миттєвої підтримки втілює в собі гармонійну взаємодію між людиною та машиною, орієнтовану на забезпечення високоякісної та ефективної служби підтримки.

Рисунок 3.1 – Діаграма роботи застосунку

Файлова структура системи

Рисунок 3.2 – Файлова структура додатку

3 ПРОЕКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

Оптимізація та тестування ефективності системи

Оптимізація та тестування ефективності системи миттєвої підтримки користувачів передбачає систематичний підхід до вимірювання та аналізу ключових показників продуктивності (KPIs). Цей процес дозволяє оцінити поточний стан системи, визначити потреби в оптимізації та оцінити ефективність внесених змін.

Ключові показники:

- Час відгуку системи (RT - Response Time)
- Пропускна спроможність системи (TP - Throughput)
- Використання ресурсів сервера (RU - Resource Utilization)

3 ПРОЕКТУВАННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ МИТТЄВОЇ ПІДТРИМКИ

Впровадження оптимізації та подальше тестування

Етапи оптимізації:

- ***Оптимізація коду:*** Перевірка коду на наявність неефективних циклів, зайвих обчислень та вдосконалення алгоритмів.
- ***Оптимізація бази даних:*** Індексація таблиць, ревізія запитів до бази даних, нормалізація або денормалізація даних для покращення швидкості виконання запитів.
- ***Масштабування:*** Впровадження горизонтального масштабування шляхом додавання додаткових серверів або використання обчислювальної хмари для розподілу навантаження.
- ***Кешування:*** Використання кешування для зменшення кількості звернень до бази даних або зовнішніх API, зокрема, впровадження Redis або Memcached.

Аналіз результатів тестування до та після оптимізації

Таблиця 3.1 – Порівняння результатів тестування

Показник	До оптимізації	Після оптимізації
Час відгуку (RT)	1500 мс	800 мс
Пропускна спроможність (TP)	100 запитів/хв	200 запитів/хв
Використання ЦП сервера (RU)	75%	60%

Тобто проведені оптимізації мали суттєвий позитивний вплив на продуктивність системи миттєвої підтримки, що демонструється відчутними покращеннями у часі відгуку, пропускній спроможності та ефективності використання ресурсів сервера.

Екранні форми

Рисунок 3.7 – Вікно ідентифікації

Екранні форми

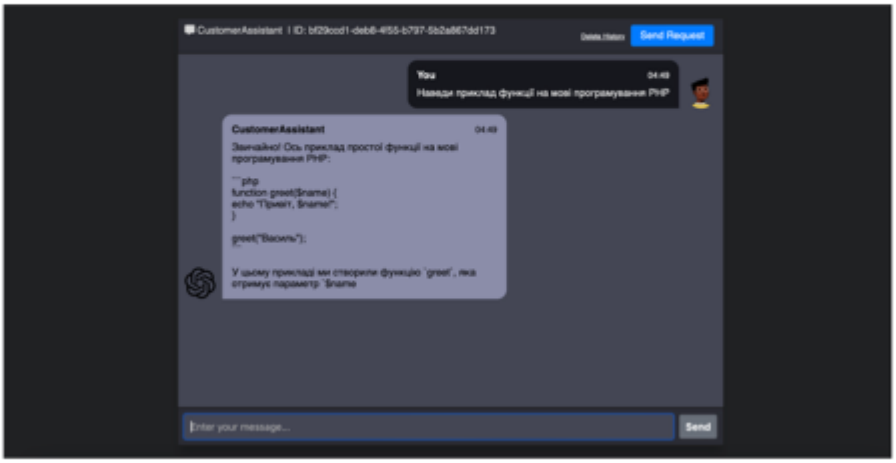


Рисунок 3.8 – Вікно із запитом та відповідью

Екранні форми

Рисунок 3.9 – Вікно із запитом та відповідью

Екранні форми

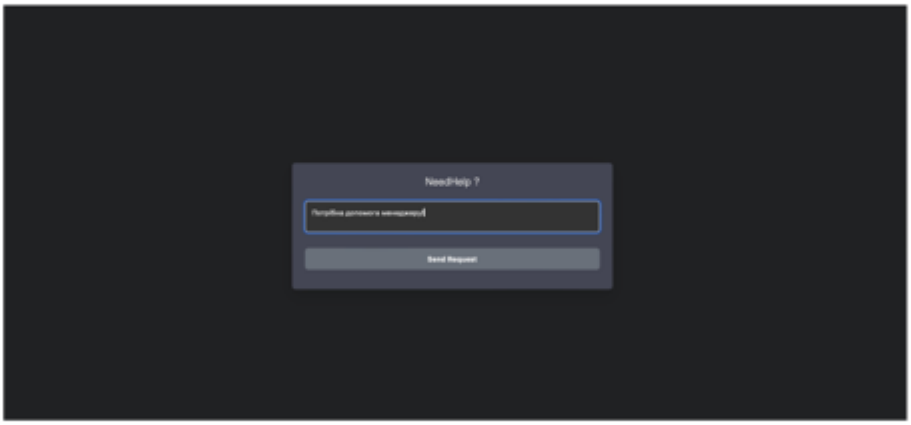


Рисунок 3.10 – Вікно підтримки

Екранні форми

Рисунок 3.11 – Вікно запитів користувача

Рисунок 3.12 – Підсумкове вікно запитів

ВИСНОВКИ

Проведено огляд і аналіз існуючих систем миттєвої підтримки, аналіз потреб та вимог користувачів щодо миттєвої підтримки. В результаті аналізу були визначені ключові функції та можливості, які мали б бути доступні в системі.

Спроектована та розроблена система миттєвої підтримки, яка базується на мові програмування PHP та використанні технології Ajax для забезпечення швидкості та інтерактивності. Використання цих технологій дозволило забезпечити високу швидкість обробки запитів та надання відповідей без потреби у перезавантаженні сторінки, що суттєво підвищує зручність користування системою.

ВИСНОВКИ

Особливу увагу було приділено процесам оптимізації та тестування системи, завдяки чому вдалося значно підвищити її продуктивність. Покращення ключових показників ефективності, таких як час відгуку, пропускна спроможність та використання ресурсів сервера, підтвердили високу ефективність реалізованих оптимізаційних заходів.

Розроблена система миттєвої підтримки має великий потенціал для використання у різних галузях, де необхідно забезпечити швидке та якісне обслуговування користувачів. Можливості подальшого розвитку та удосконалення системи, зокрема через інтеграцію з розширеними алгоритмами штучного інтелекту та розробку адаптивного інтерфейсу, відкривають нові перспективи для покращення якості обслуговування та задоволення потреб користувачів.

ВИСНОВКИ

Результати даної роботи будуть корисні для компаній, які надають онлайн-сервіси та продукти, оскільки ефективна система миттєвої підтримки дозволить підвищити задоволеність користувачів та зменшити час вирішення їхніх проблем.

В цілому, результати виконаної роботи свідчать про важливість постійного пошуку нових технологічних рішень і методів оптимізації для підвищення ефективності систем миттєвої підтримки. Вони також підкреслюють значення комплексного підходу до розробки та впровадження інновацій, що включає не тільки застосування новітніх технологій, але й увагу до потреб користувачів та особливостей їхньої взаємодії з системою.