МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РЕАЛІЗАЦІЯ СЕРВІСУ ПЛАТІЖНОЇ СИСТЕМИ ОПЛАТИ ПОСЛУГ У ВЕБЗАСТОСУНКУ САНАТОРІЮ»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.О. Малоок
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Горбенко В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Малооку Микиті Олександровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Реалізація сервісу платіжної системи оплати послуг у вебзастосунку санаторію

керівник роботи Горбенко Віталій Іванович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проаналізувати вимоги до програмного застосунку.
4. Створити програмний застосунок з інтегрованою платіжною системою.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	23.01.2024	
3.	Обробка методичних та теоретичних	20.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	16.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	17.06.2024	

Студент _____ М.О. Малоок
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Реалізація сервісу платіжної системи оплати послуг у вебзастосунку санаторію»: 64 с., 32 рис., 1 табл., 20 джерел.

ВЕБЗАСТОСУНОК, ІНТЕГРАЦІЯ ПЛАТІЖНИХ СИСТЕМ, ОБРОБКА ПЛАТЕЖІВ, ОНЛАЙН-ОПЛАТА, ПЛАТІЖНІ СИСТЕМИ, ПЛАТІЖНИЙ ШЛЮЗ, САНАТОРІЙ, ТЕСТУВАННЯ, STRIPE.

Об’єкт дослідження – процес реалізації платіжної системи у вебзастосунку для санаторію.

Мета роботи: розробка вебзастосунку та інтеграція у нього платіжної системи для забезпечення онлайн-оплати послуг санаторію.

Метод дослідження – аналіз літератури, проєктування, конструювання та тестування програмного забезпечення, тестування функціональності, методи збору та аналізу вимог до програмного, кросбраузерне тестування.

У цій роботі розглядаються теоретичні аспекти платіжних систем, типи платіжних систем та їх основні компоненти, технології та стандарти, що використовуються у платіжних системах. Проведено аналіз потреб санаторію та спроєктовано вебзастосунок з інтегрованою платіжною системою. Реалізація включає створення інформаційної сторінки, каталогу послуг, форми оплати та інтеграцію платіжного шлюзу Stripe. Результати роботи показують, що розроблений вебзастосунок відповідає всім визначеним вимогам.

SUMMARY

Bachelor’s qualifying paper “Implementation of the Payment System Service for Payment of Services in the Web Application of the Sanatorium”: 64 pages, 32 figures, 1 table, 20 references.

INTEGRATION OF PAYMENT SYSTEMS, ONLINE PAYMENT, PAYMENT GATEWAY, PAYMENT PROCESSING, PAYMENT SYSTEMS, SANATORIUM, STRIPE, TESTING, WEB APPLICATION.

The object of the study is the process of implementing a payment system in a web application for a sanatorium.

The aim of the study is to develop and integrate a payment system into a web application to ensure convenient and secure online payment for sanatorium services.

The method of research is literature analysis, web application design, software construction and testing, functionality testing, requirements gathering and analysis methods, and cross-browser testing.

This work explores the theoretical aspects of payment systems, including their types and main components, as well as the technologies and standards used in these systems. After analyzing the specific needs of the sanatorium, a web application with an integrated payment system was designed. The implementation involved creating an information page, a service catalog, a payment form, and integrating the Stripe payment gateway. The results demonstrate that the developed web application meets all the defined requirements.

ЗМІСТ

ВСТУП

В умовах сучасного світу, стрімкого розвитку інформаційних технологій та швидкого зростання популярності електронної комерції інтеграція платіжних систем у вебзастосунки набуває все більшої актуальності. Підприємства різних сфер діяльності, прагнуть надати своїм клієнтам зручний та безпечний спосіб оплати послуг через Інтернет. Це дозволяє не лише покращити якість обслуговування, але й підвищити конкурентоспроможність на ринку.

Платіжні системи забезпечують проведення фінансових операцій між покупцями та продавцями, використовуючи мережу Інтернет та інші технології та стандарти. У процесі розробки вебзастосунків виникає потреба у врахуванні аспектів безпеки даних, захисту від шахрайства та забезпечення конфіденційності транзакцій. Вибір відповідного платіжного шлюзу та його інтеграція у вебзастосунок є важливими етапами, що вимагають ґрунтовного аналізу та дослідження.

Тема даної дипломної роботи присвячена реалізації сервісу платіжної системи оплати послуг у вебзастосунку для підприємства сфери обслуговування – санаторію. Метою роботи є інтеграція платіжної системи, що забезпечить онлайн-оплату послуг санаторію. Об’єктом дослідження є процес реалізації платіжної системи у вебзастосунку, а предметом – методи та технології інтеграції платіжних систем у вебзастосунки.

У першому розділі роботи розглянуто теоретичні аспекти платіжних систем, проаналізовано сучасні види платіжних систем, їх основні компоненти та технології, що використовуються для забезпечення безпеки транзакцій. Другий розділ присвячено проєктуванню платіжної системи для санаторію, включаючи аналіз вимог, вибір архітектурного підходу та технологій, що забезпечують інтеграцію платіжної системи у вебзастосунок. У третьому розділі розглянуто процес реалізації та тестування вебзастосунку з

інтегрованою платіжною системою, проведено аналіз результатів та оцінку ефективності роботи.

У ході проведення дослідження та реалізації платіжної системи у вебзастосунку санаторію стає очевидним, що використання сучасних технологій та методів дозволяє значно спростити процес товарообігу між покупцем та продавцем. Також, це сприяє підвищенню рівня обслуговування клієнтів та зміцненню позицій бізнесу сфери обслуговування на ринку.

1 ТЕОРЕТИЧНІ АСПЕКТИ ПЛАТІЖНИХ СИСТЕМ

1.1 Огляд сучасних платіжних систем

Платіжні системи відіграють важливу роль у сучасній економіці. Вони забезпечують безперебійну обробку фінансових транзакцій між споживачами та постачальниками товарів і послуг. Платіжні системи еволюціонували від простих обмінних механізмів до складних електронних систем, що працюють у глобальному масштабі [1].

Перші платіжні системи виникли шляхом ускладнення та розвитку існуючих платіжних операцій, з необхідності обміну товарами та послугами між покупцями та продавцями. В міру розвитку технологій, з’явилися банківські системи, що дозволяли здійснювати безготівкові платежі. З кінця XX століття, з появою мережі Інтернет, почали активно розвиватися електронні платіжні системи, які забезпечують миттєві транзакції через Інтернет [2].

Платіжна система – це набір механізмів, правил, технологій та інструментів які забезпечують передачу коштів від платника до отримувача [3, с. 20]. Основними елементами платіжних систем є платіжні інструменти (картки, електронні гаманці), платіжні інтерфейси (онлайн-банкінг, мобільні додатки) та інфраструктура (платіжні шлюзи, процесингові центри) [4].

З переходом від традиційних готівкових операцій до безготівкових та електронних форм розрахунків платіжні системи стали більш складними та розвинутими. Наприклад, карткові системи, що базуються на використанні пластикових карток з інтегрованими мікросхемами або магнітними смужками, зробили фінансові трансакції більш зручними та швидкими для споживачів та торговельних точок [5].

Значний вплив на розвиток платіжних систем має також технологічний прогрес. Впровадження нових технологій, таких як безконтактні платежі за

допомогою NFC (Near Field Communication), мобільні платіжні додатки та інші інновації, розширює можливості споживачів та покращує ефективність платіжних операцій [6].

Платіжні системи є необхідною складовою сучасного економічного середовища, що впливає на спосіб здійснення фінансових трансакцій як в онлайн, так і в офлайн сегментах ринку. Огляд сучасних платіжних систем дозволяє краще розуміти їхні особливості, переваги та недоліки, а також тренди їхнього розвитку [2].

На сучасному ринку платіжних систем домінують кілька великих гравців, таких як Visa, Mastercard, PayPal та Stripe. Водночас, зростає популярність нових технологій, таких як криптовалюти (Bitcoin, Ethereum) та мобільні платіжні платформи (Apple Pay, Google Wallet). Серед вітчизняних аналогів електронних платіжних систем, можна назвати такі як: LiqPay, EasyPay, NovaPay, «Простір» та інші [2]. Важливою тенденцією є інтеграція платіжних систем у соціальні мережі та месенджери, що робить процес оплати ще зручнішим для користувачів.

Однак разом з розвитком нових технологій виникають і нові виклики, зокрема у забезпеченні безпеки та захисту особистих даних користувачів. Тому, огляд сучасних платіжних систем також включає аналіз заходів та стандартів забезпечення безпеки платіжних трансакцій та захисту конфіденційності даних.

1.2 Види платіжних систем

Платіжні системи можна класифікувати за різними критеріями, такими як тип учасників, методи розрахунків, використані технології тощо. Далі розглянемо основні види платіжних систем, що існують сьогодні, їхні особливості та сфери застосування.

Банківські платіжні системи є традиційними і найбільш поширеними.

Вони включають у себе системи міжбанківських розрахунків, такі як SWIFT, TARGET2 та інші. Ці системи забезпечують перекази між банківськими рахунками, використовуючи різні платіжні інструменти, зокрема, кредитні та дебетові картки, чеки та електронні перекази. Основними перевагами банківських платіжних систем є висока надійність, широке охоплення та регуляторна підтримка [7].

Міжнародні платіжні системи, такі як Visa і Mastercard, дозволяють здійснювати платежі по всьому світу. Вони забезпечують обробку транзакцій між банками різних країн, використовуючи міжнародні стандарти та протоколи. Ці системи зазвичай підтримують різні валюти і забезпечують високу швидкість та безпеку транзакцій. Крім того, вони пропонують додаткові сервіси, такі як програми лояльності та захист покупців [7, 8].

Електронні платіжні системи (e-wallets) набули широкого розповсюдження завдяки розвитку Інтернету. До них належать такі сервіси, як PayPal, Apple Pay, Google Pay, Skrill, та EasyPay. Ці системи дозволяють користувачам зберігати гроші в електронному вигляді і здійснювати платежі через Інтернет. Вони забезпечують швидкість і зручність транзакцій, а також мають нижчі комісії порівняно з традиційними банківськими системами. Однак, вони можуть бути обмежені в залежності від географічного розташування та нормативного середовища [9].

Мобільні платіжні системи стали популярними завдяки широкому розповсюдженню смартфонів. Вони дозволяють здійснювати платежі через мобільні додатки або SMS. Прикладами таких систем є Apple Pay, Google Wallet, та Samsung Pay. Вони використовують технології NFC (Near Field Communication) для безконтактних платежів, що забезпечує високу зручність та безпеку. Мобільні платіжні системи також активно впроваджуються у країнах, де банківська інфраструктура розвинена недостатньо. Бар’єром розвитку таких систем є достатнє розповсюдження мобільних девайсів які підтримують технологію NFC [10].

Блокчейн технології відкрили нові можливості для створення

децентралізованих платіжних систем. Криптовалюти, такі як Bitcoin, Ethereum, та інші, дозволяють здійснювати транзакції без участі посередників, що забезпечує високу швидкість, низькі витрати і повну анонімність. Ці системи базуються на принципах децентралізації, прозорості та безпеки. Водночас, вони стикаються з викликами, такими як висока волатильність та регуляторні обмеження [11].

1.3 Основні компоненти платіжних систем

Платіжні системи складаються з різноманітних компонентів, кожен з яких виконує певну функцію в процесі обробки платежів. Разом, ці компоненти, складають систему обробки оплати (див. рис. 1.1). Розуміння цих компонентів допоможе у ефективному проєктувані та інтеграції платіжних систем у вебзастосунки.

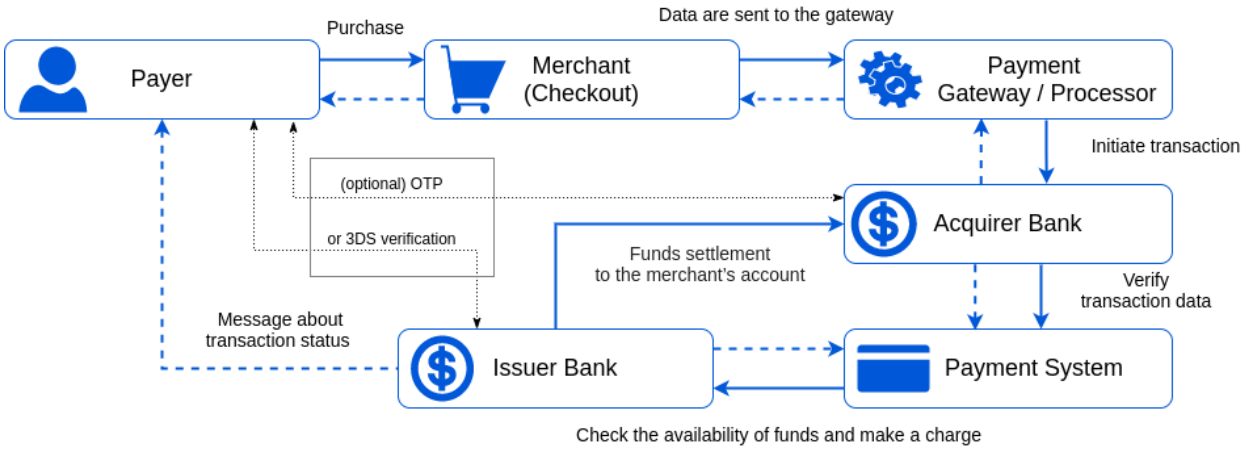

Рисунок 1.1 – Схема процесу обробки платежу

Платіжний шлюз – це інтерфейс, який з'єднує торгові майданчики (онлайн-магазини, вебзастосунки) з фінансовими установами (банками, процесинговими центрами). Шлюз приймає платіжні дані від клієнтів, шифрує їх для безпеки і передає до процесора платежів для подальшої обробки. Платіжні шлюзи часто надають додаткові сервіси, такі як запобігання

шахрайству, автоматизація повторних платежів і підтримка різних платіжних методів (кредитні картки, електронні гаманці, криптовалюти).

Процесор платежів є ключовим компонентом, який відповідає за обробку платіжних транзакцій між торговцем і фінансовою установою клієнта. Він перевіряє платіжні дані, забезпечує аутентифікацію та авторизацію транзакцій, а також ініціює переказ коштів. Процесори платежів співпрацюють з банками-еквайрами (установами, що обслуговують торговців) і емітентами карток (банками, що випускають картки клієнтам), щоб забезпечити успішне завершення транзакцій.

Емітенти – це фінансові установи, які випускають платіжні картки для клієнтів. Вони несуть відповідальність за видачу карток, управління рахунками клієнтів і здійснення платежів за їх рахунок.

Еквайри – це банки, які обслуговують торговців, приймаючи платежі за товари та послуги. Вони забезпечують інтеграцію з процесорами платежів і надають торговцям доступ до платіжних систем. Співпраця між емітентами та еквайрами є критичною для функціонування платіжних систем [3, с. 24-27].

Аутентифікація та авторизація – це процеси, що забезпечують безпеку платіжних транзакцій. Аутентифікація полягає у перевірці особи користувача, що здійснює платіж, шляхом використання паролів, PIN-кодів, біометричних даних або токенів. Авторизація – це перевірка правомірності та достатності коштів для здійснення транзакції [3, с. 161]. Платіжні системи використовують різні методи для забезпечення цих процесів, включаючи одноразові паролі (OTP), двофакторну аутентифікацію (2FA) та протоколи 3D Secure.

Розрахункові та клірингові центри відіграють важливу роль у забезпеченні завершення платіжних транзакцій. Розрахункові центри здійснюють переказ коштів між банками, забезпечуючи належне зарахування коштів на рахунки отримувачів. Клірингові центри займаються обробкою та підтвердженням транзакцій, зводячи їх до спільного розрахунку для мінімізації кількості реальних переказів. Це дозволяє підвищити ефективність і швидкість платіжних операцій [3, с. 26].

1.4 Технології та стандарти платіжних систем

Сучасні платіжні системи спираються чіткі стандарти, що забезпечують безпеку, ефективність та інтеграцію фінансових транзакцій. Розглянемо ключові технології та стандарти, що використовуються в платіжних системах:

- протоколи передачі даних;
- стандарти безпеки;
- технології шифрування та токенізації;
- програмні інтерфейси та SDK для інтеграції платіжних систем.

Далі, детально розглянемо кожен з пунктів переліку технології та стандарти платіжних систем.

Розглянемо протоколи передачі даних. Безпека передачі даних є критично важливою в платіжних системах. HTTPS (HyperText Transfer Protocol Secure) – це протокол, що забезпечує захищений обмін даними між клієнтом та сервером через Інтернет. Він використовує шифрування на основі SSL/TLS (Secure Sockets Layer / Transport Layer Security) для захисту інформації від несанкціонованого доступу та маніпуляцій. TLS є більш сучасною та безпечною версією SSL, що забезпечує цілісність даних і конфіденційність під час їх передачі [12].

PCI DSS (Payment Card Industry Data Security Standard) – це набір стандартів безпеки, розроблених для захисту даних платіжних карток. Він включає вимоги щодо зберігання, обробки та передачі даних карток, такі як шифрування, управління доступом та регулярне проведення аудитів безпеки. Впровадження PCI DSS є обов’язковим для всіх організацій, що обробляють платіжні картки [13].

Далі, зосередимо увагу на стандартах безпеки. EMV (Europay, Mastercard, and Visa) – це стандарт для чіпових карток, що забезпечує підвищений рівень безпеки порівняно з магнітними стрічками. EMV-картки містять мікропроцесор, який генерує унікальний код для кожної транзакції, що робить їх більш стійкими до шахрайства [14].

Шифрування є основним методом захисту даних у платіжних системах. Воно перетворює інформацію у вигляді зрозумілого тексту в зашифрований формат, який може бути розшифрований лише за допомогою спеціального ключа. Компанії широко використовують протоколи шифрування, такі як Secure Sockets Layer (SSL) і Transport Layer Security (TLS) [12].

Токенізація – це процес заміни чутливих даних (наприклад, номера картки) на унікальний ідентифікатор або токен. Токени не мають жодного значення поза контекстом платіжної системи, що знижує ризик витоку даних. Токенізація часто використовується разом із шифруванням для забезпечення багаторівневого захисту [12].

Інтеграція платіжних систем у вебзастосунки здійснюється за допомогою програмних інтерфейсів (API) та SDK (Software Development Kit). API дозволяють взаємодіяти з платіжними системами через стандартні протоколи передачі даних.

REST (Representational State Transfer) і SOAP (Simple Object Access Protocol) – це два популярні підходи до створення вебсервісів. REST є легшим і простішим у реалізації, що робить його популярним для вебзастосунків. SOAP є більш складним, але забезпечує додаткові функції, такі як надійність повідомлень і безпека на рівні транспортного протоколу [15].

1.5 Правові та регуляторні аспекти платіжних систем

Правові та регуляторні аспекти є ключовими елементами функціонування платіжних систем, оскільки вони визначають рамки, у яких здійснюються фінансові операції. Ці аспекти включають різноманітні закони, регуляції, стандарти та вимоги, які платіжні системи повинні виконувати для забезпечення легітимності, безпеки та надійності.

Розглянемо основні правові та регуляторні аспекти платіжних систем:

– законодавче регулювання платіжних систем: міжнародні стандарти,

національне законодавство;

- закони про захист персональних даних;
- регулювання платіжних послуг;
- ліцензування та реєстрація;
- ліцензія платіжної установи;
- реєстрація та звітність;
- протидія відмиванню грошей (AML) та фінансуванню тероризму (CFT);
- KYC (Know Your Customer);
- моніторинг транзакцій;
- забезпечення прав споживачів;
- прозорість;
- захист від шахрайства.

Тепер детально пояснимо кожний пункт переліку.

Міжнародні стандарти регулювання платіжних систем складають PCI DSS та міжнародний стандарт ISO 20022. PCI DSS (Payment Card Industry Data Security Standard) – це стандарт безпеки, розроблений міжнародними платіжними системами (Visa, MasterCard, American Express, Discover, JCB), який встановлює вимоги щодо захисту даних карткових платежів. Всі організації, що обробляють, зберігають або передають дані платіжних карт, повинні відповідати вимогам PCI DSS.

ISO 20022 – це міжнародний стандарт для електронного обміну фінансовими даними між фінансовими установами. Він забезпечує стандартизацію форматів та змісту платіжних повідомлень, сприяючи взаємодії між різними платіжними системами на глобальному рівні [17].

В різних країнах діють закони, які регулюють обробку та захист персональних даних. Наприклад, у Європейському Союзі діє Загальний регламент захисту даних (GDPR), який встановлює жорсткі вимоги до захисту особистої інформації [18].

Країни мають власні регуляторні органи, які контролюють діяльність

платіжних систем. Наприклад, у США таку функцію виконує Федеральна торгова комісія (FTC) та Офіс контролера валют (OCC), а в ЄС – Європейський центральний банк (ЄЦБ) та національні центральні банки.

Платіжні системи повинні отримати ліцензії або реєстрацію від відповідних регуляторних органів для здійснення своєї діяльності. Цей процес включає подання документів, що підтверджують відповідність вимогам, таких як фінансова стійкість, внутрішні контролі, безпека та управління ризиками.

Платіжні системи повинні отримати ліцензію від регуляторних органів, що дозволяє їм здійснювати платіжні операції. Наприклад, у ЄС платіжні установи повинні відповідати Директиві про платіжні послуги (PSD2).

Платіжні системи повинні регулярно подавати звіти до регуляторних органів, включаючи фінансові звіти, звіти про внутрішній контроль та дотримання вимог безпеки.

Платіжні системи повинні дотримуватись вимог законодавства щодо протидії відмиванню грошей та фінансуванню тероризму. Це включає впровадження процедур ідентифікації клієнтів (KYC – Know Your Customer), моніторинг транзакцій та повідомлення про підозрілі операції.

Впровадження процедур для ідентифікації та верифікації клієнтів, що включає збір та перевірку документів, що підтверджують особу.

Платіжні системи повинні впроваджувати системи моніторингу для виявлення підозрілих транзакцій та своєчасного повідомлення регуляторних органів про такі випадки.

Платіжні системи повинні забезпечувати захист прав споживачів, включаючи прозорість умов обслуговування, захист від шахрайства та можливість оскарження транзакцій.

Платіжні системи повинні надавати клієнтам чітку інформацію про умови використання своїх послуг, включаючи тарифи, умови повернення коштів та обробку скарг.

Впровадження технологій та процедур для захисту клієнтів від шахрайських операцій, таких як двофакторна аутентифікація та системи

виявлення шахрайства.

Правові та регуляторні аспекти платіжних систем є фундаментальними для забезпечення їхньої надійності, безпеки та легітимності. Виконання вимог законодавства, дотримання міжнародних стандартів, отримання необхідних ліцензій, впровадження процедур протидії відмиванню грошей та забезпечення прав споживачів є критично важливими для успішної роботи платіжних систем. Це дозволяє не тільки забезпечити захист клієнтів і знизити ризики, але й сприяє довірі до платіжних систем з боку користувачів та регуляторів [16].

1.6 Висновки до розділу 1

Перший розділ дипломної роботи присвячений теоретичним аспектам платіжних систем. Були висвітлені такі теми як: сучасні платіжні системи, види платіжних систем, основні компоненти платіжних систем, технології та стандарти платіжних систем, правові та регуляторні аспекти платіжних систем.

В ході розгляду тем були висвітлені ключові елементи та характеристики платіжних систем, що дозволяють зрозуміти їхню функціональність та структуру.

2 ПРОЄКТУВАННЯ ПЛАТІЖНОЇ СИСТЕМИ ДЛЯ САНАТОРІЮ

2.1 Вебсервіси платіжних систем

Вебсервіси платіжних систем забезпечують ключову інфраструктуру для обробки онлайн-платежів у вебзастосунках. Вони дозволяють інтегрувати платіжні функціональності, такі як прийом платежів, обробка транзакцій та управління клієнтськими даними, безпосередньо у вебінтерфейси. Розглянемо основні аспекти вебсервісів платіжних систем, їхню архітектуру, функціональні можливості та принципи інтеграції.

Архітектура вебсервісів платіжних систем зазвичай базується на клієнт-серверній моделі, де вебзастосунок клієнта взаємодіє з платіжним сервером через програмні інтерфейси (API). Ці API дозволяють виконувати різні операції, такі як ініціювання платежів, перевірка статусу транзакцій, управління рахунками тощо.

Типова архітектура включає наступні компоненти:

- клієнтський інтерфейс;
- серверний інтерфейс;
- платіжний шлюз;
- процесинговий центр.

Клієнтський інтерфейс – це частина вебзастосунку, з якою взаємодіє користувач. Вона забезпечує введення платіжних даних (номер картки, CVV, термін дії тощо) та передачу цих даних до серверної частини для обробки.

Серверний інтерфейс – серверна частина вебзастосунку, що відповідає за обробку запитів від клієнта, взаємодію з платіжним API та керування бізнес-логікою транзакцій.

Платіжний шлюз – це проміжний сервер, який забезпечує захищену передачу платіжних даних від вебзастосунку до фінансових установ для авторизації та виконання платежів.

Процесинговий центр – це сервер, який обробляє транзакції, взаємодіє з банками-еквайрами та забезпечує фінансові розрахунки між учасниками платіжного процесу.

Вебсервіси платіжних систем надають широкий спектр функцій, які можуть бути інтегровані у вебзастосунки:

– обробка платежів;
– повторювані платежі;
– управління рахунками;
– запобігання шахрайству;
– підтримка багатьох валют.

Обробка платежів – прийом та обробка платежів за допомогою різних методів, включаючи кредитні та дебетові картки, електронні гаманці, банківські перекази тощо.

Повторювані платежі – це можливість налаштування автоматичних періодичних платежів, що корисно для підписок та регулярних послуг.

Управління рахунками – це процес створення, оновлення та видалення облікових записів користувачів, зберігання платіжних даних та історії транзакцій.

Запобігання шахрайству – це означає впровадження механізмів виявлення та запобігання шахрайським операціям, таких як аналіз транзакцій, виявлення підозрілих активностей та багаторівнева аутентифікація.

Підтримка багатьох валют – означає обробку платежів у різних валютах, що особливо важливо для міжнародних вебзастосунків.

Інтеграція вебсервісів платіжних систем у вебзастосунки вимагає врахування кількох ключових принципів:

– безпека;
– надійність;
– масштабованість;
– зручність використання.

Безпека означає забезпечення захисту платіжних даних через

шифрування, дотримання стандартів безпеки (наприклад, PCI DSS) та використання безпечних протоколів передачі даних (HTTPS, TLS).

Надійність означає гарантування високої доступності та стабільності платіжних сервісів, щоб користувачі могли безперешкодно здійснювати транзакції.

Масштабованість означає можливість масштабування платіжної інфраструктури відповідно до зростання навантаження, щоб обробляти збільшену кількість транзакцій без втрати продуктивності.

Зручність використання – це забезпечення інтуїтивно зрозумілого інтерфейсу для користувачів, щоб вони могли легко здійснювати платежі з мінімальною кількістю кроків

2.2 Реалізація платіжних систем у вебзастосунках

2.2.1 Архітектурні підходи до інтеграції платіжних систем у вебзастосунки

Інтеграція платіжних систем у вебзастосунки вимагає планування архітектури для забезпечення ефективної, безпечної та масштабованої роботи. Основні архітектурні підходи включають модульну архітектуру, використання RESTful API, мікросервісну архітектуру та хмарні рішення.

Модульна архітектура передбачає розподіл функціональності вебзастосунку на незалежні модулі. Платіжний модуль відповідає за обробку транзакцій, взаємодіючи з основним застосунком через чітко визначені інтерфейси. Цей підхід спрощує інтеграцію нових платіжних систем та дозволяє легко оновлювати чи замінювати окремі компоненти без впливу на інші частини застосунку.

Використання RESTful API є популярним підходом для інтеграції платіжних систем. REST (Representational State Transfer) дозволяє взаємодіяти

з платіжними сервісами через HTTP-запити. Цей підхід забезпечує простоту та гнучкість, дозволяючи легко інтегрувати платіжні функції у вебзастосунки. RESTful API зазвичай використовує формати даних JSON або XML, що сприяє швидкій обробці та обміну інформацією між системами.

Мікросервісна архітектура розділяє застосунок на невеликі, незалежні сервіси, кожен з яких виконує певну функцію. Платіжна система може бути реалізована як окремий мікросервіс, що спрощує її управління та масштабування. Цей підхід забезпечує високу гнучкість та можливість незалежного розгортання, що особливо корисно для великих та складних систем.

Інтеграція хмарних платіжних рішень дозволяє використовувати готові сервіси від провайдерів, таких як AWS (Amazon Web Services), Google Cloud та Azure. Хмарні сервіси надають інфраструктуру для обробки платежів, забезпечують високу доступність та масштабованість. Використання хмарних рішень знижує витрати на підтримку власної інфраструктури та забезпечує швидкий доступ до нових технологій та функціональностей.

Вибір архітектурного підходу залежить від конкретних вимог проєкту, включаючи масштабованість, надійність, безпеку та доступність ресурсів для розробки та підтримки системи. Важливо провести детальний аналіз потреб та обгрунтувати вибір архітектури для забезпечення успішного впровадження платіжних систем у вебзастосунки санаторію.

2.2.2 Технічні аспекти інтеграції платіжних шлюзів у вебзастосунки

Інтеграція платіжних шлюзів у вебзастосунки вимагає врахування низки технічних аспектів, щоб забезпечити безпеку, ефективність та зручність обробки транзакцій. Основні технічні аспекти включають налаштування API, обробку даних користувачів, забезпечення безпеки та управління помилками.

Першим кроком інтеграції платіжного шлюзу є налаштування API, який

забезпечує зв’язок між вебзастосунком та платіжним провайдером. Це включає отримання API-ключів, налаштування вебхуків для отримання сповіщень про статус транзакцій та інтеграцію SDK (Software Development Kit) платіжного шлюзу, якщо він доступний [19]. Важливо забезпечити коректну автентифікацію та авторизацію запитів до API, використовуючи сучасні методи, такі як OAuth.

Обробка платіжних даних користувачів повинна здійснюватися відповідно до стандартів безпеки. Платіжні дані, такі як номери кредитних карток та CVV-коди, не повинні зберігатися на серверах вебзастосунку у незашифрованому вигляді. Замість цього слід використовувати методи токенізації та шифрування, щоб захистити конфіденційну інформацію. Також важливо забезпечити, щоб дані передавалися через захищені канали зв’язку (HTTPS).

Безпека є ключовим аспектом інтеграції платіжних шлюзів. Надійним методом захисту є впровадження багаторівневих заході безпеки, включаючи двофакторну автентифікацію, захист від атак типу CSRF (Cross-Site Request Forgery) та регулярні перевірки на наявність вразливостей. Дотримання стандартів PCI DSS (Payment Card Industry Data Security Standard) є обов’язковим для всіх організацій, що обробляють платіжні картки.

Ефективне управління помилками є важливим для забезпечення безперервної роботи платіжної системи. Це включає обробку помилок на стороні клієнта та сервера, забезпечення зрозумілих повідомлень для користувачів та ведення журналу транзакцій для подальшого аналізу. Вебзастосунок повинен мати механізми для повторної спроби транзакцій у випадку тимчасових збоїв та відстеження стану кожної транзакції для забезпечення її коректного завершення. Узагальнюючи, технічні аспекти інтеграції платіжних шлюзів у вебзастосунки вимагають комплексного та систематичного підходу, що охоплює вибір платіжного шлюзу, реалізацію автентифікації та авторизації, інтеграцію з базою даних, тестування, моніторинг та підтримку. Важливо враховувати найкращі практики розробки

програмного забезпечення та забезпечувати високу якість та надійність системи оплати для задоволення потреб клієнтів та забезпечення успішного функціонування санаторію.

2.2.3 Безпека та захист даних у контексті платіжних систем у вебзастосунках

Безпека та захист даних є критично важливими аспектами інтеграції платіжних систем у вебзастосунки. З огляду на ризики, пов’язані з обробкою фінансових транзакцій та персональних даних. Такі ризики включають:

- ризики пов’язані з управлінськими помилками;
- шахрайство;
- перехопленням даних;
- неефективність надання послуг.

Щоб запобігти це, необхідно впроваджувати багаторівневі заходи безпеки. Шифрування є основним методом захисту даних під час їх передачі та зберігання. Всі платіжні дані повинні передаватися через захищені канали зв’язку за допомогою протоколу HTTPS, який використовує SSL/TLS для шифрування інформації. Крім того, важливо шифрувати дані, які зберігаються на сервері, використовуючи сучасні алгоритми, такі як AES (Advanced Encryption Standard).

Токенізація є додатковим засобом захисту, що замінює чутливі платіжні дані на унікальні токени. Ці токени не мають значення поза контекстом конкретної платіжної системи, що зменшує ризик компрометації даних. Навіть якщо токен буде перехоплений, він не може бути використаний для несанкціонованих транзакцій.

Двофакторна автентифікація додає додатковий рівень захисту, вимагаючи від користувачів підтвердження своєї особи за допомогою двох незалежних методів: наприклад, паролю та одноразового коду, надісланого на

мобільний телефон. Це значно знижує ризик несанкціонованого доступу навіть у випадку компрометації паролю.

Захист від атак типу CSRF (Cross-Site Request Forgery) та XSS (Cross-Site Scripting) є важливим аспектом безпеки вебзастосунків. Для захисту від CSRF слід використовувати токени перевірки запитів, що додаються до форм та перевіряються на сервері. Для захисту від XSS необхідно впроваджувати фільтрацію та екранування вводу користувачів, а також використовувати Content Security Policy (CSP) для обмеження виконання небажаного коду.

Дотримання стандартів PCI DSS (Payment Card Industry Data Security Standard) є обов'язковим для всіх вебзастосунків, що обробляють платіжні дані. Це включає регулярні аудити безпеки, моніторинг транзакцій, управління доступом до даних та інші заходи, спрямовані на захист платіжної інформації.

2.2.4 Управління транзакціями та обробка помилок платіжних операцій у вебзастосунках

Управління транзакціями та обробка помилок є критичними аспектами функціонування платіжних систем у вебзастосунках. Це включає забезпечення коректного виконання платіжних операцій, своєчасне виявлення та вирішення помилок, а також гарантування безперервної роботи системи.

Коректне управління транзакціями забезпечує цілісність і консистентність даних. Це досягається через наступні механізми:

- атомарність транзакцій;
- журналювання;
- ідентифікація транзакцій.

Тепер детально пояснимо значення кожного механізму.

Кожна транзакція повинна бути атомарною, тобто або завершитися повністю, або не завершитися взагалі. Це запобігає частковому виконанню операцій, що може призвести до некоректних даних.

Ведення журналу транзакцій допомагає відстежувати кожну операцію та її статус. Це важливо для аудиту, аналізу та вирішення проблем у разі збоїв.

Кожна транзакція повинна мати унікальний ідентифікатор, що дозволяє легко відстежувати її стан та історію.

Ефективна обробка помилок забезпечує стабільність та надійність платіжної системи. Основні аспекти включають:

- виявлення помилок;
- категоризація помилок;
- обробка помилок;
- логування.

Розглянемо кожний аспект обробки помилок.

Система повинна мати механізми для автоматичного виявлення помилок під час виконання транзакцій. Це можуть бути як апаратні, так і програмні засоби моніторингу. Помилки слід категоризувати за типом (наприклад, мережеві, серверні, клієнтські) та серйозністю. Це допомагає визначити пріоритети для їхнього вирішення.

В залежності від типу помилки, система повинна виконувати відповідні дії. Це може бути повторна спроба виконання транзакції, повернення коштів або інформування користувача про помилку та подальші кроки.

Всі помилки повинні логуватися з детальною інформацією про контекст, що дозволяє аналізувати причини збоїв та запобігати їх повторенню.

2.2.5 Порівняння та аналіз переваг та недоліків різних підходів

Існує кілька підходів до інтеграції платіжних систем у вебзастосунки, кожен з яких має свої переваги та недоліки. Основні підходи включають використання готових платіжних платформ (наприклад, PayPal, Stripe), розробку власних рішень та інтеграцію з банківськими шлюзами (див. табл. 1.1).

Таблиця 1.1 – Таблиця порівняльної характеристики підходів до інтеграції платіжних систем

Тип підходу	**Переваги**	**Недоліки**
Готові платіжні платформи	Простота інтеграції: готові платіжні платформи надають зручні API та SDK. Безпека: високий рівень безпеки, відповідність стандартам PCI DSS та регулярні оновлення. Надійність: готові рішення забезпечують високу доступність та масштабованість.	Витрати: високі комісії за транзакції можуть бути значним недоліком для бізнесу. Обмежена кастомізація: обмежені можливості для налаштування під специфічні потреби бізнесу.
Власні рішення	Гнучкість: можливість повної кастомізації під специфічні вимоги бізнесу. Контроль: повний контроль над усіма аспектами платіжної системи, включаючи безпеку та обробку даних.	Складність: високі вимоги до ресурсів для розробки та підтримки. Безпека: необхідність забезпечення відповідності стандартам безпеки та регулярних оновлень.
Банківські шлюзи	Довіра: високий рівень довіри з боку користувачів завдяки співпраці з відомими банками. Інтеграція з банківськими послугами: можливість використання додаткових банківських послуг, таких як кредити та розстрочки.	Складність інтеграції: вимагає більше часу та зусиль для налаштування та інтеграції. Обмеження: можливі обмеження у функціональності та гнучкості у порівнянні з іншими рішеннями.

2.3 Проєкт вебзастосунку санаторію

Проєктування вебзастосунку санаторію з інтегрованою платіжною системою вимагає комплексного підходу, який включає аналіз вимог, розробку архітектури, вибір технологій, реалізацію та тестування. Метою є створення надійного, безпечного та зручного у використанні інструменту для управлінням вибору послуг та оплатами.

Першим етапом є детальний аналіз потреб санаторію та його клієнтів. Визначаються основні функціональні вимоги, такі як вибір послуг, оплата послуг, генерація звітів та інтеграція з існуючими системами санаторію. Нефункціональні вимоги включають безпеку, продуктивність, масштабованість та зручність використання. Важливо також врахувати регуляторні вимоги, такі як відповідність стандартам захисту даних.

Архітектура вебзастосунку повинна забезпечувати гнучкість, масштабованість та безпеку. Вибір архітектурного стилю, такого як RESTful API, є доцільним для забезпечення простоти та легкого настроювання окремих компонентів. Основні компоненти включають фронтенд, бекенд, та платіжний модуль.

Фронтенд відповідає за інтерфейс користувача та реалізується з використанням HTML, CSS, JavaScript. Бекенд обробляє бізнес-логіку використовуючи фреймворк Node.js.

Для інтеграції платіжної системи буде використано готове рішення – Stripe. Ця платіжна платформа надає зручний інтерфейс для обробки платежів та забезпечує високий рівень безпеки.

Реалізація вебзастосунку починається з розробки прототипу, який включає основні функціональні можливості. Після затвердження прототипу розробляється повноцінний застосунок з урахуванням всіх вимог. Важливим етапом є інтеграція платіжної системи, що включає налаштування API, забезпечення безпеки та обробку транзакцій.

Тестування вебзастосунку здійснюється на всіх етапах розробки.

Функціональне тестування перевіряє коректність виконання всіх бізнес-процесів, включаючи бронювання та оплату. Нефункціональне тестування оцінює продуктивність, безпеку та зручність використання застосунку.

2.4 Висновки до розділу 2

У другому розділі дипломної роботи розглядається ключові елементи та характеристики платіжних систем, архітектурні підходи до інтеграції платіжних систем, технічні аспекти інтеграції платіжних шлюзів, що включають вибір відповідного платіжного шлюзу, безпека при інтеграції платіжних систем та управління транзакціями. Метою розгляду цих тем є проєктування проєкту для санаторію.

При проєктуванні вебзастосунку були розглянуті такі аспекти як: аналіз вимог, вибір архітектурного підходу та технологій, що забезпечують інтеграцію платіжної системи у вебзастосунок.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПЛАТІЖНОЇ СИСТЕМИ ДЛЯ САНАТОРІЮ

Перед розробкою платіжної системи для вебзастосунку санаторію треба провести детальний аналіз потреб і вимог, які визначать функціональні можливості та технічні характеристики майбутньої системи. Такий аналіз допоможе створити платіжну систему, що задовольняє специфічні потреби санаторію і його клієнтів.

3.1 Аналіз потреб та вимог санаторію щодо платіжної системи

Першим кроком аналізу буде вивчення специфіки санаторного бізнесу та особливостей його клієнтів. Санаторій, як правило, надає різноманітні послуги, включаючи лікувальні процедури, харчування, розважальні та спортивні заходи. Клієнти можуть бути різного віку та соціального статусу, з різними потребами та очікуваннями.

На основі першого кроку аналізу визначаються основні потреби та вимоги клієнтів щодо платіжної системи. Наприклад, у нашому випадку вимоги будуть наступними:

а) зручність для користувачів:
   1) простота використання;
   2) підтримка різних методів оплати;
   3) зручність та швидкість оплати;

б) безпека транзакцій:
   1) захист даних;
   2) запобігання шахрайству;
   3) конфеденційність даних;

в) інтеграція з існуючими системами:
   1) підтримка API.

Зручність для користувачів є важливою вимогою для програмного застосунку, роз’яснімо деталі.

Клієнтами є широка група людей різного віку та різними потребами. З цього випливає, що клієнти санаторію повинні мати можливість легко здійснювати платежі за послуги через вебзастосунок. Інтерфейс має бути інтуїтивно зрозумілим, а процес платежу – мінімально складним. Санаторій повинен забезпечити підтримку різних методів оплати, таких як кредитні та дебетові картки, електронні гаманці, банківські перекази та мобільні платежі. Це дозволить задовольнити потреби різних груп клієнтів. Клієнти не повинні довго очікувати здійснення оплати та має бути забезпечена зручність оплати за послуги без зайвих перешкод та затримок.

Безпека проведення платежів є основною вимогою, від якої залежить інший функціонал вебзастосунку.

Платіжна система повинна забезпечувати високий рівень захисту даних користувачів, включаючи шифрування платіжної інформації та дотримання стандартів безпеки, таких як PCI DSS. Важливо впровадити механізми для виявлення та запобігання шахрайським операціям, такі як аналіз поведінки користувачів та багаторівнева аутентифікація. З урахуванням конфіденційності медичної та фінансової інформації, клієнти мають очікувати високого рівня захисту даних під час проведення транзакцій.

Наявність розширених API для взаємодії з іншими системами та сервісами, що використовуються санаторієм, сприятиме легкій інтеграції та розширенню функціональності.

Наступним кроком, сформуємо вимоги до платіжної системи:

а) функціональні вимоги:

1) обробка платежів у реальному часі;
2) підтримка кількох валют;
3) автоматизація повторюваних платежів;

б) нефункціональні вимоги:

1) надійність;

2) масштабованість;

3) відмовостійкість;

в) правові та регуляторні вимоги:

1) дотримання законодавства;

2) документування.

Роз’яснимо функціональні вимоги до вебзастосунку.

Платіжна система повинна забезпечувати швидку обробку транзакцій без затримок. Зокрема, санаторій може мати клієнтів з різних країн, тому важливо забезпечити можливість здійснення платежів у різних валютах. Також має місце можливість налаштування автоматичних платежів для постійних клієнтів, які користуються послугами санаторію на регулярній основі.

Роз’яснимо нефункціональні вимоги до вебзастосунку.

Платіжна система повинна працювати безперебійно, забезпечуючи високу доступність та стійкість до збоїв. Система повинна легко масштабуватися відповідно до зростання кількості користувачів та обсягу транзакцій. Необхідне забезпечення безперервної роботи платіжної системи навіть у випадку збоїв апаратного або програмного забезпечення.

Роз’яснимо правові та регуляторні вимоги до вебзастосунку.

Платіжна система повинна відповідати вимогам місцевого та міжнародного законодавства щодо захисту даних та фінансових транзакцій. Всі транзакції повинні бути ретельно документовані, забезпечуючи можливість проведення аудитів та розслідувань у разі необхідності.

3.2 Проєктування функціональності та інтерфейсів вебзастосунку

Після визначення вимог до програмного застосунку, наступає етап проєктування системи для санаторію. Враховуючи різноманітні потреби клієнтів, а також сучасні тренди у дизайні та розробці вебзастосунків, необхідно ретельно спроєктувати функціональність та інтерфейси для

забезпечення максимальної зручності, ефективності та задоволення користувачів.

Одним із ключових аспектів проєктування є створення дизайну, який буде адаптивним та інтуїтивно зрозумілим для користувачів будь-якого рівня технічної підготовки. Інтерфейс вебзастосунку має бути легко зрозумілим та зручним у використанні навіть для тих користувачів, які не мають великого досвіду використання комп’ютерів чи мобільних пристроїв. Для цього важливо використовувати простий інтерфейс, що дозволить швидко зорієнтуватися та виконати потрібні дії.

Важливою функціональністю вебзастосунку є його інтеграція з платіжною системою. Користувачі мають мати можливість швидко та зручно здійснювати оплату за послуги без необхідності переходу на інші платіжні платформи. Інтеграція повинна бути безпечною та надійною, забезпечуючи захист особистих та фінансових даних користувачів.

Оскільки очікується, що санаторій може приймати клієнтів з різних країн, важливо забезпечити підтримку різних та валют у вебзастосунку. Це дозволить забезпечити зручність та доступність системи для широкого кола користувачів, а також сприятиме розширенню аудиторії санаторію.

Узагальнюючи, аналіз потреб користувачів вебзастосунку, можна підвести висновок проєктування. Додаток має виконувати наступне: забезпечення зручного, безпечного та ефективного використання вебзастосунку, що допоможе покращити користувацький досвід та ефективність роботи санаторію. а також врахувати сучасні технології та тренди у розробці програмного забезпечення.

3.3 Вибір технологій та інструментів для реалізації вебзастосунку

Вибір технологій та інструментів для реалізації вебзастосунку є наступним ключовим етапом, що впливає на продуктивність, масштабованість, безпеку та зручність використання системи. Далі

розглянемо обрані технології для розробки вебзастосунку.

Розробку вебзастосунку вирішено розділити на фронтенд, бекенд та інтеграцію платіжної системи. Основними технології для фронтенду, були обрані:

– HTML5;
– CSS3;
– JavaScript.

Обґрунтуємо вибір технології для розробки фронтенду.

HTML5 та CSS3 використовуються для створення структури та стилізації вебсторінок. HTML5 дозволяє створювати семантичні елементи, що покращує доступність та SEO-оптимізацію сайту. CSS3 забезпечує адаптивний дизайн, який дозволяє вебзастосунку коректно відображатися на різних пристроях.

JavaScript є основною мова програмування для реалізації інтерактивності на вебсторінках. Використовується для обробки подій, взаємодії з сервером та динамічного оновлення контенту без перезавантаження сторінки.

Основні технології для бекенду стали:

– Node.js;
– Express.js.

Обґрунтуємо вибір технології для розробки бекенду.

Платформу Node.js було обрано для виконання JavaScript коду на сервері. Node.js забезпечує високу продуктивність завдяки асинхронній обробці запитів та подій. Використання однієї мови програмування для фронтенду та бекенду спрощує розробку та підтримку коду.

Express.js є легким вебфреймворком для Node.js, що надає інструменти для швидкої та зручної розробки вебдодатків та API. Express.js забезпечує просту маршрутизацію запитів, обробку даних форм та інтеграцію з різними шаблонами баз даних.

Для інтеграції платіжної системи було обрано платіжний шлюз Stripe.

Stripe надає безліч можливостей для обробки платежів, забезпечуючи високу безпеку та зручність для користувачів, що і вплинуло на його вибір [20].

Платіжна система Stripe має багато переваг, серед ключових особливостей, можна зазначити наступі.

Підтримка різних платіжних методів: Stripe підтримує широкий спектр платіжних методів, включаючи кредитні та дебетові картки, електронні гаманці (наприклад, Apple Pay та Google Pay), банківські перекази та інші локальні способи оплати.

Простота інтеграції: Stripe має добре задокументоване API, що дозволяє швидко інтегрувати платіжну систему у вебзастосунок. Завдяки детальним інструкціям та прикладам коду розробники можуть легко налаштувати процес обробки платежів.

Безпека: Stripe забезпечує високий рівень безпеки, відповідаючи стандартам PCI DSS. Всі платіжні дані шифруються та захищаються від несанкціонованого доступу. Крім того, Stripe надає інструменти для запобігання шахрайству та моніторингу підозрілих транзакцій.

Гнучкість: Stripe надає різні рішення для обробки платежів, включаючи Stripe Checkout та Stripe Elements. Stripe Checkout дозволяє швидко створювати готові платіжні сторінки, тоді як Stripe Elements надає можливість повної кастомізації платіжних форм, щоб вони відповідали дизайну вебзастосунку.

3.4 Особливості інтеграції платіжної системи Stripe

Вибір і власне інтеграція платіжної системи є критичним аспектом розробки вебзастосунків, що включають обробку фінансові транзакції. Stripe є однією з найпопулярніших і найнадійніших платіжних платформ, що пропонує широкий спектр можливостей для розробників. Далі у цьому розділі розглянемо ключові особливості інтеграції Stripe, у вебзастосунок.

Інтеграція платіжної системи Stripe відбувається за допомогою низки

методів, які забезпечують обробку платежів. Далі наведені методи інтеграції Stripe:

- stripe.js;
- stripe checkout;
- stripe checkout;
- мобільні sdk.

Далі детально про кожний з цих методів.

Stripe.js – це метод що дозволяє вбудувати форму оплати безпосередньо на сторінці вебзастосунку. Можна використовувати стандартну форму оплати Stripe або створити власну, використовуючи елементи Stripe.js. Це дає можливість здійснювати платежі без перенаправлення користувачів на інші сторінки.

Stripe надає повноцінний API для здійснення платежів та керування платіжними процесами. Це дає розробникам повний контроль над платіжним процесом та можливість інтегрувати платіжну систему Stripe з будь-якими складними функціями вебзастосунку.

Stripe Checkout є готовим рішення від Stripe для швидкої інтеграції платіжної системи. З цим методом можна легко додати кнопку оплати до вебсайту або вебзастосунку, і Stripe відповідатиме за процес оплати. Це досягається шляхом додавання прямого посилання на форму оплати (див. рис. 3.1), яка створюється за допомогою Stripe Dashboard (див. рис. 3.2).

Stripe надає SDK для мобільних платформ, таких як iOS та Android, що дозволяють інтегрувати платіжну систему в мобільні додатки.

Кожен з наведених методів має свої переваги і підходить для різних сценаріїв використання. Вибір конкретного методу залежить від конкретних потреб під час розробки вебзастосунку та вподобань в інтеграції платіжної системи.

Далі наведемо приклад інтеграції Stripe API у проєкт. Цей процес складається з кількох етапів що розглядаються нижче.

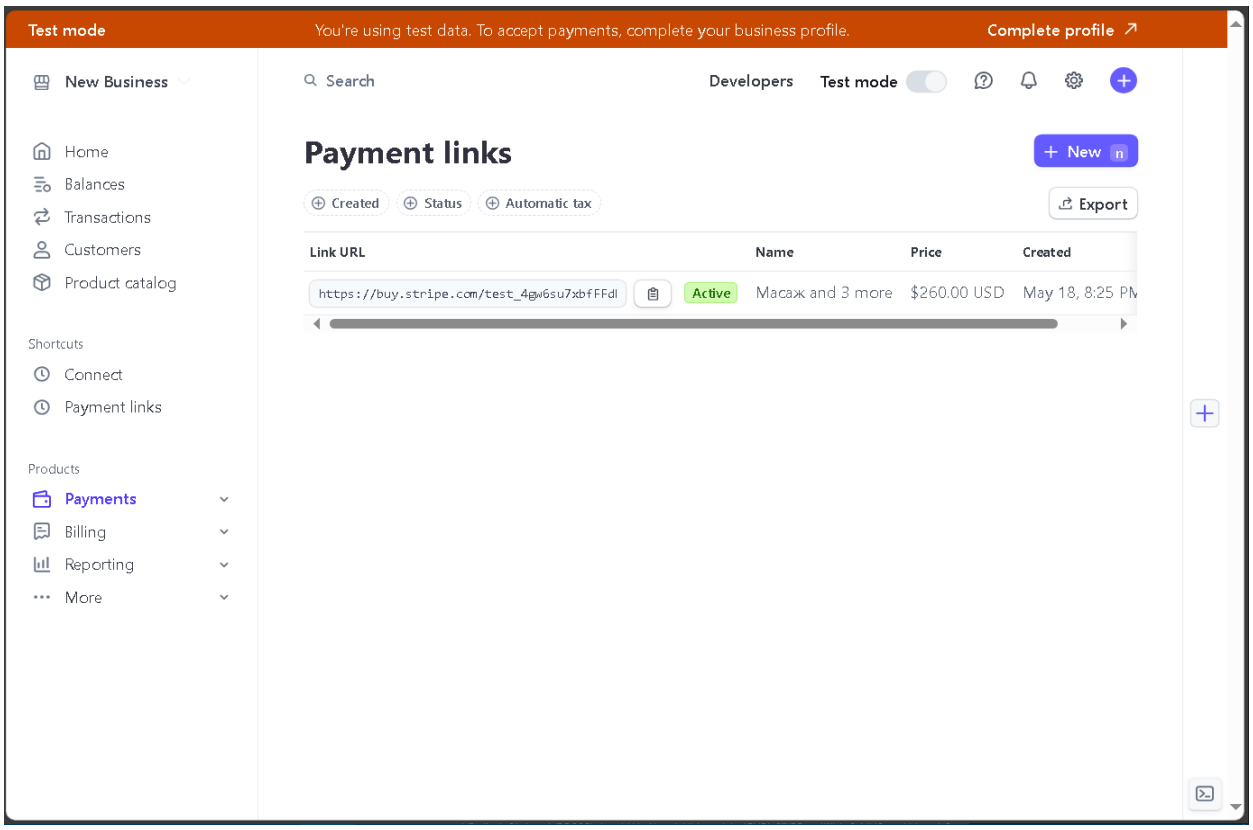

Рисунок 3.1 – Інтерфейс Stripe Dashboard під час налаштування Stripe Checkout

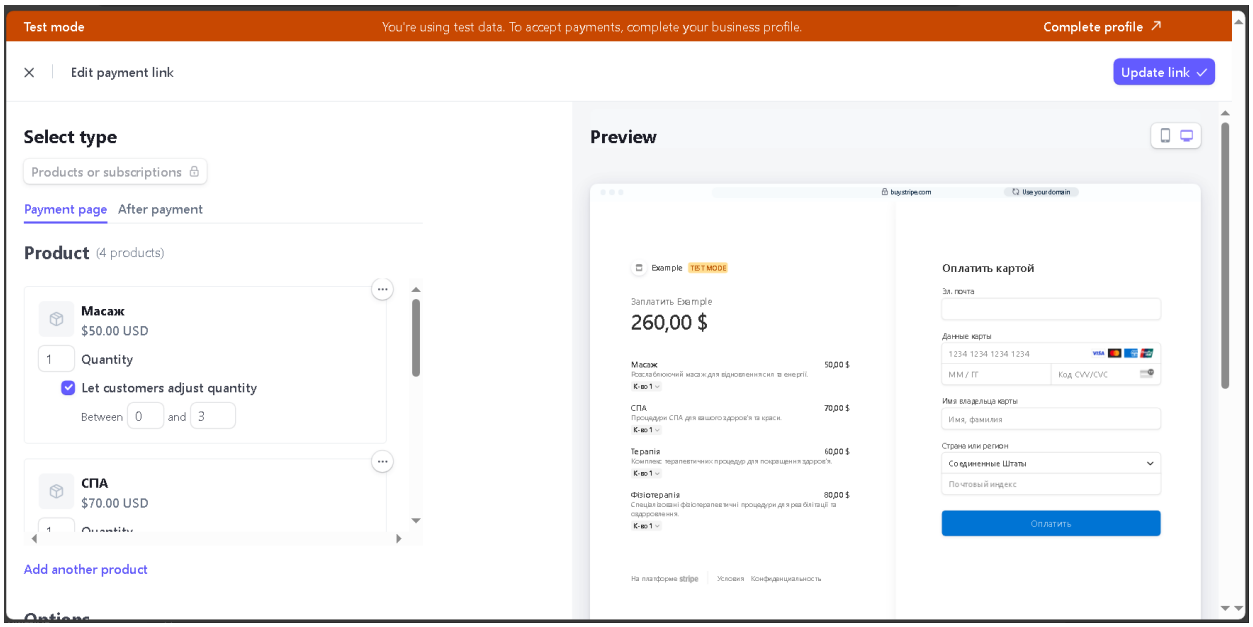

Рисунок 3.2 – Налаштування та попередній перегляд виду форми оплати Stripe Checkout

Робота з Stripe API починається з встановлення бібліотеки Stripe. Для роботи з інструментами Stripe необхідно встановити офіційну бібліотеку Stripe

для Node.js. Це можна зробити за допомогою менеджера пакетів npm, як показано у прикладі (див. рис. 3.3).

```
npm install stripe
```

Рисунок 3.3 – Команда npm для встановлення бібліотеки Stripe

Для початку роботи із бібліотекою Stripe потрібно отримати ідентифікаційні ключі (див. рис. 3.4). Вони потрібні для авторизації користувача та зв'язування його із записом користувача Stripe. Для ініціалізації користувача використовується секретний ключ.

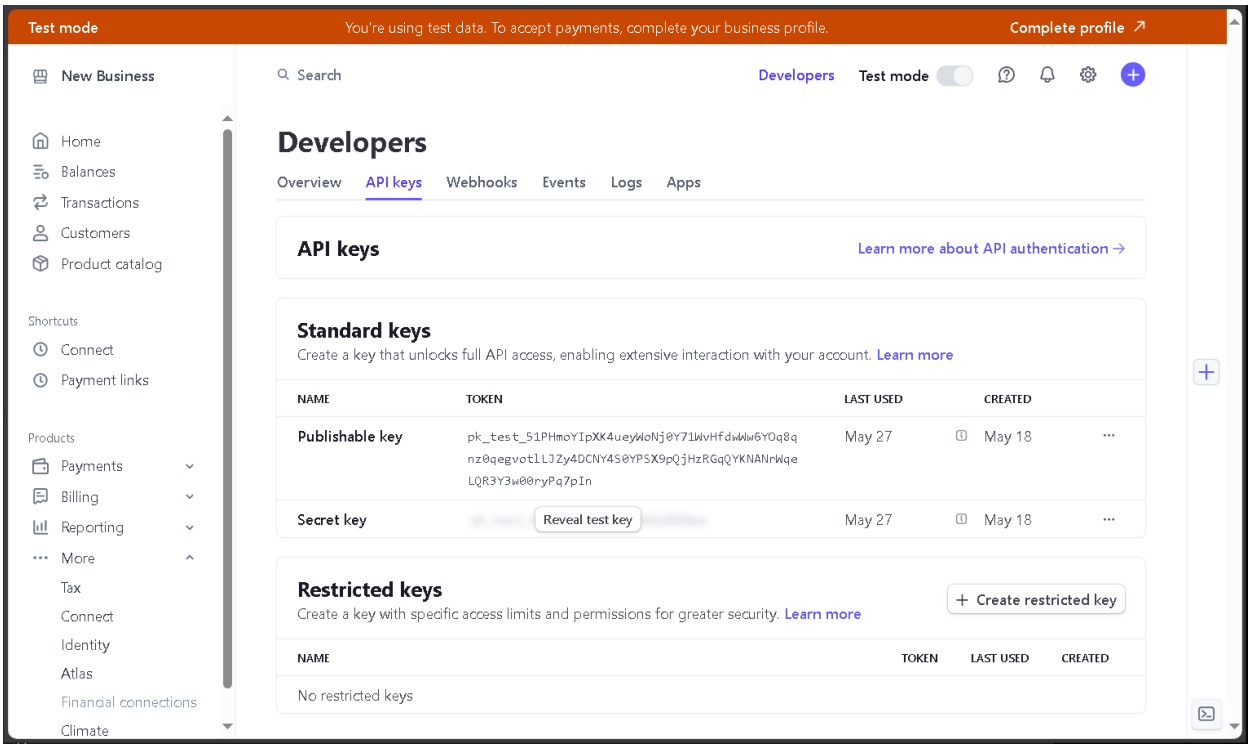

Рисунок 3.4 – Сторінка Stripe Dashboard з інформацією про ідентифікаційні ключі

Stripe надає окремі ключі для тестового і виробничого середовищ.

Тестові ключі (Test Keys) використовуються для тестування інтеграції. Усі операції, виконані з використанням тестових ключів, є симуляціями.

Виробничі ключі (Live Keys) використовуються для реальних транзакцій у виробничому середовищі. Всі операції з цими ключами є справжніми і впливають на реальні фінансові операції.

Приклад тестових ключів:

– публічний тестовий ключ: pk_test_TYooMQauvdEDq54NiTphI7jx;

– секретний тестовий ключ: sk_test_4eC39HqLyjWDarjtT1zdp7dc.

Далі слідує налаштування сервера. На сервері створюється маршрут для обробки запитів на створення платіжних сесій. Сервер використовує бібліотеку Stripe для взаємодії з API Stripe і створення платіжних сесій (див. рис. 3.5).

```
const stripe = require('stripe')('secret-key-here');
app.post('/create-checkout-session', async (req, res) => {
  const { name, email, service, price } = req.body;
  const session = await stripe.checkout.sessions.create({
    payment_method_types: ['card'],
    line_items: [{
      price_data: {
        currency: 'usd',
        product_data: {
          name: service,
        },
        unit_amount: price,
      },
      quantity: 1,
    }],
    mode: 'payment',
```

Рисунок 3.5 – Налаштування маршруту для обробки запитів до Stripe

Далі створимо форми оплати. На клієнтській частині створюється форма для вибору послуг та введення платіжних даних. Форма взаємодіє з сервером, надсилаючи запити для створення платіжних сесій та обробки результатів (див. рис. 3.6).

Інтеграція Stripe в вебзастосунок має ряд переваг, включаючи простоту інтеграції, високий рівень безпеки, гнучкість налаштувань, підтримку різних

платіжних методів, розширені можливості аналітики та звітності, а також локалізацію та підтримку різних валют. Використання цієї платіжної системи дозволяє виконати вимоги до вебзастосунку які були вказані вище.

```
<form id="payment-form">
  <input type="text" id="name" name="name" placeholder="Name" required>
  <input type="email" id="email" name="email" placeholder="Email" required>
  <select id="service" name="service" required>
    <option value="Service1">Service 1 - $100</option>
    <option value="Service2">Service 2 - $150</option>
    <option value="Service3">Service 3 - $200</option>
    <option value="Service4">Service 4 - $250</option>
  </select>
  <button type="submit">Pay Now</button>
</form>

<script>
  const stripe = Stripe('publishable-key-here');

  const paymentForm = document.getElementById('payment-form');
  paymentForm.addEventListener('submit', async (event) => {
    event.preventDefault();

    const { name, email, service } = event.target;
    const price = service.options[service.selectedIndex].dataset.price;

    const response = await fetch('/create-checkout-session', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        name: name.value,
        email: email.value,
        service: service.value,
        price: price,
      }),
    });

    const session = await response.json();

    const result = await stripe.redirectToCheckout({
      sessionId: session.id,
    });
```

Рисунок 3.6 – Створення форми оплати Stripe

3.5 Розробка та реалізація вебзастосунку з інтегрованою платіжною системою

Після визначення потреб та вимог санаторію, вибору відповідних технологій та інструментів, настав час перейти до фактичної розробки та реалізації вебзастосунку з інтегрованою платіжною системою. Цей етап включає в себе створення функціональності, дизайну інтерфейсу та інтеграцію платіжної системи.

Структура вебзастосунку, що відповідає вимогам зазначеним вище, матиме наступний вигляд:

а) головна сторінка:
   1) містить інформацію про санаторій, його послуги та переваги;
   2) відображає список доступних послуг з їх описом та ціною;

б) сторінка оплати:
   1) користувач може переглянути обрану послугу, ввести кількість товару;
   2) інтегрована з платіжним шлюзом stripe для обробки платежів;

в) сторінка оплати Stripe:
   1) форма вводу даних оплати користувача;
   2) користувач підтверджує оплату;
   3) кнопка підтвердження оплати;
   4) відображає повідомлення про успішну оплату або помилку.

Файлова структура проєкту матиме наступний вигляд:

а) Server:
   1) Public:
      – image;
      – cancel.html;
      – index.html;
      – payment.html;
      – script.js;

- styles.css;
- success.html;

2) .env;
3) package.json;
4) package-lock.json;
5) server.js.

Першим кроком є створення функціональності, яка відповідає визначеним вимогам раніше до цього. Це включає в себе реалізацію основних функцій, перегляд інформації про послуги та ціни, оплата за послуги та інші. Кожна функція повинна бути ретельно протестована, щоб забезпечити її коректну роботу та зручність для користувачів.

Для забезпечення приємного та зручного користувацького досвіду необхідно розробити привабливий та інтуїтивно зрозумілий дизайн інтерфейсу. Це включає в себе створення естетичного дизайну, який буде сприяти легкому сприйняттю та використанню вебзастосунку. Дизайн також повинен бути адаптивним, щоб забезпечити коректне відображення на різних пристроях та розширеннях екранів.

Розробка починається з проєктування архітектури системи. Основні компоненти включають фронтенд та бекенд.

Фронтенд включає використання HTML5, CSS3 та JavaScript забезпечує побудову інтуїтивного та адаптивного інтерфейсу. Комбінація HTML5 та CSS3 використовуються для створення компонентів інтерфейсу, що полегшує управління додатком.

Бекенд використовуватиме Node.js з Express.js для створення серверу. Згідно вимог, він має бути маштабований та мати певну швидкодію, що забезпечать інструменти Node.js та Express.js.

Далі зосередимося на розробці фронтенду.

Головна сторінка містить список послуг, що надаються санаторієм. Кожна послуга включає назву, ціну та опис (див. рис. 3.7). При натисканні на послугу користувач перенаправляється на сторінку оплати (див. рис. 3.8).

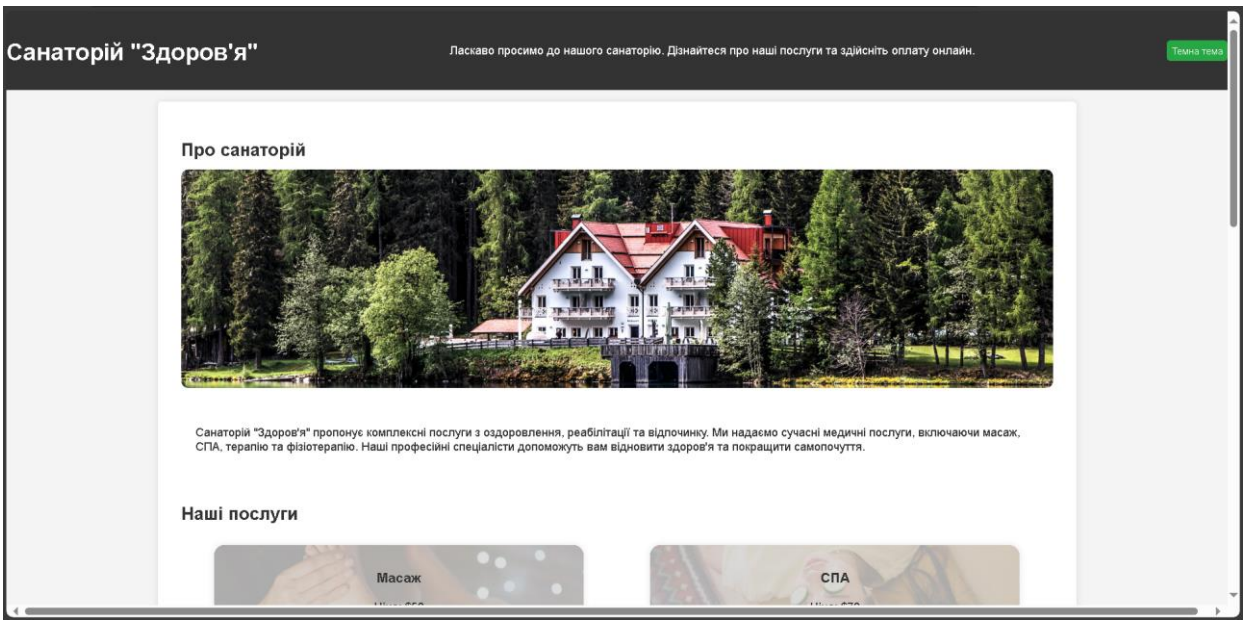

Рисунок 3.7 – Вигляд головної сторінки розроблюваного вебзастосунку

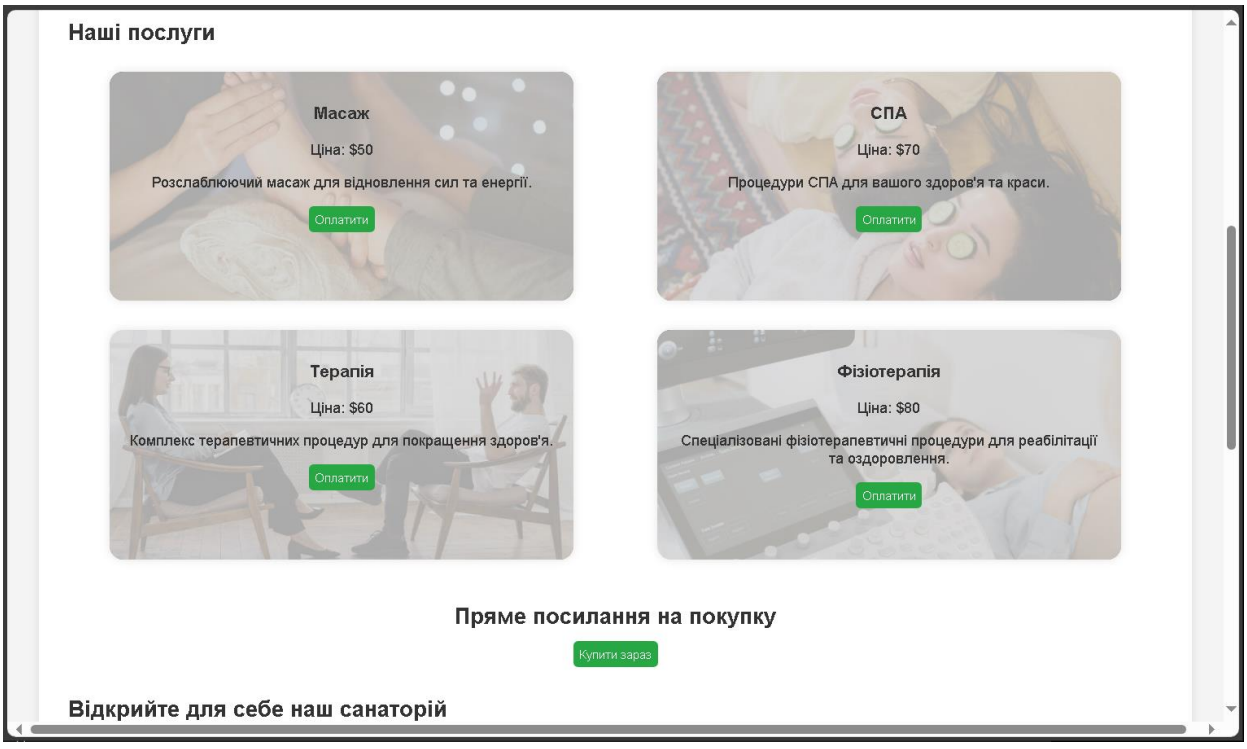

Рисунок 3.8 – Вигляд переліку послуг на головній сторінці розроблюваного вебзастосунку

На головній сторінці, всі послуги розміщуються у комірках з кнопкою «Оплатити». При її натисканні здійснюється перехід на сторінку оплати. Шаблон комірки послуги наведений нижче (див. рис. 3.9).

```
<div class="service-card" style="background-image: url('image/massage.jpg');">
  <div class="card-content">
    <h3>Масаж</h3>
    <p>Ціна: $50</p>
    <p>Розслаблюючий масаж для відновлення сил та енергії.</p>
    <button onclick="location.href='payment.html?id=1'">Оплатити</button>
  </div>
</div>
```

Рисунок 3.9 – Приклад реалізації продукту на головній сторінці

Далі слідує розробка бекенду проєкту.

На сервері створюються маршрути для обробки запитів, пов'язаних з послугами та платіжками. Express.js використовується для налаштування маршрутів. На сервері зберігаються списки послуг з їхніми даними. В такій конфігурації у користувачів не буде прямого доступу до такої інформації як ціна товару.

Послуги зберігаються на сервері у наступному форматі (див. рис. 3.10).

```
const storeItems = new Map([
  [1, { priceInCents: 5000, name: 'Масаж', description: 'Розслаблюючий масаж для відновлення сил та енергії.' }],
  [2, { priceInCents: 7000, name: 'СПА', description: 'Процедури СПА для вашого здоров\'я та краси.' }],
  [3, { priceInCents: 6000, name: 'Терапія', description: 'Комплекс терапевтичних процедур для покращення здоров\'я.' }],
  [4, { priceInCents: 8000, name: 'Фізіотерапія', description: 'Спеціалізовані фізіотерапевтичні процедури для реабілітації та оздоровлення.' }]
])
```

Рисунок 3.10 – Код реалізації зберігання товарів на сервері

Після розробки базового фунціоналу вебзастосунку, наступає інтеграція Stripe.

Stripe забезпечує безпечну обробку платежів та легку інтеграцію з вебзастосунком. Далі наведена інтеграція Stripe у вебзастосунку.

Створимо платіжну сесію на сервері. Сесія зберігатиме дані про назву товару, його кількість (див. рис. 3.11). Сервер обробляє запити на створення платіжних сесій з використанням Stripe API.

```
app.post("/create-checkout-session", async (req, res) => {
 try {
     const session = await stripe.checkout.sessions.create({
       payment_method_types: ["card"],
       mode: "payment",
       line_items: req.body.items.map(item => {
         const storeItem = storeItems.get(item.id)
         return {
           price_data: {
             currency: "usd",
             product_data: {
               name: storeItem.name,
             },
             unit_amount: storeItem.priceInCents,
},
           quantity: item.quantity,
         }
       }),
       success_url: `${process.env.SERVER_URL}/success.html`,
       cancel_url: `${process.env.SERVER_URL}/cancel.html`,
     })
     res.json({ url: session.url })
   } catch (e) {
     res.status(500).json({ error: e.message })
```

Рисунок 3.11 – Код створення платіжної сесії на сервері

Сесія складається з наступних компонетів:

– payment_method_types: ["card"];

– mode: "payment";

– line_items;

– success_url та cancel_url.

Коли клієнт надсилає POST-запит до /create-checkout-session з товарами в тілі запиту, сервер створює нову сесію Stripe, налаштовує її з відповідними товарами, їх цінами та кількостями, а також URL-адресами для успіху та скасування. Якщо сесію створено успішно, сервер повертає клієнту URL для перенаправлення на Stripe. У випадку помилки сервер повертає повідомлення про помилку.

Налаштуємо переадресацію на сторінку оплати. Як вже було показано на етапі розробки фронтенду (див. рис. 3.9), користувач перенаправляється на сторінку оплати. Механізм перенаправлення показаний нижче (див. рис. 3.12).

```
<button onclick="location.href=‘payment.html?id=2’">Оплатити</button>
```

Рисунок 3.12 – Код переадресації на сторінку оплати

Інтерфейс сторінки оплати має два поля: ідентифікаційний код послуги та кількість одиниць товару. Цей функціонал потрібен для того щоб користувач перевірив, чи обрав він бажану послугу та вказати кількість товару, в залежності від того скільки осіб користуватиметься послугами санаторію. Якщо вся інформація вірна, користувач має натиснути кнопку «Продовжити». Сторінка оплати матиме наступний вигляд (див. рис. 3.13).

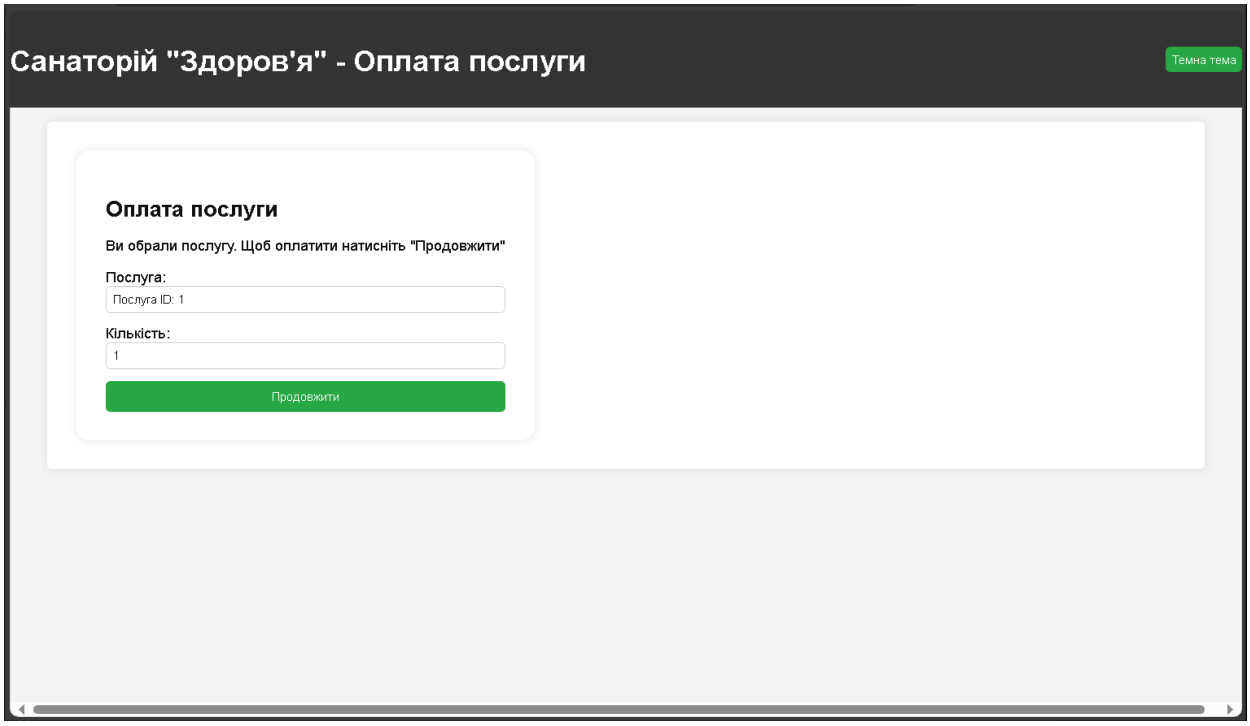

Рисунок 3.13 – Вигляд сторінки оплати розроблюваного вебзастосунку

Переадресація на сторінку оплати Stripe здійснюється наступним чином (див. рис. 3.14).

```
<button type="submit">Продовжити</button>
```

Рисунок 3.14 – Код переадресації на сторінку оплати Stripe

Після створення основної функціональності та інтерфейсу вебзастосунку настає час для інтеграції платіжної системи. Це включає в себе налаштування API платіжної системи, створення форм оплати та обробку платежів. Під час інтеграції необхідно забезпечити безпеку та надійність транзакцій, а також врахувати різні платіжні методи та валюти.

Інтеграція платіжної системи на стороні користувача показана нижче (див. рис. 3.15).

```
paymentForm.addEventListener('submit', async (event) => {
    event.preventDefault();
    console.log('Form submitted');
    const selectedQuantity = document.getElementById('quantity').value;
    fetch("/create-checkout-session", {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        items: [
          { id: parseInt(serviceId), quantity: parseInt(selectedQuantity) }
        ]
      })
    })
    .then(res => {
      if (res.ok) return res.json();
      return res.json().then(json => Promise.reject(json));
    })
    .then(({ url }) => {
      window.location = url;
    })
```

Рисунок 3.15 – Код реалізації інтеграція платіжної системи на стороні користувача

Після натискання кнопки «Продовжити» перед очами постає форма оплати Stripe (див. рис. 3.16). Її тестування наведене у наступному розділі.

Далі, покажемо як працює залежність кількості товару на зміну кінцевої суми. Повернемося на крок назад та встановимо кількість товару з ідентифікаційним кодом 1 – 4 одиниці (див. рис. 3.17).

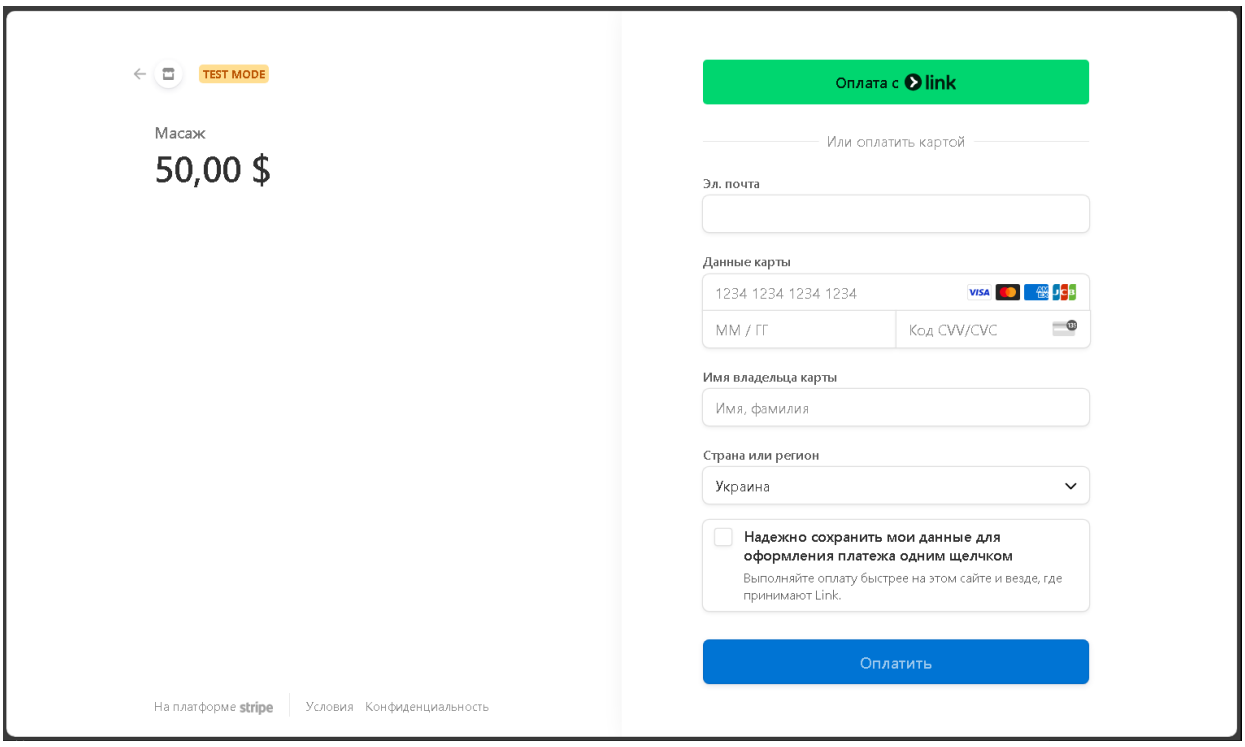

Рисунок 3.16 – Вигляд сторінки оплати Stripe

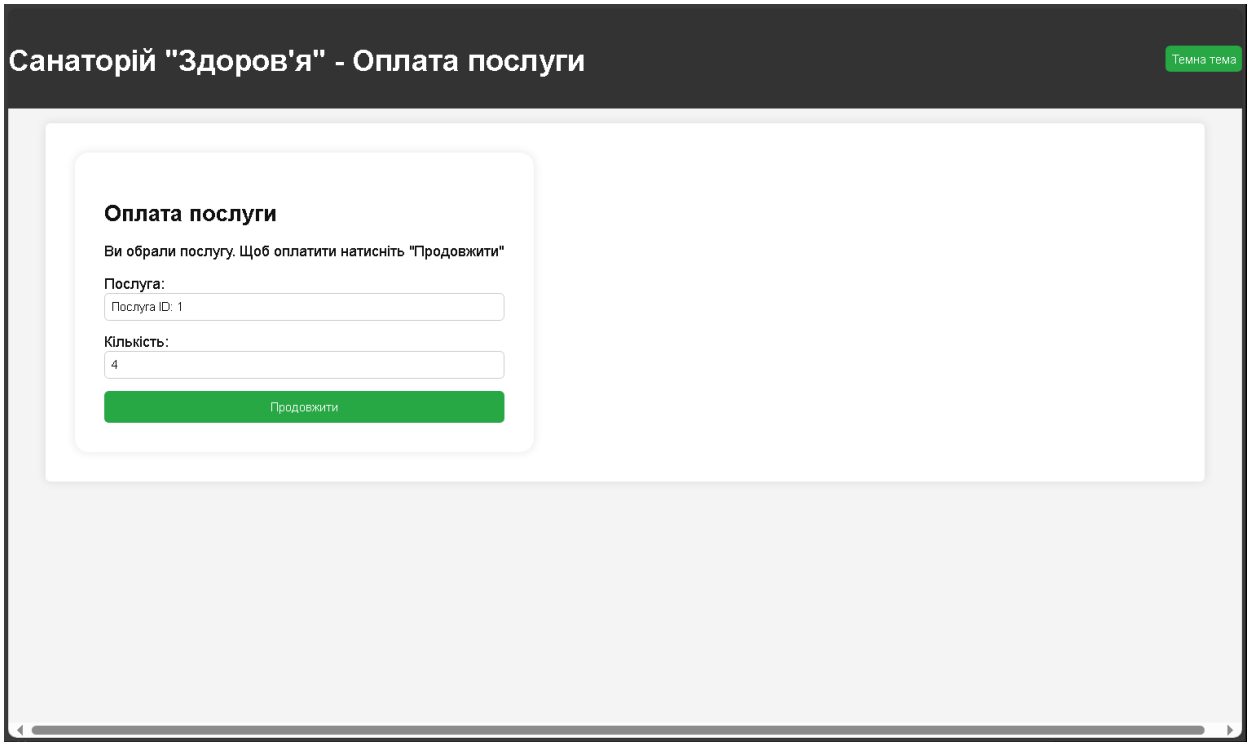

Рисунок 3.17 – Вигляд заповненої сторінки оплати вебзастосунку

З рисунку наведеного нижче (див. рис. 3.18), можна побачити що, ціна змінилася.

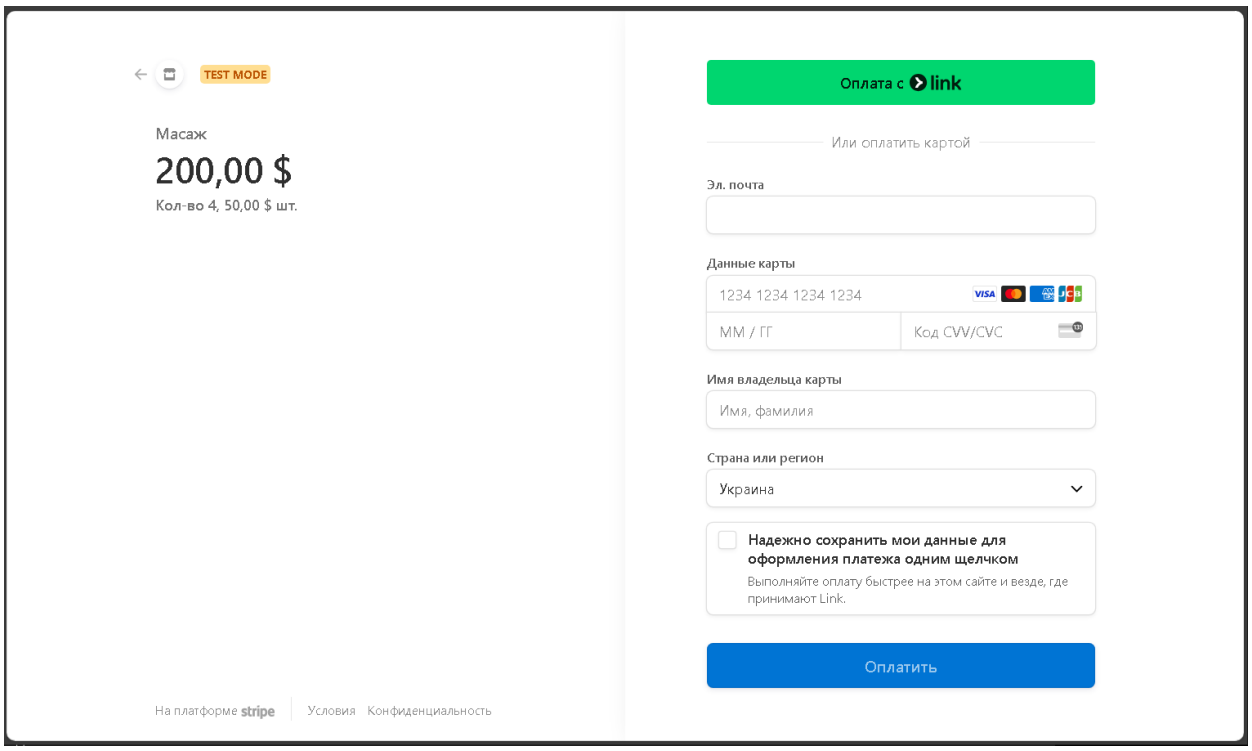

Рисунок 3.18 – Вигляд сторінки оплати Stripe з іншими даними про кількість товару

У випадках успішної або невдалої оплати товару застосунок має два варіанти поведінки, відповідно. При відміні оплати користувач побачить сторінку з невдалою оплатою (див. рис. 3.19).

Рисунок 3.19 – HTML-сторінка з повідомленням про невдалу оплату

У разі успішності трансакції, користувач спостерігатиме сторінку успішної оплати (див. рис. 3.20).

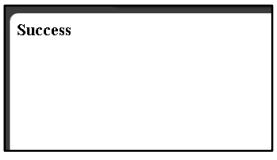

Рисунок 3.20 – HTML-сторінка з повідомленням про вдалу оплату

Фрагмент коду що відповідає за цю поведінку знаходиться на сервері. У даному випадку це має таку реалізацію, проте можливо встановити будь-який вебсайт для переходу після оплати товару. Наведений нижче код відповідає за цю функцію (див. рис. 3.21).

```
  }),
  success_url: `${process.env.SERVER_URL}/success.html`,
  cancel_url: `${process.env.SERVER_URL}/cancel.html`,
})
```

Рисунок 3.21 – Код переадресації в залежності від успішності оплати

Додатково, на вебсайті присутня кнопка оплати товару, реалізована інакшим чином – за допомогою Stripe Checkout (див. рис. 3.22).

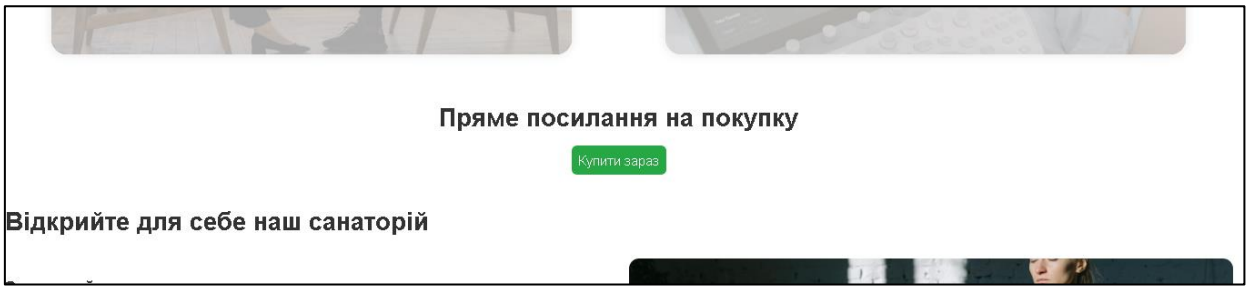

Рисунок 3.22 – Кнопка оплати товару, реалізована через Stripe Checkout на головній сторінці сайту

Stripe Checkout надає готовий платіжний інтерфейс, який можна легко інтегрувати у вебзастосунок за допомогою URL-посилання. Інтерфейс включає всі необхідні елементи для введення платіжних даних, а також підтримує адаптивний дизайн, що робить його зручним для використання на різноманітних пристроях (див. рис. 3.23).

Узагальнюючи, розробка та реалізація вебзастосунку з інтегрованою платіжною системою вимагає комплексного підходу та використання сучасних технологій та методів розробки. Важливо дотримуватися кращих практик у кожному етапі розробки, щоб забезпечити якість та ефективність результуючого вебзастосунку.

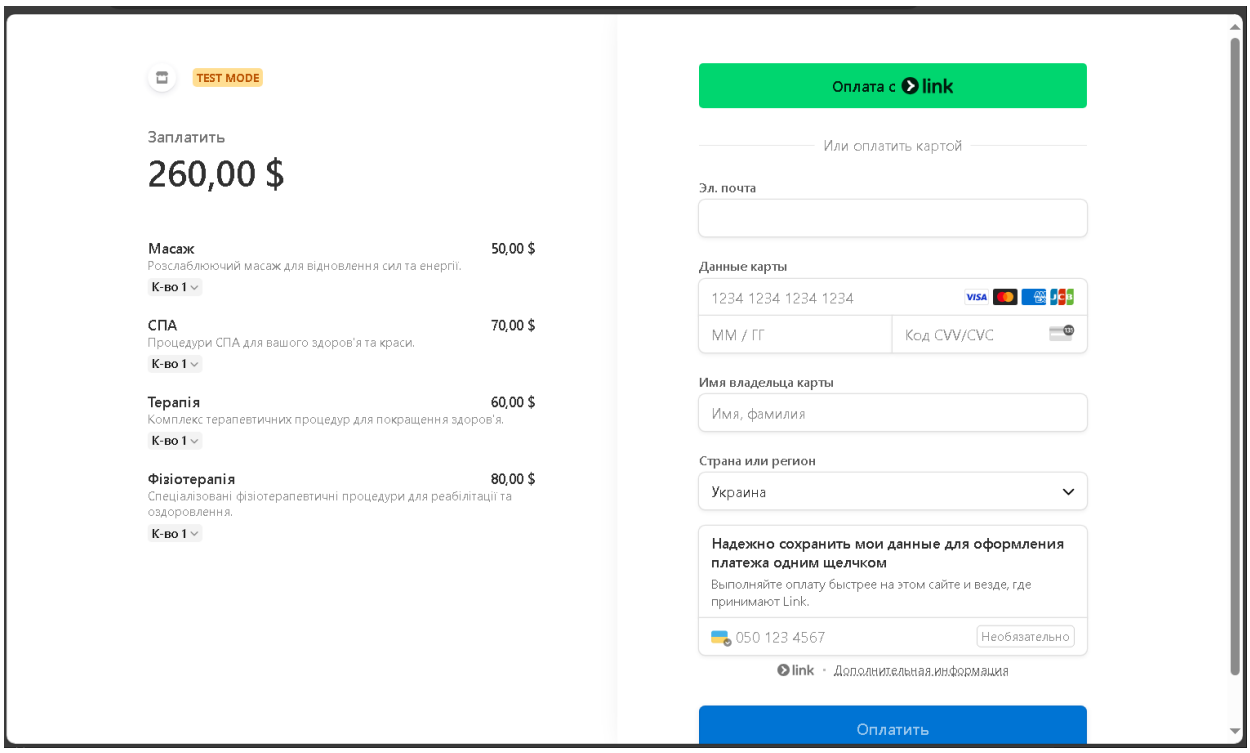

Рисунок 3.23 – Форма оплати реалізована через Stripe Checkout

3.6 Тестування та валідація роботи вебзастосунку та платіжної системи

Завершальним етапом розробки вебзастосунку є проведення тестування та валідації всіх його компонентів. Це включає функціональне тестування, щоб переконатися, що всі функції працюють коректно, а також тестування безпеки, щоб виявити та усунути потенційні уразливості. Після успішного завершення тестування вебзастосунок готовий до експлуатації та використання користувачами санаторію.

Після завершення розробки вебзастосунку та інтеграції платіжної системи настає етап тестування та валідації. Ціллю тестування є перевірка коректності функціонування всіх компонентів вебзастосунку відповідно до вимог. Цей етап важливий для переконанняся у правильній роботі компонентів системи та зручності використання для користувачів.

Проведемо ручне тестування фунціоналу системи. Це включає

виконання тестових сценаріїв вручну для перевірки основних функцій, таких як вибір послуг, введення даних, оформлення замовлення тощо.

Першим кроком оберемо товар під назвою «Масаж» (див. рис. 3.24). Його ціна вказана як 50 доларів США. Натиснемо кнопку «Оплатити» щоб перейти до оплати товару.

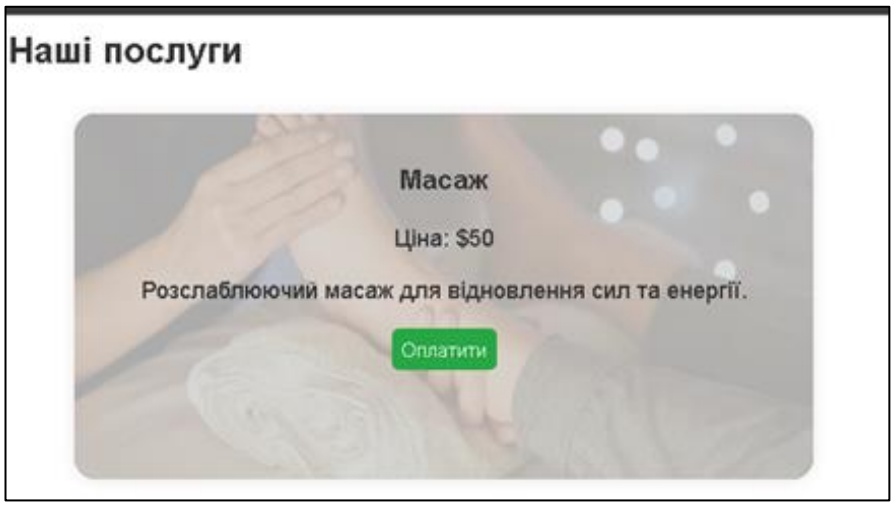

Рисунок 3.24 – Вид послуги на головній сторінці сайту

Сторінка оплати надає наступну інформацію: Ідентифікаційний код послуги – 1. Встановимо кількість одиниць товару – 4. Після цього можна натиснути кнопку «Продовжити». Зображення для супроводу дії було наведено у попередньому пункті.

Далі заповнимо форму оплати. Для того, щоб успішно здійснити платіж через інтегровану платіжну систему, користувач повинен ввести дані у форму оплати. Це включатиме наступну інформацію:

а) інформація про платіжну картку:
   1) номер картки: 16-значний номер платіжної картки;
   2) термін дії картки: місяць і рік, до якого картка дійсна;
   3) CVC код: трьохзначний код безпеки на зворотному боці картки (для American Express це може бути чотири цифри);

б) інформація про власника картки:
   1) ім’я та прізвище власника картки: як вказано на картці;

2) адреса виставлення рахунку: треба зазначити країну або регіон перебування;

в) контактна інформація:

1) електронна пошта: для відправки підтвердження платежу та квитанції;
2) телефонний номер: може використовуватися для зв’язку з користувачем у випадку виникнення питань щодо платежу.

Для того щоб протестувати платіжну системи Stripe необхідно використовувати спеціальні тестові дані, які Stripe надає для цього. Тестові дані дозволяють симулювати різні сценарії платежів без фактичного здійснення фінансових транзакцій.

Основні тестові картки які надає Stripe:

а) успішний платіж:

1) номер картки: 4242 4242 4242 4242;
2) термін дії: будь-яка дата у майбутньому (наприклад, 12/34);
3) cvc: будь-які три цифри (наприклад, 123);

б) платіж з помилкою автентифікації:

1) номер картки: 4000 0000 0000 9995;
2) термін дії: будь-яка дата у майбутньому;
3) cvc: будь-які три цифри;

в) недостатньо коштів:

1) номер картки: 4000 0000 0000 9995;
2) термін дії: будь-яка дата у майбутньому;
3) cvc: будь-які три цифри;

г) відхилена картка:

1) номер картки: 4000 0000 0000 0002;
2) термін дії: будь-яка дата у майбутньому;
3) cvc: будь-які три цифри;

д) картка, що вимагає додаткову автентифікацію (3d secure):

1) номер картки: 4000 0025 0000 3155;

2) термін дії: будь-яка дата у майбутньому;

3) cvc: будь-які три цифри.

Використаємо тестові дані для отримання успішний платіжу. Сторінка оплати матиме наступний вигляд (див. рис. 3.25).

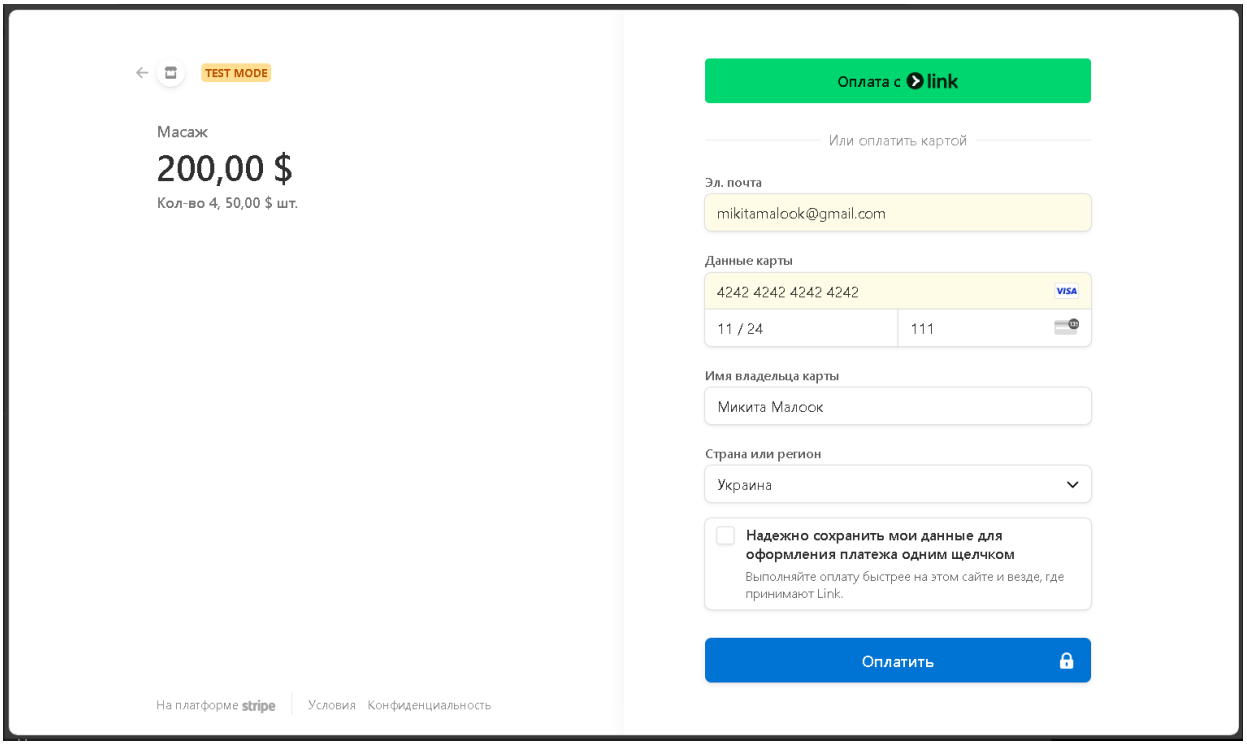

Рисунок 3.25 – Форма оплати Stripe, заповнена тестовими даними

Після натискання кнопки «Сплатити» можна спостерігати результатами які повертане система. Згідно коду додатку, вебсайт повертає сторінку результату успішної оплати (див. рис. 3.26).

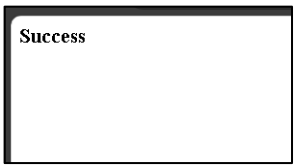

Рисунок 3.26 – Відповідь серверу, про успішну оплату

Додатково перевіримо роботи прямого посилання на покупку товарів. Встановимо кількість усіх товарів – 1. Тестові дані залишаються аналогічними (див. рис. 3.27).

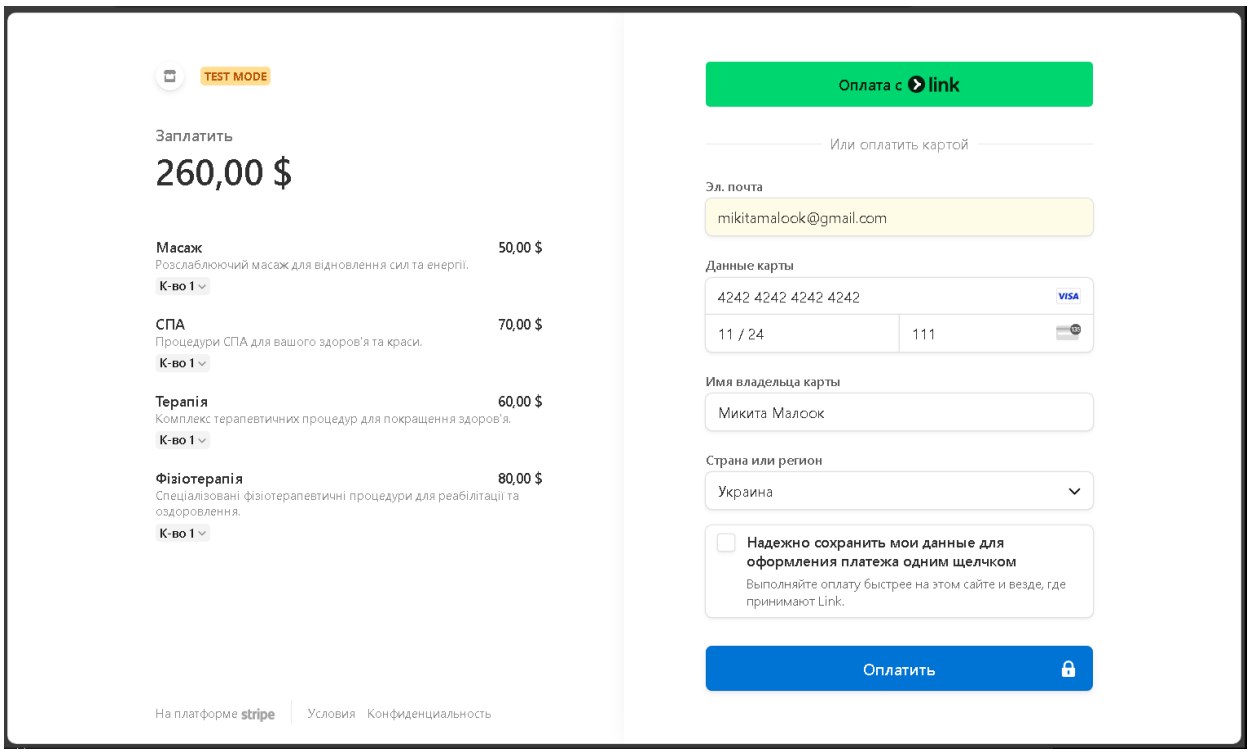

Рисунок 3.27 – Форма оплати Stripe Checkout, заповнена тестовими даними

Так як механізм прямої покупки, на всьому етапі виконання, обробляється виключно за допомогою Stripe, то він повертає повідомлення про успішну оплату всередині форми оплати (див. рис. 3.28).

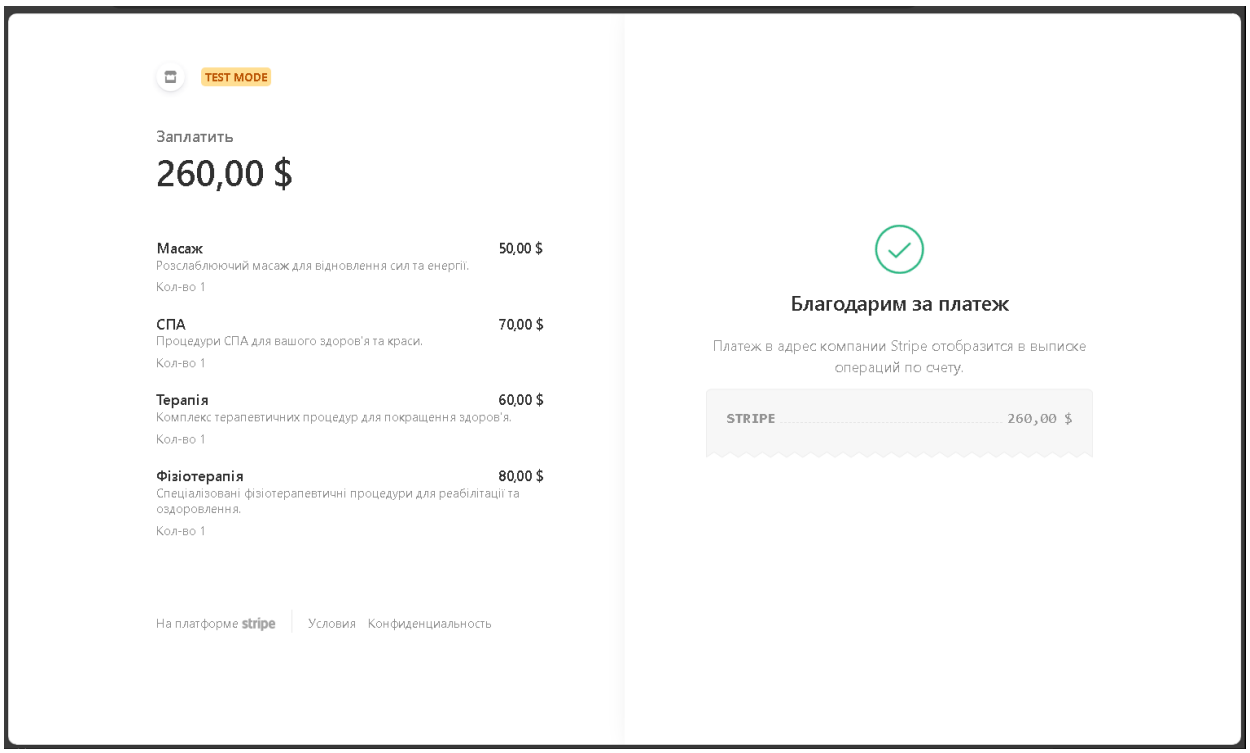

Рисунок 3.28 – Відповідь Stripe, про успішну оплату

Перевіримо виконання трансакцій використовуючи інстументи Stripe. Stripe Dashboard дозволяє користувачам переглядати всі транзакції, здійснені через систему. Це включає як успішні платежі, так і невдалі спроби. Транзакції можуть бути фільтровані за різними критеріями, такими як дата, сума, статус тощо. Це дозволяє легко відстежувати поточний стан платежів та ідентифікувати потенційні проблеми (див. рис. 3.29).

На панелі «Transactions» можна побачити записи про проведені трансакції. Наразі це два записи про проведені раніше, тестові платежі (див. рис. 3.30, 3.31).

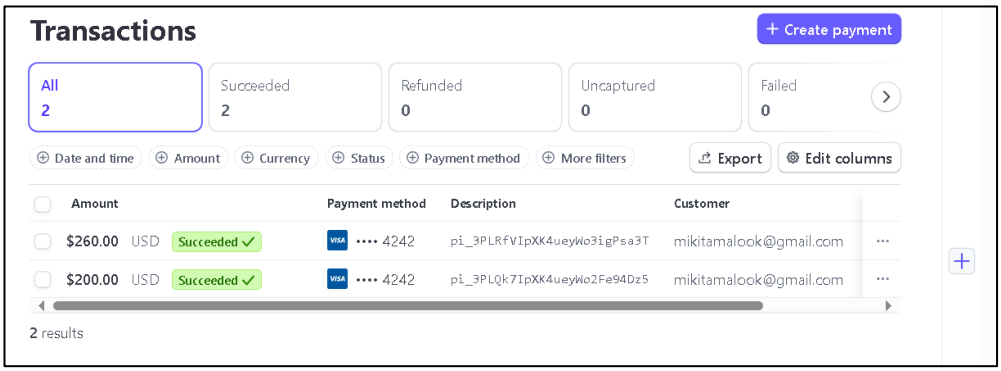

Рисунок 3.29 – Панель Stripe Dashboard – «Transactions»

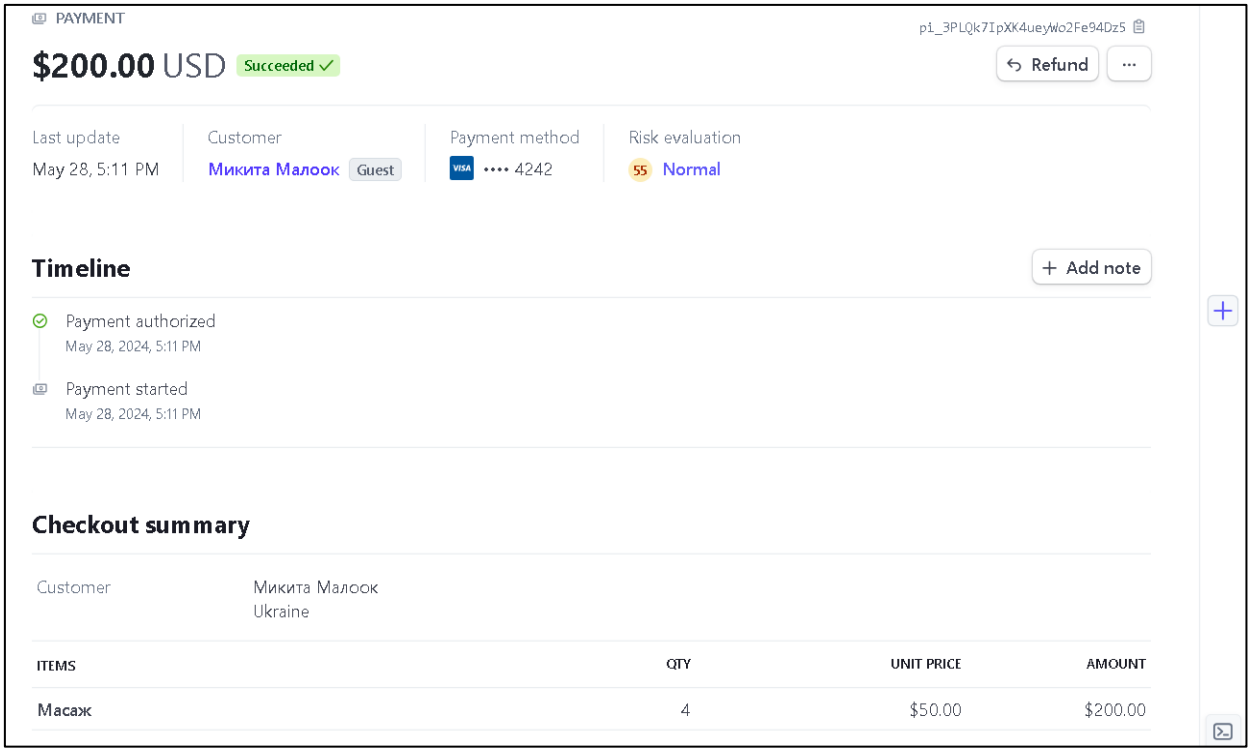

Рисунок 3.30 – Результат XML запиту LanguagesGet

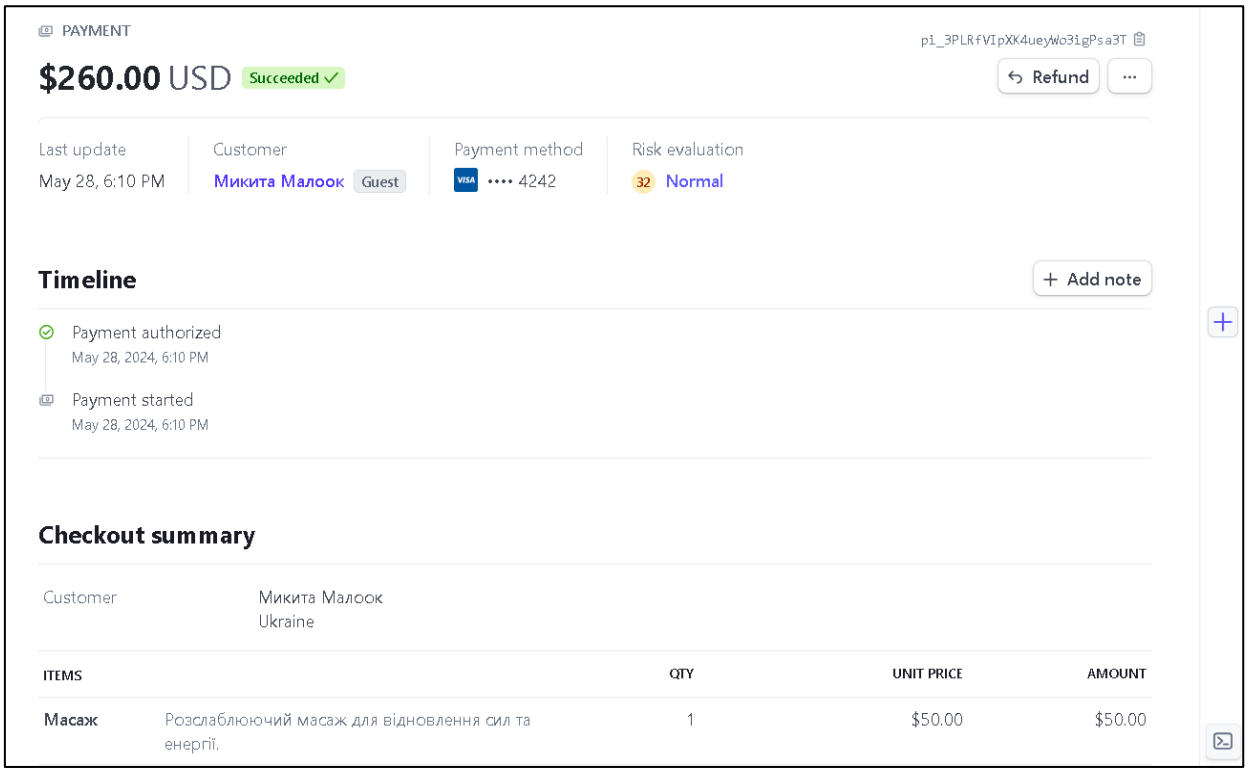

Рисунок 3.31 – Результат XML запиту LanguagesGet

Далі проведемо кросбраузерне тестування.

Оскільки користувачі можуть використовувати вебзастосунок на різних пристроях та в різних браузерах, важливо перевірити його сумісність з різними платформами. Це включає тестування на різних браузерах: Chrome, Firefox, Edge.

Перевірені браузери:

– Google Chrome;
– Mozilla Firefox;
– Microsoft Edge.

Перевірені пристрої:

– десктопи (Windows, macOS);
– мобільні пристрої (iOS, Android).

Перевірені функції:

а) відображення сторінок:

1) головна сторінка;

2) сторінка з інформацією про послуги;
3) сторінка оформлення замовлення;

б) функціональність форми:
1) введення інформації про платіжну картку;
2) введення інформації про власника картки;
3) введення контактної інформації;
4) натискання кнопки «Продовжити» та обробка результатів;
5) натискання кнопки «Сплатити» та обробка результатів;

в) реакція на різні сценарії платежів:
1) успішний платіж;
2) недостатньо коштів на картці;
3) відхилення платежу;
4) додаткова автентифікація (3D Secure);

г) відповідність дизайну:
1) коректне відображення шрифтів, кольорів, кнопок та інших елементів UI;
2) адаптивність дизайну на мобільних пристроях;

д) взаємодія користувача:
1) навігація між сторінками;
2) клікання на елементи;
3) відображення повідомлень про успішні або неуспішні платежі.

Оглянемо результати.

Google Chrome:

– десктоп: всі сторінки відображаються коректно, форми працюють без помилок, успішно оброблено всі сценарії платежів;
– мобільні пристрої: адаптивний дизайн працює належним чином, жодних проблем з функціональністю не виявлено.

Mozilla Firefox:

– десктоп: всі функції працюють без помилок, дизайн сторінок відповідає вимогам, успішно оброблено всі сценарії платежів;

– мобільні пристрої: відображення сторінок коректне, жодних проблем з функціональністю не виявлено.

Microsoft Edge:

– десктоп: всі сторінки відображаються і працюють правильно, успішно пройдені всі тести на функціональність;
– мобільні пристрої: адаптивний дизайн і функціональність працюють без проблем.

Отже, можемо зробити наступні висновки.

Відображення сторінок: усі перевірені браузери корректно відображають сторінки вебзастосунку як на десктопах, так і на мобільних пристроях. Дизайн є адаптивним і зручним для користувачів різних пристроїв.

Функціональність форми: всі основні браузери успішно обробляють дані, введені у форму оплати. Жодних проблем з введенням або обробкою даних не виявлено.

Обробка платежів: успішно пройдені всі сценарії платежів у всіх браузерах. Відображення повідомлень про статус платежів коректне.

Реакція на різні сценарії платежів: система коректно обробляє запити на різні сценарії платежів у всіх перевірених браузерах.

Відповідність дизайну: проблем з форматуванням елементів не були виявлені. В цілому дизайн відповідає вимогам в всіх браузерах.

Підсумовуючи, тестування та валідація функціоналу вебзастосунку є критичними етапами в процесі розробки. Вони забезпечують надійність, безпеку та стабільність розроблюваного продукту.

3.7 Висновки до розділу 3

У третьому розділі було детально розглянуто процес розробки та тестування вебзастосунку з інтегрованою платіжною системою для санаторію. Цей розділ охопив такі теми, як аналіз потреб та вимог санаторію,

проєктування функціональності та інтерфейсів, вибір технологій та інструментів для реалізації, а також тестування і валідацію роботи вебзастосунку та платіжної системи.

Був виконаний аналіз потреб та вимог санаторію, що дозволив визначити основні функціональні вимоги до вебзастосунку. Детально і покроково розглядається розробка програмного продукту. Під час розробки було надано увагу якості програмного забезпечення, всі компоненти системи були ретельно протестовані. Внаслідок цього вдалося реалізувати поставлену задачу – функціональний вебзастосунок з інтегрованою системою оплати.

ВИСНОВКИ

У процесі виконання дипломної роботи було розглянуто основи роботи та проблеми платіжних систем, а також детально проаналізовано їх види, структура, технології. Зокрема, було проаналізовано стандарти, які забезпечують безпеку і надійність платіжних транзакцій.

Особлива увага приділена основним компонентам платіжних систем та аспектам, які необхідно враховувати при інтеграції платіжних сервісів у вебзастосунки. Проаналізовано різні архітектурні підходи до інтеграції платіжних систем, включаючи вебсервіси та платіжні шлюзи. На основі проведеного аналізу обрано платіжний шлюз Stripe.

У ході проєктування вебзастосунку для санаторію було враховано вірогідні вимоги користувачів. Внаслідок цього, для розроблюваного додатку та платіжної системи було спроєктовано функціональність, що включає інформаційну сторінку, каталог послуг та форму оплати. Реалізація вебзастосунку передбачала інтеграцію платіжної системи для обробки онлайн-платежів.

Проведено тестування вебзастосунку, включаючи, ручне тестування, кросбраузерне тестування та тестування платіжної системи за допомогою тестових даних Stripe. Результати тестування показали, що вебзастосунок працює коректно та відповідає всім визначеним вимогам.

Загалом, результати роботи показують, що розроблений вебзастосунок для санаторію з інтегрованою платіжною системою Stripe відповідає всім визначеним вимогам. Використання сучасних технологій та методів дозволило створити ефективний інструмент для обробки платіжних транзакцій, який відповідає сучасним стандартам безпеки та зручності.

Проведене дослідження довело, що запропонований підхід до інтеграції платіжних систем у вебзастосунки є ефективним і може бути використаний для подальшого розвитку платіжних систем.

ПЕРЕЛІК ПОСИЛАНЬ

1. До питання визначення поняття «платіжна система» в Україні. URL: http://pgp-journal.kiev.ua/archive/2017/10/32.pdf (дата звернення: 23.02.2024).
2. Еволюція платіжних систем. URL: https://tinyurl.com/4vfhrnc8 (дата звернення: 23.02.2024).
3. Чайковський Я. І. Платіжні системи : навчальний посібник. Тернопіль, 2008. С. 18–66. URL: http://dspace.wunu.edu.ua/bitstream/316497/536/1/платіжні%20системи.pdf (дата звернення: 24.02.2024).
4. Що таке платіжна система та які з них працюють в Україні. Сучасні платіжні рішення для вашого бізнесу. URL: https://fondy.ua/uk/knowledge/payment-system/ (дата звернення: 28.02.2024).
5. Mastercard New Payments Index: Consumer Appetite for Digital Payments Takes Off. Mastercard Incorporated – Investor Relations. URL: https://investor.mastercard.com/investor-news/investor-news-details/2021/Mastercard-New-Payments-Index-Consumer-Appetite-for-Digital-Payments-Takes-Off/ (дата звернення: 29.02.2024).
6. How to accept contactless NFC payments from customers | Stripe. URL: https://stripe.com/resources/more/how-to-accept-contactless-nfc-payments-from-customers (дата звернення: 30.02.2024).
7. Підходи до класифікації платіжних систем в Україні. URL: http://www.pdu-journal.kpu.zp.ua/archive/4_2018/15.pdf (дата звернення: 03.03.2024).
8. How Visa & Mastercard have Impacted the Payment Industry? | LinkedIn. URL: https://www.linkedin.com/pulse/how-visa-mastercard-have-impacted-payment-industry-webpays-dp3bc (дата звернення: 05.03.2024).
9. Digital wallets: How they work and how to accept them | Stripe. URL: https://stripe.com/resources/more/digital-wallets-101 (дата звернення: 07.03.2024).

10. Mobile payment is not all the same: The adoption of mobile payment systems depending on the technology applied. URL: https://openaccess.uoc.edu/bitstream/10609/112806/1/RamosLuna_TFSC2019_Mobilepayment.pdf (дата звернення: 08.03.2024).
11. Bitcoin – Open source P2P money. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 08.03.2024).
12. Payment security explained: A guide for businesses | Stripe. URL: https://stripe.com/resources/more/payment-security (дата звернення: 11.03.2024).
13. PCI Security Standards Council – Protect Payment Data with Industry-driven Security Standards, Training, and Programs. URL: https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf (дата звернення: 12.03.2024).
14. Міжнародні стандарти та історія EMV | Mastercard Ukraine. URL: https://www.mastercard.ua/uk-ua/business/merchants/safety-security/emv-chip.html (дата звернення: 15.03.2024).
15. Get Started with PayPal REST APIs. PayPal API reference. URL: https://developer.paypal.com/api/rest/ (дата звернення: 28.03.2024).
16. Правове регулювання платіжних систем. URL: http://lsej.org.ua/10_2023/171.pdf (дата звернення: 06.04.2024).
17. ISO – International Organization for Standardization. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-2:v1:en (дата звернення: 11.04.2024).
18. Payment systems. European Central Bank. URL: https://www.ecb.europa.eu/paym/pol/systems/html/index.en.html (дата звернення: 15.04.2024).
19. Stripe API Reference | Stripe. URL: https://docs.stripe.com/api (дата звернення: 18.05.2024).
20. Documentation. Stripe-Documentation. URL: https://docs.stripe.com/ (дата звернення: 18.05.2024).