МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

ЗВІТ

з технологічної практики

«Розробка і супроводження програмного продукту»

на тему Веб-застосунок для вирішення технічних проблем без допомоги першої лінії підтримки

Виконав: здобувач освіти 4 курсу,
групи 45
спеціальності 121
Інженерія програмного забезпечення
Медведь Р.Г.
(прізвище та ініціали)

Керівник _____ Олійник В.В.
(підпис) (прізвище та ініціали)

Полтава – 2024

ЗМІСТ

ВСТУП

1. ПОСТАНОВКА ЗАВДАННЯ

Під час практики в компанії "Аврора", я працював над створенням веб-застосунку, який допоможе співробітникам магазинів самостійно вирішувати поширені технічні проблеми. Завдання, яке було поставлене переді мною, полягало в розробці інтуїтивно зрозумілого інструменту, що дозволяє знайти рішення проблем без необхідності дзвінка до першої лінії технічної підтримки. Це мало суттєво зменшити навантаження на техпідтримку та забезпечити швидкість реагування на типові запити.

Вимоги до продукту включали створення адаптивного інтерфейсу, доступного на комп’ютерах і мобільних пристроях. Застосунок мав сортувати проблеми за категоріями, надавати покрокові інструкції з вирішення, а також відображати контакти технічної підтримки у випадках, коли самостійно розв’язати проблему неможливо. Основними функціональними вимогами були швидкий доступ до бази знань, зручна навігація між категоріями та забезпечення надійності роботи програми.

На реалізацію цього проєкту було відведено чотири тижні. Перший тиждень був присвячений аналізу вимог та проектуванню, під час якого я вивчав потреби співробітників магазинів і розробляв структуру застосунку. Протягом другого та третього тижнів я реалізовував функціональність, використовуючи HTML, CSS та JavaScript, а також фреймворк Bootstrap для створення адаптивного дизайну. Останній тиждень був присвячений тестуванню програми, виправленню помилок і презентації готового продукту.

1.1. Основні вимоги до продукту

Під час розробки веб-застосунку для компанії "Аврора" були визначені функціональні та нефункціональні вимоги, які необхідно було врахувати для створення якісного продукту.

Функціональні вимоги описують, які можливості має надавати програма. Одним із ключових завдань було забезпечення користувачів базою знань для вирішення поширених технічних проблем. Застосунок мав включати категоризацію проблем, щоб користувачі могли швидко знайти відповідь залежно від типу ситуації, наприклад, відсутність інтернету, несправність касового обладнання чи збої в роботі КСО. Крім того, програма мала містити покрокові інструкції з вирішення кожної проблеми та функцію для відображення контактної інформації технічної підтримки на випадок, якщо проблема не вирішується. Для зручності передбачалася можливість повернення до головного меню в один клік і чітка структура інтерфейсу.

Нефункціональні вимоги включали забезпечення швидкої роботи та зручності користування. Одним із пріоритетів було створення адаптивного інтерфейсу, що коректно відображається на різних пристроях, включно з комп’ютерами, планшетами та смартфонами. Застосунок мав бути безпечним, щоб виключити можливість несанкціонованого доступу до даних. Крім того, важливою вимогою була масштабованість, що дозволяє додавати нові категорії проблем та оновлювати базу знань без необхідності серйозного доопрацювання програми. Відмовостійкість теж була врахована: програма має залишатися функціональною навіть у випадках короткочасних збоїв.

Вимоги замовника підкреслювали важливість простоти використання для співробітників магазинів, більшість із яких не мають спеціалізованих технічних знань. Саме тому акцент було зроблено на зрозумілості інтерфейсу, мінімумі зайвих елементів та можливості швидко знайти потрібну інформацію.

Програма була реалізована як веб-застосунок із клієнтською частиною, що працює в браузері. Вона забезпечує зручну навігацію по категоріях технічних проблем, таких як: відсутність інтернету, несправності касового обладнання чи проблеми з КСО (касами самообслуговування). Користувачі можуть швидко знайти потрібну інформацію та слідувати покроковим

інструкціям. Додатково застосунок надає контакти технічної підтримки, якщо проблема не вирішується.

У ході розробки я використовував такі інструменти, як Visual Studio Code для написання коду, Git для контролю версій, а також Bootstrap для швидкої побудови адаптивного інтерфейсу. Розробка була виконана з урахуванням принципів зручності користування, тому інтерфейс програми є максимально простим і зрозумілим.

Під час роботи я вдосконалив свої навички у веб-розробці, зокрема в написанні адаптивного HTML-коду, використанні CSS для створення сучасного дизайну, а також у роботі з JavaScript для динамічних елементів інтерфейсу. Я навчився більш ефективно працювати з Bootstrap і глибше зрозумів принципи побудови користувацьких інтерфейсів.

Що стосується порівняння з аналогічними продуктами, веб-застосунок "Аврора" має перевагу у вузькій спеціалізації. Він розроблений з урахуванням потреб компанії та є більш легким у використанні порівняно з універсальними рішеннями. Програма також легко оновлюється, що дозволяє вносити нові категорії чи оновлення без значних зусиль.

Одним із нереалізованих функціоналів стала інтеграція з базою даних, яка могла б автоматично оновлювати інформацію в застосунку. Це було викликано обмеженими термінами роботи. Однак основна функціональність програми повністю відповідає вимогам замовника, і вона може бути використана в реальній роботі магазинів мережі "Аврора".

1.2. Вимоги до інтерфейсу

Основні вимоги користувачів:

— Чіткий і зрозумілий поділ на категорії проблем.

— Можливість швидкого повернення до головного меню.

— Мінімалістичний дизайн із акцентом на зручність.

GUI-фреймворк: Використано Bootstrap для побудови інтерфейсу. Прийняті рішення: Забезпечення зручного доступу до основних функцій, використання адаптивного дизайну для сумісності з мобільними пристроями.

2. ПЛАНУВАННЯ СИСТЕМИ

У процесі розробки веб-застосунку для компанії "Аврора" я застосував ітеративну модель життєвого циклу програмного забезпечення. Ця модель була обрана через її гнучкість і можливість поетапного вдосконалення функціоналу програми. Використання ітеративного підходу дозволило на кожному етапі отримувати зворотний зв’язок від замовника, щоб вчасно вносити коригування та відповідати реальним потребам бізнесу.

Розробка програми велася відповідно до принципів Agile. Ця методологія дозволила ефективно планувати короткі ітерації, кожна з яких була спрямована на реалізацію окремого набору функцій. Основними перевагами використання Agile були часті демонстрації прогресу замовнику, можливість швидкої адаптації до змін у вимогах та забезпечення прозорості процесу розробки.

Розробка проекту складалася з кількох ключових етапів:

— Аналіз вимог: на цьому етапі я збирав інформацію про потреби компанії "Аврора" та її співробітників. Було проведено аналіз типових проблем, які виникають у роботі магазинів, та визначено основні функціональні та нефункціональні вимоги до продукту.

— Проектування: в ході проектування я створив прототип інтерфейсу користувача, який включав меню категорій, функцію пошуку рішень та інтеграцію контактів техпідтримки. Було також розроблено базову архітектуру програми: клієнт-серверну структуру, де клієнтська частина працює як веб-застосунок.

— Реалізація: на цьому етапі я написав код програми, використовуючи HTML, CSS, JavaScript і фреймворк Bootstrap для створення адаптивного дизайну. Було реалізовано основну функціональність, включно з пошуком рішень, відображенням інструкцій і контактних даних.

— Тестування: після реалізації функціональності я провів тестування програми. Основний акцент був зроблений на перевірці коректності

навігації, адаптивності дизайну та стабільності роботи. Виявлені помилки були виправлені до фінальної версії.

— Презентація та впровадження: останнім етапом було представлення програми замовнику. Після отримання схвалення продукт був переданий для використання у внутрішній мережі компанії.

Для організації роботи над проектом я використовував такі інструменти:

— Trello — для управління завданнями та моніторингу виконання ітерацій. У кожній картці завдання описувався етап розробки, який потрібно завершити.

— GitHub — для контролю версій, збереження коду та спільної роботи. GitHub також використовувався для відстеження змін та історії коду.

— Visual Studio Code — основний редактор для написання та тестування коду.

Ітеративний підхід забезпечив можливість поетапного вдосконалення програми та швидкої адаптації до змін у вимогах. Agile-методологія дозволила ефективно співпрацювати із замовником, регулярно отримувати зворотний зв’язок і своєчасно вносити корективи.

Таким чином, життєвий цикл проекту був побудований з урахуванням потреб компанії, забезпечуючи якісну реалізацію функціональності програми у встановлені терміни.

2.1. Архітектура системи

Розроблений веб-застосунок має клієнт-серверну архітектуру, яка підходить для масштабованих систем і забезпечує чіткий поділ між клієнтською та серверною частинами. Клієнтська частина відповідає за взаємодію з користувачем і відображення інтерфейсу, а серверна (можлива в майбутньому) може бути інтегрована для роботи з базою даних або API.

У розробці застосунку використовувалися наступні шаблони проектування:

— MVC (Model-View-Controller): хоча серверної частини наразі немає, структура коду орієнтована на розділення логіки (Model), інтерфейсу

користувача (View) і можливого контролера для керування діями користувача.

— Шаблон "Одинична відповідальність": кожен модуль (HTML, CSS, JavaScript) відповідає за конкретний аспект функціональності.

— Адаптивний дизайн (Responsive Design): використовувався Bootstrap для забезпечення коректного відображення програми на різних пристроях.

Програма складається з підсистем і компонентів таких як: інтерфейс користувача (UI), навігація, логіка роботи, статика (майбутня інтеграція).

Код програми був організований таким чином, щоб забезпечити його зрозумілість, легкість у підтримці та можливість подальшого розширення. Основна структура коду поділена на три частини: HTML, CSS та JavaScript, кожна з яких відповідає за окремі аспекти роботи веб-застосунку.

HTML використовується для структурування сторінки. Всі елементи розміщені у логічних секціях, таких як заголовок, контейнер із кнопками та блоки з текстовими підказками. Це дозволяє легко додавати нові категорії або змінювати існуючі. HTML-код максимально простий, що полегшує його читання та редагування.

CSS відповідає за стиль і адаптивність інтерфейсу. Для кожного елемента використані класи, які задають конкретні стилі, такі як кольори кнопок, розміри тексту, відступи та поведінка при наведенні. Дизайн реалізований із використанням принципів адаптивності, щоб забезпечити зручність використання програми на різних пристроях.

JavaScript реалізує динамічну поведінку веб-застосунку. Основний функціонал включає:

— Відображення підменю при натисканні на кнопку.

— Приховування підменю, коли відкривається інша категорія.

— Відображення текстових рішень після вибору конкретної кнопки в підменю.

Код JavaScript організований у вигляді функцій, кожна з яких виконує окреме завдання. Наприклад, функція toggleSubMenu() відповідає за відкриття

та закриття підменю, а функція showText() — за відображення відповідного тексту. Такий підхід робить код модульним і зрозумілим.

Особливу увагу було приділено простоті логіки. Усі взаємодії з елементами сторінки реалізовані через методи роботи з DOM. Це дозволяє легко відслідковувати, як кожна функція змінює стан елементів.

Для полегшення розширення програми передбачена можливість додавання нових категорій проблем і рішень без необхідності змінювати основну логіку коду. Достатньо створити нові блоки HTML із кнопками та текстами, а також додати їх до відповідних списків у JavaScript.

Код повністю коментований, що дозволяє швидко зрозуміти його функціонал навіть без попереднього знайомства з проектом.

Результатом такого підходу є структура коду, яка не лише відповідає вимогам поточного проекту, а й дозволяє його легко адаптувати до майбутніх змін або додаткових вимог.

Для деталізації архітектури створені такі UML-діаграми:

— Діаграма класів: відображає структуру основних компонентів, включаючи класи для логіки відображення підменю та взаємодії з користувачем.

— Діаграма діяльностей: описує потік роботи користувача від вибору категорії проблеми до отримання рішення.

— Діаграма послідовностей: демонструє порядок взаємодії між користувачем, інтерфейсом і логікою роботи JavaScript.

Програма була розгорнута на платформі GitHub Pages, що забезпечує доступ до неї через веб-браузер без необхідності налаштування серверної частини. Посилання на розгорнутий проєкт: Демо-версія програми.

Програма розроблена таким чином, що її можна легко доповнити:

— Додати інтеграцію з базою даних для автоматичного оновлення рішень.

— Розширити категорії та функціональність.

— Підключити API для отримання інформації від служби техпідтримки в реальному часі.

Таким чином, архітектура продукту забезпечує зручність роботи користувачів та можливість масштабування відповідно до майбутніх потреб компанії "Аврора".

2.2. Тестування

У процесі роботи над веб-застосунком для компанії "Аврора" було проведено комплексне тестування, щоб переконатися у правильній роботі програми, її зручності та відповідності вимогам. Я застосував кілька видів тестування, кожне з яких мало свої цілі та результати.

Перш за все було проведено *юніт-тестування*. Воно полягало у перевірці коректності роботи окремих функцій JavaScript, які відповідають за ключові дії, наприклад відкриття та закриття підменю або відображення текстових інструкцій. Для цього використовувався фреймворк Jest, який дозволяє автоматизувати процес перевірки функцій. Наприклад, я тестував функцію toggleSubMenu, щоб переконатися, що вона коректно відкриває підменю при натисканні кнопки та одночасно закриває інші відкриті підменю.

Далі було проведено *тестування сумісності,* яке перевіряло, чи коректно працює програма в різних браузерах, таких як Google Chrome, Mozilla Firefox і Microsoft Edge. Також я протестував роботу програми на різних пристроях, включаючи комп’ютери, планшети та смартфони. Це дало змогу переконатися, що інтерфейс адаптується до екранів різного розміру, а всі функції працюють однаково стабільно.

Особливу увагу я приділив *тестуванню доступності.* За допомогою інструменту Lighthouse у Google Chrome я перевірив, наскільки зручно користуватися програмою людям із різними потребами. Для цього оцінювалися такі показники, як контрастність кольорів, розмір шрифтів і можливість навігації за допомогою клавіатури або сенсорного екрана. В результаті тестів було виявлено, що деякі кольорові схеми потребують змін для забезпечення більшої читабельності.

Також я виконав *інтеграційне тестування*, яке перевіряло взаємодію між різними компонентами програми. Це включало перевірку роботи кнопок, підменю та текстових блоків у комплексі. Наприклад, я переконався, що після натискання кнопки підменю відкривається потрібний текст, а не випадковий.

Крім того, було проведено *стрес-тестування*, щоб оцінити, як програма працює з великою кількістю категорій і текстових блоків. Я навмисно додав багато додаткових елементів у код, щоб перевірити, чи зберігається стабільність роботи. Виявилося, що при великій кількості кнопок інтерфейс потребував додаткового коригування, оскільки деякі кнопки виходили за межі видимої області екрана.

Під час тестування було виявлено кілька проблем, які вдалося усунути. Наприклад, функція відкриття підменю спочатку не приховувала інші відкриті підменю, що створювало плутанину. Це було виправлено шляхом додавання логіки приховування всіх підменю перед відкриттям нового. Також під час тестування доступності я виявив, що контрастність кольорів деяких елементів була недостатньою, і це було виправлено шляхом зміни кольорової схеми. У результаті стрес-тестування я адаптував верстку, щоб кнопки залишалися в межах екрану навіть при великій їх кількості.

Після тестування програма отримала кілька важливих покращень. Зокрема, було додано кнопку "Назад до меню" для зручності навігації. Також замовник запропонував додати функцію пошуку за ключовими словами, яка стане частиною майбутніх оновлень.

Тестування показало, що програма відповідає більшості вимог, а внесені зміни зробили її ще більш зручною та ефективною для кінцевих користувачів.

2.3. Інструкція з використання системи

Крок 1: Вибір категорії проблеми

Після відкриття програми користувач бачить головне меню з категоріями. Кожна кнопка відповідає певній категорії, наприклад: "Інтернет

не працює", "Проблеми з КСО", "Не грає музика". Для вибору категорії натисніть на відповідну кнопку (Рисунок 1.1).

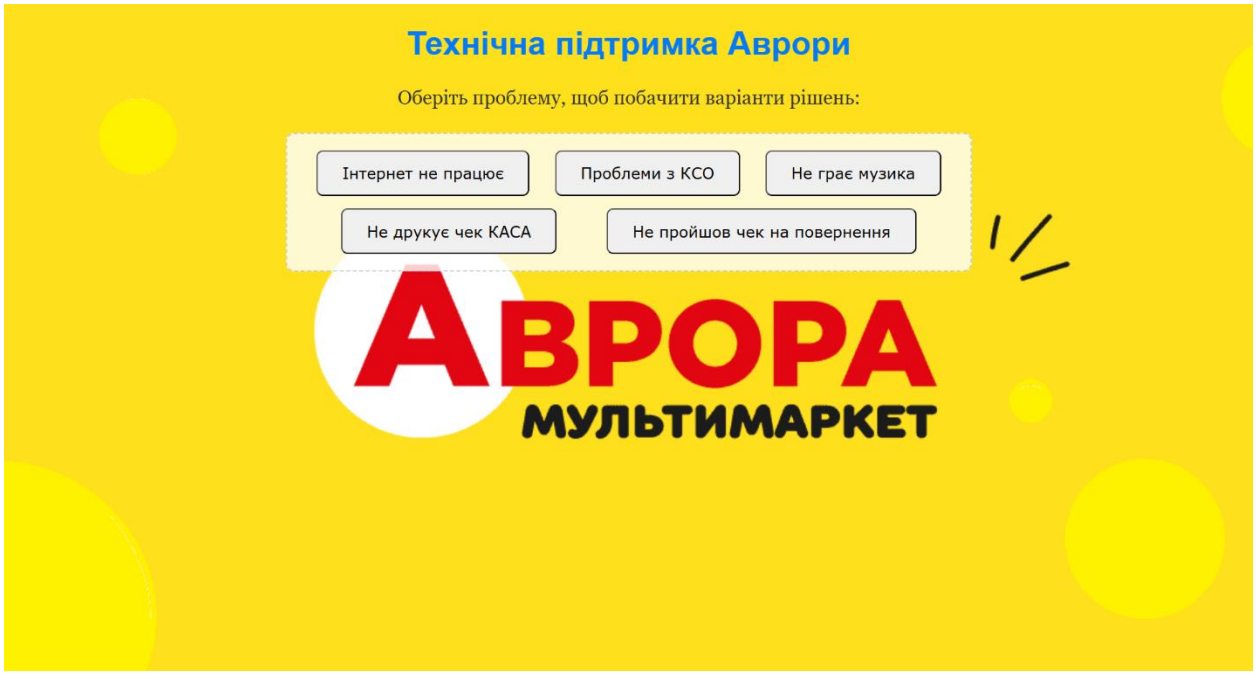

Рисунок 1.1 - Вибір категорії проблеми

Крок 2: Відкриття підменю з можливими рішеннями

Після натискання кнопки відкривається підменю з кількома варіантами дій:

— Можливі причини проблеми.

— Покрокові інструкції вирішення.

— Контактна інформація для техпідтримки.

Виберіть потрібний варіант, натиснувши відповідну кнопку (Рисунок 1.2).

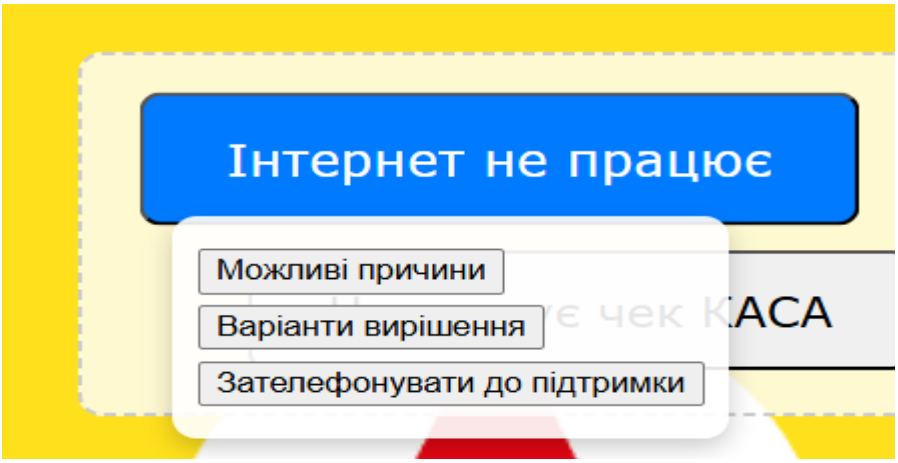

Рисунок 1.2 – Підменю

Крок 3: Ознайомлення з текстовою інструкцією

Після вибору підменю відображається текстовий блок із відповідною інформацією. Наприклад, для "Можливі причини" буде наведено список можливих несправностей, які могли спричинити проблему. Для "Варіанти вирішення" програма запропонує покрокові інструкції (Рисунок 1.3).

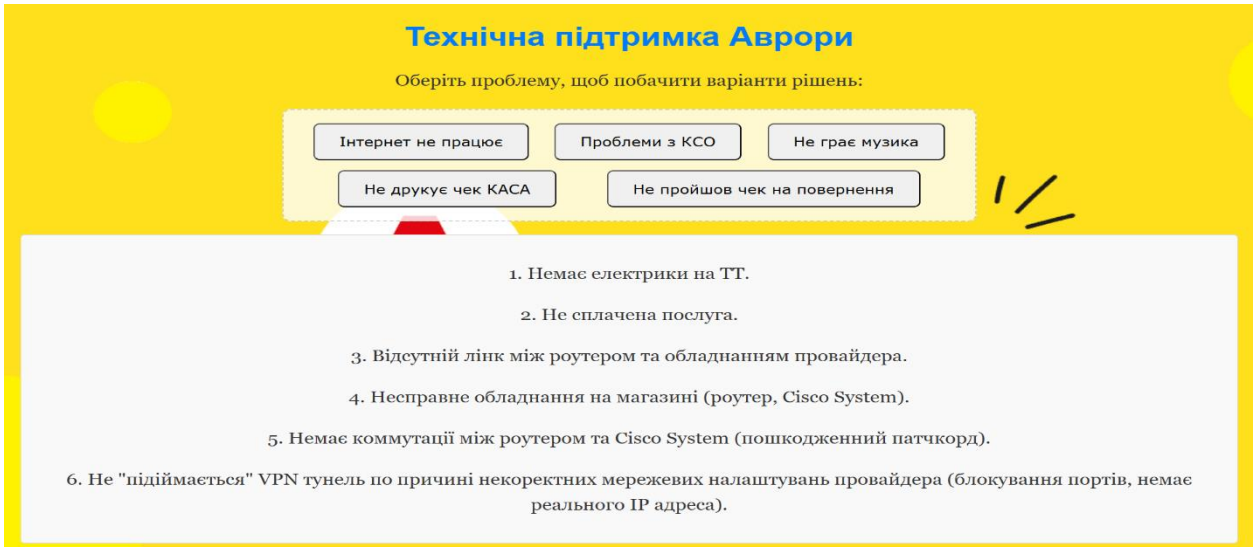

Рисунок 1.3 - Вигляд текстової інструкції “Можливі причини”

Прецедент 1: Користувач хоче дізнатися про можливі причини проблеми

1. Відкрийте програму у веб-браузері.
2. Натисніть кнопку з назвою категорії проблеми, наприклад, "Інтернет не працює".
3. У підменю натисніть кнопку "Можливі причини".
4. Прочитайте список можливих несправностей у текстовому блоці.

Прецедент 2: Користувач шукає покрокові інструкції вирішення

1. Виберіть категорію у головному меню.
2. У підменю натисніть кнопку "Варіанти вирішення".
3. Слідкуйте за покроковими інструкціями, які відобразяться у текстовому полі.

Прецедент 3: Користувач хоче зателефонувати до техпідтримки

1. Виберіть категорію проблеми.
2. У підменю натисніть кнопку "Зателефонувати до підтримки".
3. В текстовому полі відобразиться номер телефону техпідтримки.

ВИСНОВКИ

Під час проходження практики в компанії "Аврора" переді мною було поставлено завдання створити веб-застосунок, який допоможе співробітникам магазинів самостійно вирішувати поширені технічні проблеми. Основна мета полягала у зменшенні навантаження на першу лінію технічної підтримки, а також у прискоренні вирішення типових проблем без дзвінків до техпідтримки.

У ході роботи я проаналізував потреби користувачів, спроєктував інтерфейс і розробив веб-застосунок, який працює на основі HTML, CSS, JavaScript і Bootstrap. Програма забезпечує доступ до покрокових інструкцій вирішення поширених проблем, таких як збої в роботі інтернету, кас самообслуговування або касових апаратів. У разі неможливості вирішення проблеми самостійно користувач отримує контакти технічної підтримки.

В результаті було реалізовано всі основні функціональні вимоги: простий і адаптивний інтерфейс, категоризацію проблем, швидкий доступ до інструкцій і контактів техпідтримки. Однак через обмежені терміни не було додано інтеграцію з базою даних, яка могла б автоматично оновлювати інформацію в застосунку.

Під час роботи я вдосконалив свої навички веб-розробки, зокрема в написанні адаптивного HTML і CSS, програмуванні на JavaScript і використанні фреймворку Bootstrap. Знання, отримані з дисциплін "Основи програмування", "Проектування інформаційних систем" і "Веб-технології", стали основою для виконання цього завдання. Я також здобув практичний досвід роботи з інструментами GitHub і Jest для контролю версій та тестування коду.

Замовник позитивно оцінив готовий продукт і висловив зацікавленість у його подальшому розвитку, зокрема в додаванні нових категорій і функцій, таких як пошук за ключовими словами.

Ця робота стала важливим кроком у моєму професійному розвитку. Я отримав практичний досвід розробки реального продукту, навчився ефективно

працювати з вимогами замовника та застосовувати теоретичні знання в практичних умовах. Надалі я планую вдосконалювати створений застосунок, інтегруючи нові функції та технології, а також використовувати цей досвід у подальшій професійній діяльності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мартін, Р. К. "Чистий код: створення, аналіз та вдосконалення". Харків: Видавництво "Фоліо", 2020.
2. Еріксон, М. "Проектування зручних інтерфейсів: теорія та практика". Київ: Наукова думка, 2019.
3. "HTML Living Standard". WHATWG, https://html.spec.whatwg.org/.
4. "CSS Documentation". Mozilla Developer Network, https://developer.mozilla.org/en-US/docs/Web/CSS.
5. "JavaScript Documentation". Mozilla Developer Network, https://developer.mozilla.org/en-US/docs/Web/JavaScript.
6. "Bootstrap Documentation". Bootstrap Official Website, https://getbootstrap.com/docs/.
7. "Jest Testing Framework". Jest Official Website, https://jestjs.io/docs/.
8. "Lighthouse Documentation". Google Developers, https://developer.chrome.com/docs/lighthouse/.
9. Фаулер, М. "Шаблони корпоративних застосунків". Львів: Видавництво "Літера", 2021.
10. Алонсо, П. "Розробка адаптивних веб-застосунків". Київ: Видавництво "Основа", 2018.

ДОДАТОК А. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ПРОГРАМНИХ ПРОДУКТІВ АНАЛОГІЧНОГО ПРИЗНАЧЕННЯ

	Продукт 1 (Zendesk)	Продукт 2 (Freshdesk)	Створений програмний продукт
Адаптивність	+	+	+
Багатокористувацький режим	+	+	-
Інтуїтивно-зрозумілий інтерфейс користувача	+	+	+
Наявність української мови	-	-	+
Можливість імпорту/експорту даних	+	+	-
Крос-платформність (наявність версій для Linux/MacOS)	+	+	+
Автентифікація з використанням соціальних облікових записів (Google/Facebook/Twitter тощо)	+	+	-
Наявність довідкових матеріалів та документації	+	+	+
Можливості з налагодження та розширення функціональності (Rest API/SDK)	+	+	-
Вартість ліцензії	Безкоштовно	Від 49$ на користувача/місяць	Від 15$ на користувача/місяць

ДОДАТОК Б. UML ДІАГРАМА ПРЕЦЕДЕНТІВ

Рисунок 2.1 - UML Діаграма прецедентів

ДОДАТОК В. ПРОТОТИП ІНТЕРФЕЙСУ

Це прототип інтерфейсу веб-застосунку який був зроблений перед ти як почати писати функціонал й робити косметичні покращення самого сайту.

Вирішив робити на основі цього бо в голові був приблизний результат який не погано виглядає й має дуже простий інтерфейс, так все й вийшло (Рисунок 1.4).

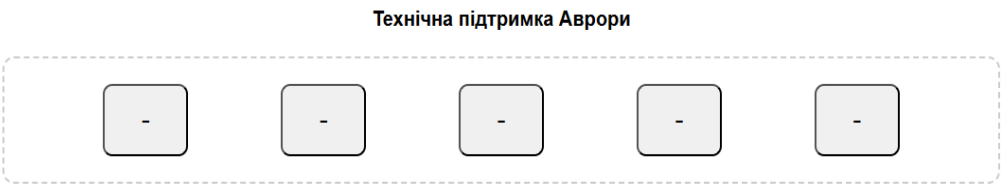

Рисунок 2.2 - Прототип інтерфейсу

ДОДАТОК Г. UML ДІАГРАМА КЛАСІВ

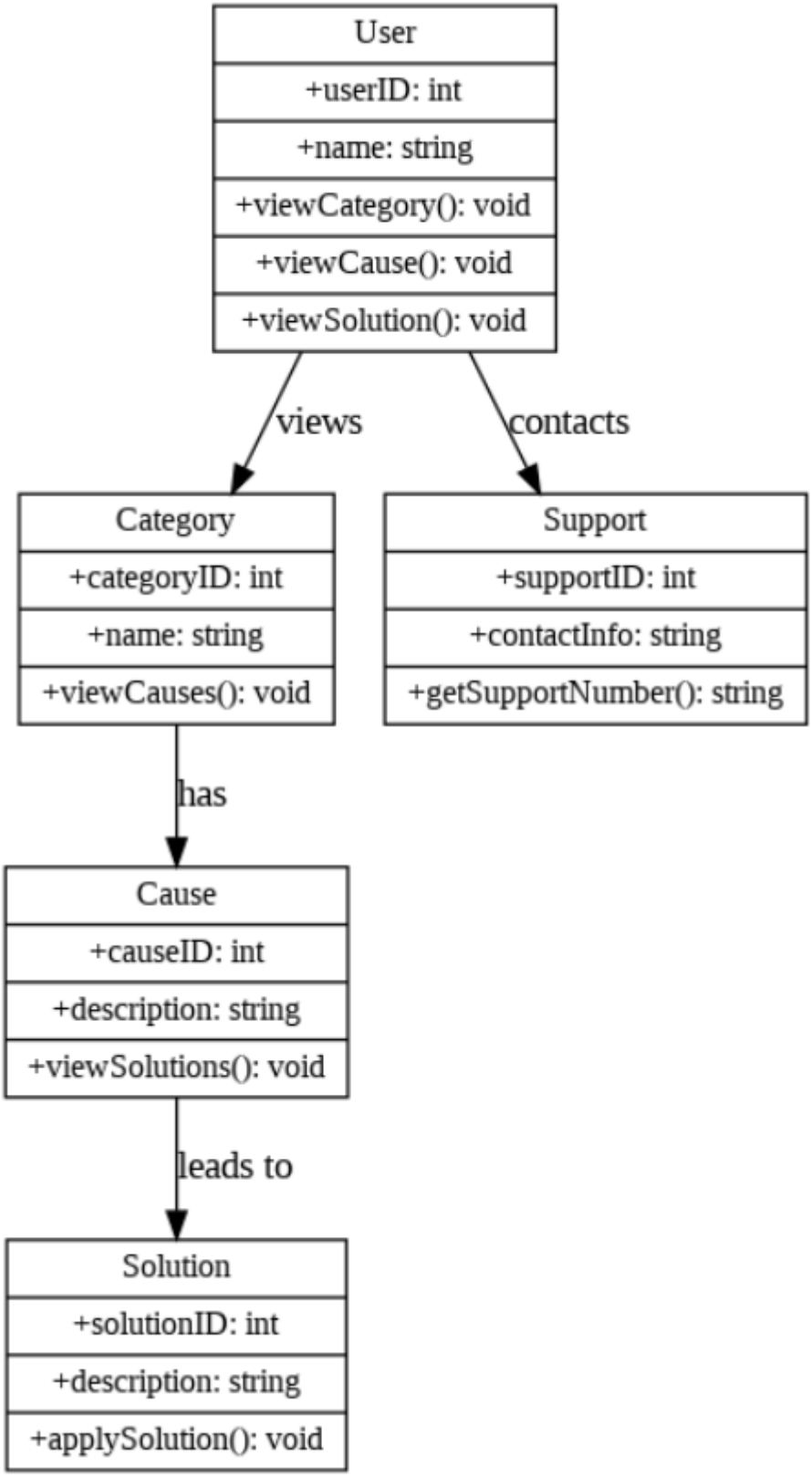

Рисунок 2.3 – UML Діаграма класів

ДОДАТОК Д. СТРУКТУРА САЙТУ

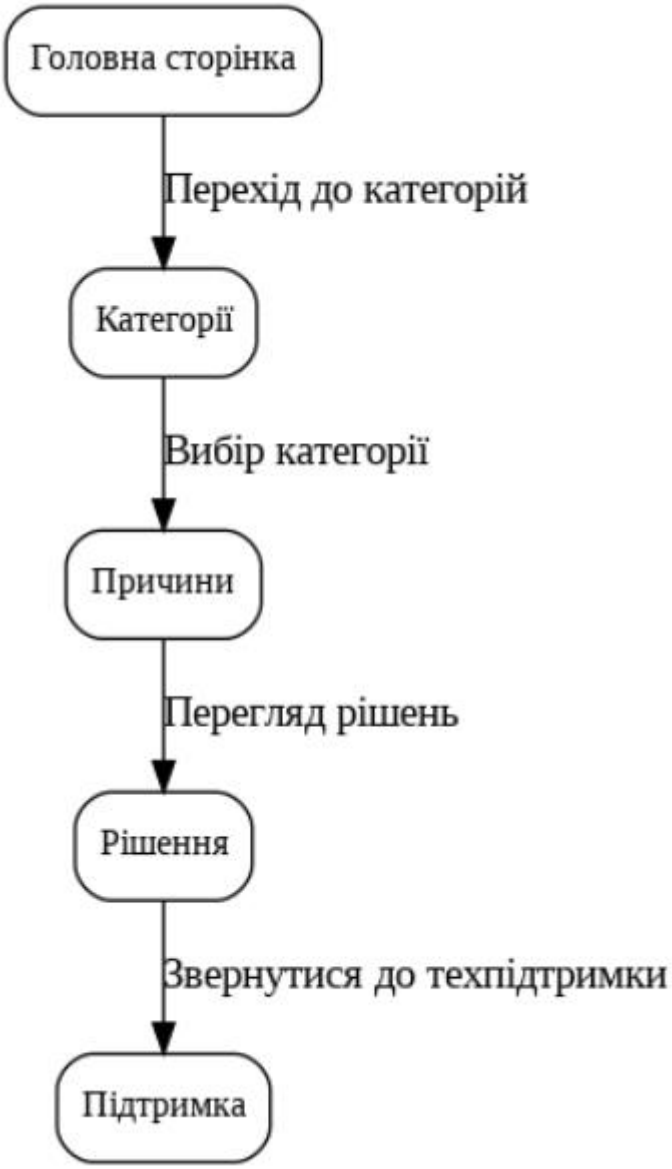

Рисунок 2.3 - Структура Сайту

ДОДАТОК Д. ВИХІДНІ КОДИ

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Технічна підтримка Аврори</title>
  <style>
    body {
      font-family: "Times New Roman", Times, serif;
      margin: 20px;
      color: #343a40;
      background-image: url('unnamed.png');
      background-repeat: no-repeat;
      background-size: cover;
      background-position: center;
      background-attachment: fixed;
    }
    h1 {
      font-family: 'Roboto', sans-serif;
      font-size: 2.5rem;
      font-weight: bold;
      color: #007bff;
      text-align: center;
      margin-bottom: 20px;
    }
    p {
      font-family: 'Georgia', serif;
      font-size: 1.5rem;
      line-height: 1.6;
      text-align: center;
      color: #333;
    }
    .container {
      width: 100%;
```

```
  max-width: 800px;
  margin: 20px auto;
  padding: 20px;
  border: 2px dashed #ccc;
  border-radius: 10px;
  background-color: rgba(255, 255, 255, 0.8);
  display: flex;
  flex-wrap: wrap;
  gap: 15px;
  justify-content: space-evenly;
}
.button-custom {
  font-family: Verdana, sans-serif;
  font-size: 1.2rem;
  padding: 15px 30px;
  border-radius: 8px;
  transition: transform 0.3s ease, background-color 0.3s ease;
}
.button-custom:hover {
  transform: scale(1.1);
  background-color: #007bff;
  color: #fff;
}
.submenu {
  display: none;
  position: absolute;
  background-color: rgba(255, 255, 255, 0.9);
  padding: 10px;
  border-radius: 10px;
  box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
}
.submenu button {
  display: block;
  margin: 5px 0;
}
```

```
    .text-container {
      display: none;
      margin-top: 20px;
      padding: 1%;
      background-color: #f9f9f9;
      border: 1px solid #ddd;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <h1>Технічна підтримка Аврори</h1>
  <p>Оберіть проблему, щоб побачити варіанти рішень:</p>

  <div class="container">
     <button id="btn1" class="btn btn-primary button-custom" onclick="toggleSubMenu(event, 'submenu1')">Інтернет не працює</button>
    <button id="btn2" class="btn btn-secondary button-custom" onclick="toggleSubMenu(event, 'submenu2')">Проблеми з КСО</button>
     <button id="btn3" class="btn btn-success button-custom" onclick="toggleSubMenu(event, 'submenu3')">Не грає музика</button>
     <button id="btn4" class="btn btn-danger button-custom" onclick="toggleSubMenu(event, 'submenu4')">Не друкує чек КАСА</button>
     <button id="btn5" class="btn btn-warning button-custom" onclick="toggleSubMenu(event, 'submenu5')">Не пройшов чек на повернення</button>
  </div>

  <div id="submenu1" class="submenu">
        <button class="btn btn-outline-secondary" onclick="showText('text1')">Можливі причини</button>
        <button class="btn btn-outline-secondary" onclick="showText('text2')">Варіанти вирішення</button>
       <button class="btn btn-danger" onclick="showText('text3')">Зателефонувати до підтримки</button>
  </div>
```

```
 <div id="submenu2" class="submenu">
        <button class="btn btn-outline-secondary" onclick="showText('text4')">Можливі причини</button>
        <button class="btn btn-outline-secondary" onclick="showText('text5')">Варіанти вирішення</button>
       <button class="btn btn-danger" onclick="showText('text6')">Зателефонувати до підтримки</button>
 </div>
 <div id="submenu3" class="submenu">
        <button class="btn btn-outline-secondary" onclick="showText('text7')">Можливі причини</button>
        <button class="btn btn-outline-secondary" onclick="showText('text8')">Варіанти вирішення</button>
       <button class="btn btn-danger" onclick="showText('text9')">Зателефонувати до підтримки</button>
 </div>
 <div id="submenu4" class="submenu">
        <button class="btn btn-outline-secondary" onclick="showText('text10')">Можливі причини</button>
        <button class="btn btn-outline-secondary" onclick="showText('text11')">Варіанти вирішення</button>
       <button class="btn btn-danger" onclick="showText('text12')">Зателефонувати до підтримки</button>
 </div>
 <div id="submenu5" class="submenu">
        <button class="btn btn-outline-secondary" onclick="showText('text13')">Можливі причини</button>
        <button class="btn btn-outline-secondary" onclick="showText('text14')">Варіанти вирішення</button>
       <button class="btn btn-danger" onclick="showText('text15')">Зателефонувати до підтримки</button>
 </div>

 <div id="text1" class="text-container">
  <p>1. Немає електрики на ТТ.</p>
```

2. Не сплачена послуга.

3. Відсутній лінк між роутером та обладнанням провайдера.

4. Несправне обладнання на магазині (роутер, Cisco System).

5. Немає коммутації між роутером та Cisco System (пошкодженний патчкорд).

6. Не "підіймається" VPN тунель по причині некоректних мережевих налаштувань провайдера (блокування портів, немає реального IP адреса).

- Переконайтесь, що кабель підключений до роутера і комп’ютера.

- Перевірити чи всі дроти до кінця підключені. Наприклад: немає лінку між роутером та Cisco Switch - перевірити з'єднання, патч-корд.

- Вимкніть роутер з розетки на 30 секунд, потім увімкніть знову для перезавантаження.

- Переглянте індекатори роутера та маршрутезатора, переконайтесь що все горить вірно.

Телефон підтримки: 0800-123-456.

1. Зависло КСО.

2. Помилка відкриття зміни.

3. Зміна відкрита не сьогодні, тобто не закрили в кінці робочого дня.

4. Не пройшла оплата.

5. Завис чек на КСО.

- Якщо КСО зависло то спробуйте підключитись до неї зі зворотньої сторни за допомогою клавіатури й перезапустити натиснувши комбінацію клавіш ALT+F4 або F5.

- Перевірити чи РРО на зв’язку та чи запущена служба GMSKso Api ServerWPID, якщо так - перевірити що пише в логах на базовому ПК.

- Якщо зміну закривали, перевіряємо логи/зетки.

- Оплата неуспішна – невірний пароль, недостатньо коштів, таймаут, закінчився папір.

```
  <p>- Зупиняємо службу -> Відміняємо даний чек -> Запускаємо службу -> Якщо чек на сайті все ще висить - оновлюємо сторінку.</p>
 </div>
 <div id="text6" class="text-container">
  <p>Телефон підтримки: 0800-654-321.</p>
 </div>
 <div id="text7" class="text-container">
  <p>1. Проблема з Orange(Помаранчова коробка).</p>
  <p>2. Ручка підсилювача.</p>
  <p>3. Проблема з дротами.</p>
  <p>4. Не вімкнувся самостійно.</p>
 </div>
 <div id="text8" class="text-container">
  <p>- Перезавантажити Orange.</p>
  <p>- Змінити кабель аукс.</p>
   <p>- Спробуйте покрутити ручку підсилювача, можливо просто звук стоїть на мінімум.</p>
  <p>- Якщо не заграла музика то пропишіть в спеціальній програмі номер магазина й цифру 1 вибравши внутрішній Orange або цифру 2 якщо не грає наружний.</p>
  <p>- Перевірте всі дроти на підключення й чи вставлені до кінця.</p>
 </div>
 <div id="text9" class="text-container">
  <p>Телефон підтримки: 0800-123-456.</p>
 </div>
 <div id="text10" class="text-container">
  <p>1. Проблема з касою.</p>
  <p>2. Відсутній лінк.</p>
  <p>3. Проблема з ПРРО.</p>
  <p>4. Проблема з інтернет розеткою.</p>
  <p>5. Закінчилась бумага для друку.</p>
 </div>
 <div id="text11" class="text-container">
  <p>- Перезавантажити ПРРО.</p>
  <p>- Перевставити в іншу інтернет розетку.</p>
  <p>- Змінити порт на свічу.</p>
```

```
  <p>- Перевірити живлення в розетці.</p>
</div>
<div id="text12" class="text-container">
  <p>Телефон підтримки: 0800-654-321.</p>
</div>
<div id="text13" class="text-container">
  <p>1. Не справна система.</p>
  <p>2. Помилка при поверненні коштів</p>
</div>
<div id="text14" class="text-container">
  <p>- Дзвонити на лінію роздрібу через 0.</p>
</div>
<div id="text15" class="text-container">
  <p>Телефон підтримки: 0800-654-321.</p>
</div>

<script>
  function toggleSubMenu(event, submenuId) {
    const submenu = document.getElementById(submenuId);

    const allSubmenus = document.querySelectorAll('.submenu');
    allSubmenus.forEach(menu => {
      if (menu !== submenu) {
        menu.style.display = 'none';
      }
    });

    submenu.style.display = submenu.style.display === 'block' ? 'none' : 'block';

    submenu.style.top = `${event.target.offsetTop + event.target.offsetHeight}px`;
    submenu.style.left = `${event.target.offsetLeft}px`;
  }

  function showText(textId) {
    const allTexts = document.querySelectorAll('.text-container');
```

```
      allTexts.forEach(text => {
        text.style.display = 'none';
      });

      const selectedText = document.getElementById(textId);
      if (selectedText) {
        selectedText.style.display = 'block';
      }
    }
  </script>
</body>
</html>
```

ДОДАТОК И. РЕЗУЛЬТАТИ ТЕСТУВАННЯ

1. Юніт-тестування (Jest)

Юніт-тестування проводилося за допомогою фреймворку Jest для перевірки окремих функцій JavaScript. Основними об'єктами тестування були функції, що відповідали за відкриття підменю та відображення текстових інструкцій.

Функція toggleSubMenu:

— Мета тесту: Перевірити, чи функція коректно відкриває підменю при натисканні на кнопку і закриває інші відкриті підменю.

— Результат: Функція пройшла тест, коректно працюючи з усіма підменю.

— Виявлені проблеми: Початкова версія функції не приховувала відкриті підменю перед відкриттям нового, що призводило до плутанини.

— Виправлення: Додано логіку приховування всіх підменю перед відкриттям нового.

2. Тестування сумісності

Тестування сумісності проводилося для перевірки коректної роботи програми в різних браузерах та на різних пристроях.

Браузери: Google Chrome, Mozilla Firefox, Microsoft Edge.

Пристрої: Комп'ютери, планшети, смартфони.

— Результат: Веб-застосунок працював коректно у всіх браузерах і на пристроях різного типу. Програму тестували на різних розмірах екранів.

— Виявлені проблеми: В деяких браузерах елементи могли перекриватися на маленьких екранах.

— Виправлення: Додано адаптивні стилі для різних розмірів екранів, щоб забезпечити правильне відображення.

3. Тестування доступності (Lighthouse)

Для перевірки доступності використовувався інструмент Lighthouse у Google Chrome.

Ключові параметри тестування:

— Контрастність кольорів.

— Розмір шрифтів.

— Можливість навігації за допомогою клавіатури та сенсорного екрана.

Результат: Програма отримала хороші оцінки по більшості параметрів, але була виявлена проблема з контрастністю деяких кольорів.

— Виявлені проблеми: Деякі елементи інтерфейсу мали низьку контрастність, що ускладнювало читання.

— Виправлення: Змінена кольорова схема для покращення контрасту та читабельності.

4. Інтеграційне тестування

Інтеграційне тестування проводилося для перевірки взаємодії між різними компонентами програми.

Мета тесту: Перевірити взаємодію кнопок, підменю та текстових блоків.

Результат: Всі компоненти працювали належним чином, підменю відкривалися при натисканні кнопок, текстові блоки з'являлися коректно.

— Виявлені проблеми: Деякі кнопки не взаємодіяли з підменю через неправильне налаштування подій.

— Виправлення: Перевірено правильне налаштування обробників подій для кожної кнопки.

5. Стрес-тестування

Стрес-тестування проводилося для оцінки стабільності програми при великій кількості елементів.

Мета тесту: Перевірити, як програма працює з великою кількістю категорій і текстових блоків.

Результат: Програма стабільно працювала при додаванні великої кількості елементів.

— Виявлені проблеми: Кнопки виходили за межі видимої області екрана при великій кількості елементів.

— Виправлення: Адаптовано верстку, щоб кнопки залишалися в межах екрану навіть при великій кількості елементів.

ДОДАТОК Е. ЗНІМКИ ЕКРАНУ

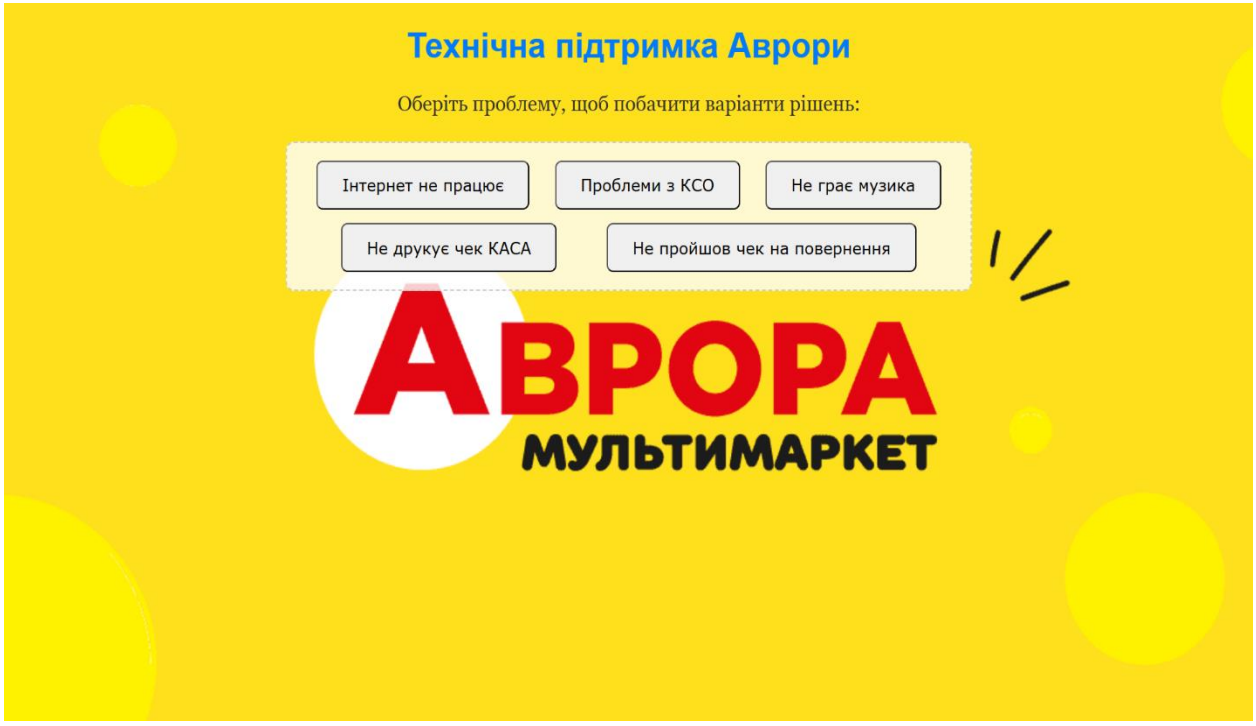

Рисунок 1.1 - Вибір категорії проблеми

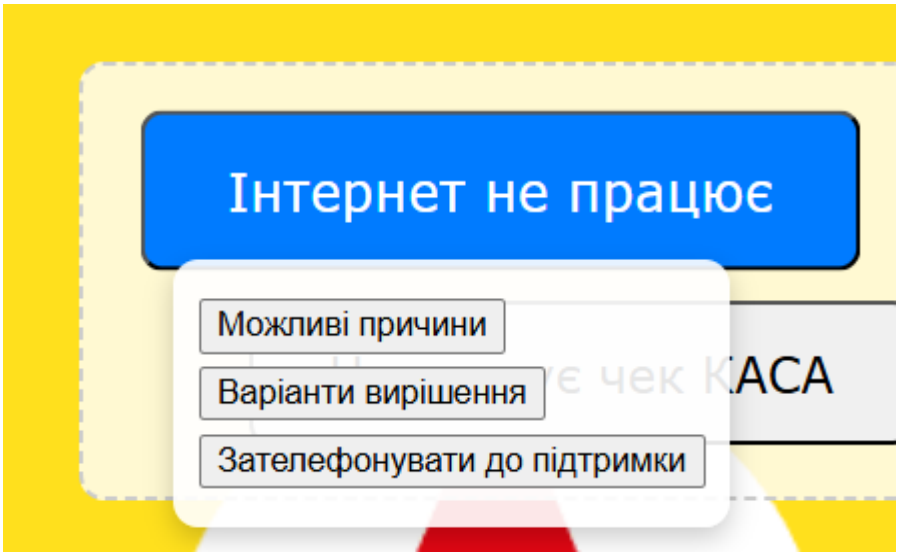

Рисунок 1.2 - Підменю

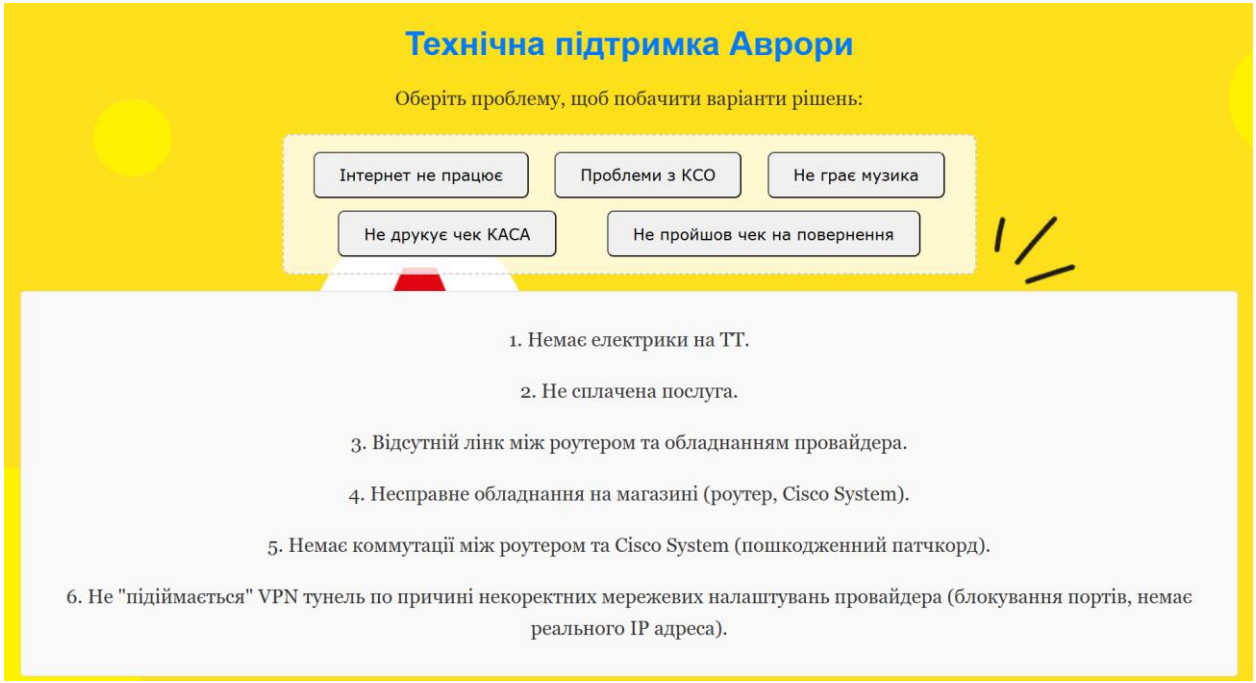

Рисунок 1.3 - Вигляд текстової інструкції “Можливі причини”

Рисунок 1.4 - Вигляд текстової інструкції “Варіанти вирішення”

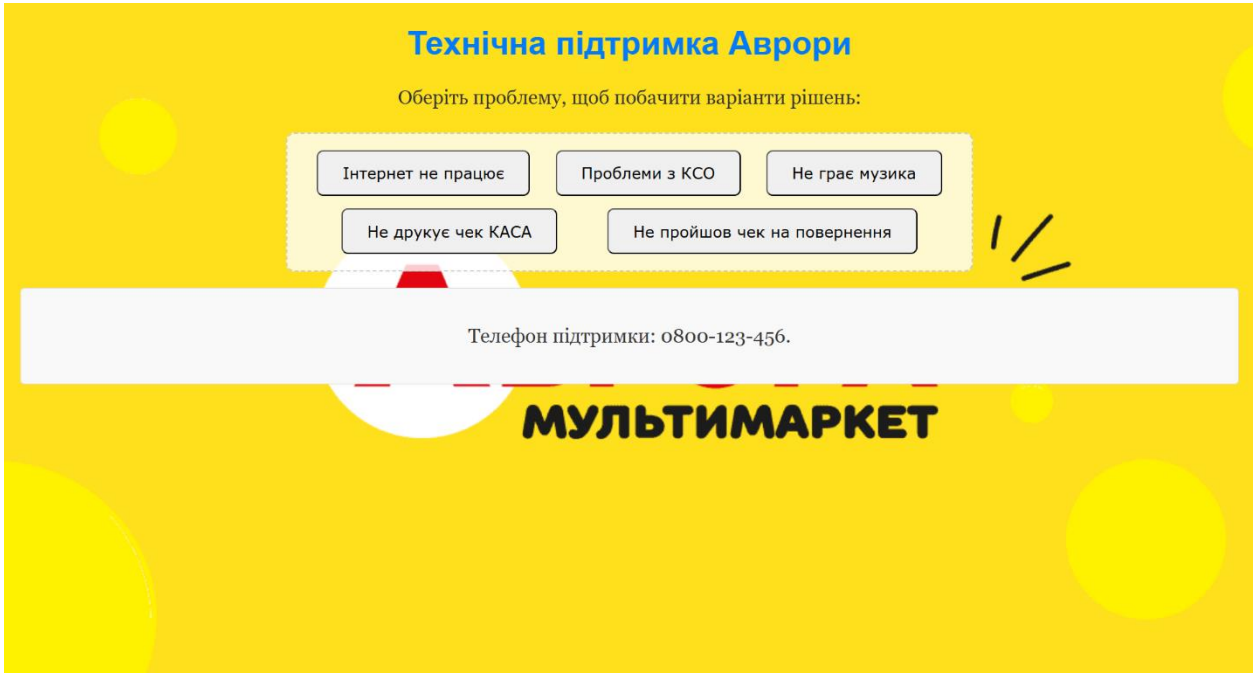

Рисунок 1.5 - Вигляд текстової інструкції “Зателефонувати до підтримки”