



Универзитет „Св. Кирил и Методиј“ во Скопје  
**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И  
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

## ДИПЛОМСКА РАБОТА

Тема:

***Платформа за е-трговија „Neura Cartix“  
базирана на вештачка интелигенција***

Ментор:

проф. д-р Димитар Трајанов

Изработил:

Јована Чочевска - 201107

e-mail: [jovana.chochevska@students.finki.ukim.mk](mailto:jovana.chochevska@students.finki.ukim.mk)

*Скопје, декември 2024*

## Содржина:

Апстракт .....	3
Вовед .....	4
Краток опис на апликацијата .....	5
Користени технологии .....	5
Java spring boot.....	6
JS – React .....	7
PostgreSQL .....	8
Open AI api .....	8
Архитектура на решението .....	10
Имплементација на решението .....	12
База на податоци.....	12
Backend.....	13
Models (модели) .....	13
JPA Repository (репозиториуми) .....	13
Services (сервиси) .....	14
Controllers (контролери).....	14
Stripe (сервис за наплата).....	15
Authentication (автентикација и авторизација) .....	16
Frontend .....	18
Интеграција со Open AI .....	19
Кориснички интерфејс.....	22
Подобрувања и нови функционалности .....	33
Заклучок .....	35
Користена литература.....	36

## Апстракт

*Електронската трговија стана суштинска компонента на модерниот бизнис, нудејќи неспоредлива погодност, пристапност и ефикасност и за клиентите и за бизнисите. Оваа теза го претставува дизајнот и имплементацијата на сеопфатна апликација за е-трговија која комбинира робусни карактеристики за купување фокусирани на корисниците со напредни административни алатки за ефективно управување. Апликацијата поддржува непречено прелистување производи, безбедна обработка на плаќањата и следење на нарачките во реално време, а исто така им овозможува на администраторите да управуваат со нарачките, да генерираат фактури (import и export на документи) и ефикасно да ракуваат со корисничките податоци.*

*Една од најистакнатите карактеристики на апликацијата е интеграцијата на интелигентен чет-бот за препораки, напојуван со OpenAI API, кој обезбедува персонализирани предлози за производи врз основа на преференциите и интеракциите на корисниците. Од техничка перспектива, оваа теза ја истражува архитектурата и деталите за имплементацијата, вклучувајќи ги клучните методологии, рамки и API. Работата исто така вклучува објаснувања за базата на кодови, нагласувајќи ја основната логика зад основните функционалности и содржи детални слики од екранот на корисничкиот интерфејс за да се обезбеди визуелен контекст.*

*Идната работа на апликацијата се фокусира на подобрување на приспособливоста и перформансите преку контејнеризација, транзиција кон архитектура на микроуслуги и хостирање на платформата на облак инфраструктура. Овие подобрувања имаат за цел да ја направат апликацијата поприлагодлива и способна да се справи со растечките барања на корисниците.*

*Оваа теза покажува како современите технологии и принципите на дизајн може да се искористат за да се создаде динамична, прифатлива за корисниците и административно ефикасна платформа за е-трговија.*

## Вовед

Во модерната ера на дигитална трансформација, е-трговијата се појави како доминантна сила, револуционизирајќи го начинот на кој бизнисите работат и клиентите доаѓаат до производите и услугите. Брзиот напредок на технологијата и зголеменото потпирање на дигиталните платформи создадоа потреба за софистицирани решенија за е-трговија кои се грижат и за удобноста на клиентите и за деловната ефикасност. Оваа дипломска работа го претставува развојот на сеопфатна апликација за е-трговија која интегрира непрекорни функционалности за купување со напредни административни алатки и автоматизација управувана од вештачка интелигенција се со цел да се испорача иновативно, корисничко ориентирано искуство.

Во сржта на платформата за е-трговија е онлајн продавницата која е погодна за корисниците, истата вклучува суштински карактеристики како што се безбедна обработка на плаќањата, интуитивна навигација и следење на нарачките во реално време. Подобрувањето на искуството на клиентите е интелигентен чет-бот за препораки, напојуван со OpenAI API. Овој чет-бот користи напредна обработка на природен јазик за да обезбеди персонализирани предлози за производи и насоки врз основа на преференциите и интеракциите на корисниците. Интеграцијата на оваа функција покажува како вештачката интелигенција може да се користи за да се зголеми задоволството на клиентите и да се поттикне продажбата.

Административната страна на платформата се фокусира на ефикасноста и организацијата, адресирање на оперативните предизвици со кои се соочуваат бизнисите при управувањето со нивните онлајн продавници. Оваа компонента обезбедува алатки за ракување со нарачки, генерирање фактури и управување со кориснички податоци. Администраторите можат да гледаат детални списоци на нарачки, да извезуваат податоци за нарачки за надворешна употреба и прикачување на кориснички информации во системот. Овие способности им овозможуваат на бизнисите да одржуваат добро структурирана задница, да ги оптимизираат нивните работни текови и да донесуваат одлуки на база на постоечките податоци. Комбинацијата на динамичен интерфејс за клиентите и робустен административен систем гарантира дека апликацијата ги поддржува сите аспекти на операциите на е-трговија.

## Краток опис на апликацијата

Практичноста, пристапноста и ефикасноста на онлајн купувањето ги направија платформите за е-трговија неопходни алатки за бизнисите кои сакаат да го прошират својот дофат и да го подобрат задоволството на клиентите.

Оваа дипломска работа го истражува развојот и имплементацијата на апликација за е-трговија дизајнирана да обезбеди сеопфатна организација на производи и да го подобри корисничкото искуство преку иновативни карактеристики. Апликацијата претставува робустен систем кој комбинира корисничко искуство за онлајн купување со напредни алатки за управување за администраторите.

Компонентата на е-продавница го олеснува беспрекорното купување, нудејќи функционалности како што се прелистување производи, безбедна обработка на плаќање и следење на купувањето во реално време. Исклучителна карактеристика е интеграцијата на четбот за препораки, базиран на OpenAI API, кој на корисниците им обезбедува персонализирани предлози за производи врз основа на нивните преференции и интеракции, подобрувајќи го искуството на клиентите.

Од административна страна, апликацијата вклучува алатки за ефикасно управување со нарачките и ракување со податоците од клиентите. Администраторите можат да гледаат детални списоци на нарачки, да генерираат фактури и да извезуваат податоци за понатамошна анализа. Дополнително, платформата поддржува прикачување на кориснички информации и управување со сеопфатна база на податоци за корисници, овозможувајќи насочен маркетинг и подобрени односи со клиентите.

Целта на оваа теза е да го прикаже дизајнот и имплементацијата на оваа апликација за е-продавница, со детали за употребените технологии и методологии. Оваа работа покажува како современите алатки и API може да се искористат за да се создаде динамична, интерактивна и ефикасна платформа за е-трговија која ги задоволува потребите и на крајните корисници и на деловните администратори.

## Користени технологии

При развојот на мојата апликација, користев Spring Boot за backend делот, React за frontend делот и PostgreSQL за базата на податоци како што е прикажано на Слика 1.

## Java spring boot

Java Spring Boot е рамка со отворен код дизајнирана да го поедностави развојот на веб-апликации базирани на програмскиот јазик. Со помош на рамката Spring Boot, може полесно да се обрне внимание на градење на повеќе функции, а помалку подесување на поставките, што всушност ја прави суштинска алатка за модерен развој на веб апликации. Рамката го следи принципот на конвенција над конфигурацијата, што значи дека доаѓа со разумни стандардни за брзо започнување и извршување на проектите. На пример, задачите како што се управувањето со зависности, поставувањето конфигурации на апликациите и иницијализирањето на опкружувањата за време на траење се во голема мера автоматизирани, намалувајќи го потребниот рачен напор. Оваа едноставност им овозможува на програмерите побрзо да градат и развиваат апликации.

Рамката доаѓа со низа моќни карактеристики што ја прават разноврсна. На пример, вклучува зависности од стартер, кои се однапред дефинирани пакети на библиотеки приспособени за специфични функционалности, како што се развој на веб или интеракција со бази на податоци. Исто така, има вградени сервери како Tomcat и Jetty, овозможувајќи им на апликациите да работат независно без да бараат надворешен сервер. Овој пристап го поедноставува процесот на распоредување и го прави развојот пофлексибилен.

Spring Boot поддржува флексибилни опции за распоредување, дозволувајќи им на апликациите да работат како самостојни JAR-датотеки или да бидат распоредени како WAR-датотеки во традиционалните сервери. Исто така, обезбедува одлична поддршка за градење безбедни апликации со интеграција на Spring Security, која ги поедноставува задачите како автентикација, авторизација и контрола на пристап заснована на улоги. За апликации управувани од бази на податоци, Spring Data JPA нуди апстракција над најчесто користените операции на бази на податоци, што ја олеснува интеракцијата со релационите бази на податоци.

Придобивките од користењето Spring Boot се прошируваат на неговата приспособливост и разноврсност. Тој е способен да поддржува како мали проекти така и големи системи на ниво на претпријатие, што го прави погоден за различни сценарија. Од градење на RESTful API и микросервиси до апликации на облак, Spring Boot се истакнува во обезбедувањето моќни и ефикасни решенија. Покрај тоа, неговата обемна заедница и сеопфатната

документација го прават погодно за почетници, а истовремено обезбедуваат напредни алатки за искусни програмери [1, 2, 3].

## JS – React

React е библиотека со отворен код JavaScript развиена од Facebook за градење кориснички интерфејси, првенствено користени за веб-апликации. Широко се користи за создавање брзи, динамични и интерактивни апликации од предниот дел со користење на архитектура базирана на компоненти. React се фокусира на ефикасно управување со корисничкиот интерфејс (UI) со ажурирање само на неопходните делови на DOM (Document Object Model), што му овозможува високи перформанси.

Сржта на концептот на React се повторливите компоненти кои претставуваат самостојни блокови и дефинираат како треба да се појавува и да се однесува одреден дел од интерфејсот. Оваа модуларност им овозможува на програмерите да создаваат сложени апликации со комбинирање на помали делови. React користи и виртуелен DOM, лесен приказ во меморијата на вистинскиот DOM, за да го оптимизира прикажувањето и да ги подобри перформансите на апликацијата.

Една од истакнатите карактеристики на React е неговата декларативна синтакса, која им овозможува на програмерите да го дефинираат интерфејсот врз основа на состојбата на апликацијата. Овој пристап го прави кодот попредвидлив и полесен за дебагирање. Еднонасочниот проток на податоци на React гарантира дека промените на податоците се шират на контролиран и предвидлив начин, што го поедноставува управувањето со самата апликација. За поголеми апликации, библиотеките за управување како Redux или Context API може да се користат за одржување на сложени текови на податоци.

React е многу фаворизиран поради неговата флексибилност, ефикасност и приспособливост. Без разлика дали станува збор за градење на едноставна веб страница или проект од голем обем, React ги обезбедува алатките и шемите потребни за создавање беспрекорно и динамично корисничко искуство. Активната заедница, обемните ресурси и робусноста дополнително ја зацврстуваат позицијата на оваа алатка како водечки избор за развој на frontend делот од апликациите [4, 5].

## PostgreSQL

PostgreSQL, честопати наречен Postgres, е моќен систем за управување со релациона база на податоци (RDBMS) со отворен код, познат по својата робусност, доверливост и напредни карактеристики. Првично развиен во 1980-тите, еволуира во една од најразновидните и најшироко користени бази на податоци за денешните апликации.

PostgreSQL е дизајниран за складирање и управување со структурирани податоци, следејќи го релациониот модел. Поддржува сложени прашања и врски со податоци користејќи SQL (structured query language - структуриран јазик за пребарување). Она што го издвојува PostgreSQL е неговата поддршка за напредни типови на податоци, како што се JSON/JSONB за полуструктурирани податоци и низи, што го прави погоден за различни случаи на употреба, вклучувајќи ги и оние кои бараат модели на хибридни бази на податоци.

Една од клучните предности на PostgreSQL е неговата екстензибилност. Програмерите можат да дефинираат сопствени типови на податоци, функции и оператори за да ја приспособат базата на податоци на специфичните потреби на апликациите.

PostgreSQL се истакнува во перформансите и приспособливоста. Може ефикасно да ракува со големи збирки на податоци и истовремени корисници, благодарение на неговата поддршка за напредни техники за индексирање и оптимизација на барањата.

Како меѓуплатформска база на податоци, PostgreSQL е компатибилен со различни оперативни системи, вклучувајќи Linux, macOS и Windows. Беспрекорно се интегрира со модерни рамки за апликации и јазици, како што се Java (преку JDBC), Python (преку Psycopg) и JavaScript (преку Node.js). Нејзината природа на отворен код гарантира дека е доверлива и високо приспособлива, со активна заедница која обезбедува континуирани ажурирања и поддршка [6].

## Open AI api

OpenAI е водечка организација за истражување и распоредување на вештачка интелигенција, основана во 2015 година со цел луѓето да имаат секојдневна корист од вештачката интелигенција (AGI - Artificial general intelligence). Како организација се познати по развивањето на врвни модели со вештачка интелигенција, вклучувајќи големи јазични модели како GPT (Generative Pre-training Transformer - Генеративен претходно обучен трансформер), кои се способни да разберат и генерираат текст сличен на човекот.



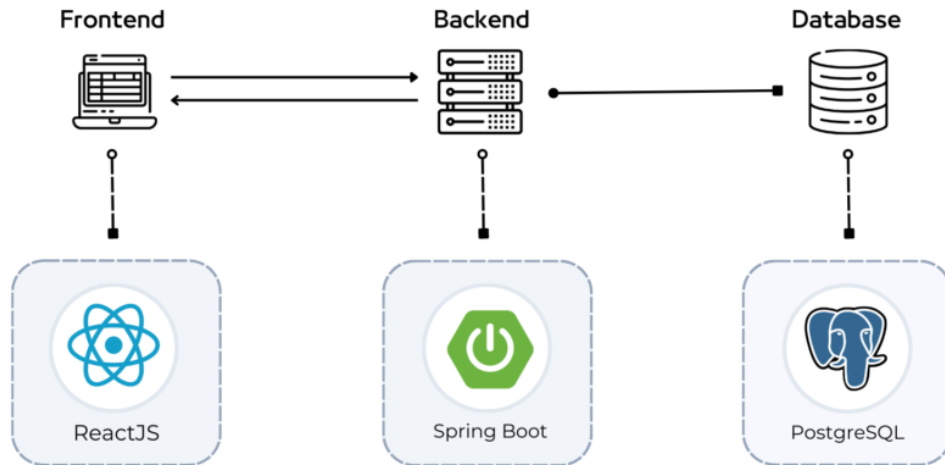
Основната технологија зад моделите на OpenAI е длабокото учење кое е подгрупа на машинско учење што користи вештачки невронски мрежи за обработка на огромни количини на податоци и учење сложени процеси. Овие модели се обучени за различни збирки на податоци, овозможувајќи им да извршуваат задачи како што се обработка на природни јазици (NLP - Natural language processing), препознавање слики, создавање содржина, па дури и помош за кодирање.

Технологијата на OpenAI е многу разновидна. Неговите модели можат да генерираат текст сличен на човекот. Можат да бидат во улога на чет-ботови, да резимираат големи документи, да преведуваат јазици и да обезбедат помош за креативно пишување. Покрај текстот, OpenAI се прошири и во други домени, како што е DALL·E за генерирање слики и Codex за програмирање и генерирање код.

Една од клучните предности на алатките на OpenAI е нивната пристапност. Програмерите и бизнисите можат да интегрираат сегменти и функционалности од вештачката интелигенција во нивните апликации преку API, овозможувајќи функции како што се интелигентна поддршка за корисници, напредна аналитика и автоматизација. Решенијата на OpenAI се скалабилни, опслужувајќи како мали проекти така и апликации на ниво на претпријатие.

OpenAI е исто така посветен на етичкиот развој на вештачката интелигенција. Вложува во истражување за да се справи со усогласувањето со вештачката интелигенција, осигурувајќи дека неговите модели се безбедни, непристрасни и усогласени со човечките вредности.

Организацијата активно соработува со академијата, индустријата и владите за да ја унапреди одговорната употреба на вештачката интелигенција и да ги ублажи потенцијалните ризици гарантирајќи посветен етички развој [7].



Слика 1: архитектурата на апликацијата со frontend ReactJS, backend Spring Boot и база на податоци PostgreSQL, покажувајќи ја комуникацијата на слоевите

## Архитектура на решението

Архитектурата на веб-апликацијата е поделена на три главни компоненти: frontend, backend и базата на податоци. Оваа структура ја следи шемата за дизајн на Model-View-Controller (MVC), која обезбедува јасно раздвојување на зависностите, подобрувајќи ја одржливоста и приспособливоста на апликацијата.

Frontend делот го претставува презентацискиот слој на апликацијата. Тој е одговорен за прикажување на кориснички интерфејси, управување со состојбата на апликацијата и ракување со корисничките интеракции. Изграден со помош на React, frontend-от комуницира со backend-от преку повици на API, разменува податоци и динамично одговара на дејствата на корисникот. Со управувањето со навигацијата и состојбата, предниот дел обезбедува беспрекорно и интерактивно корисничко искуство.

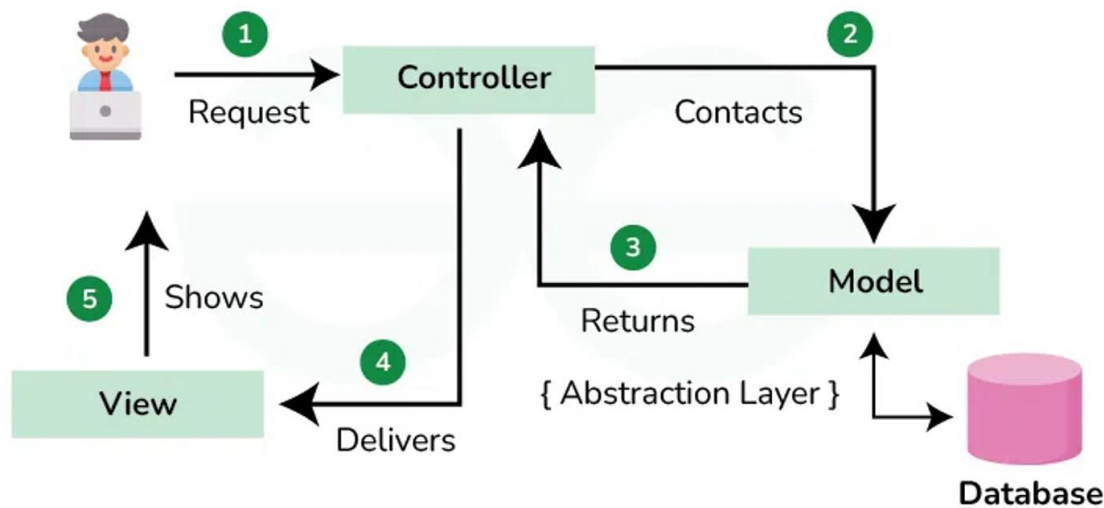
Backend делот служи како слој на апликација, обработувајќи ја деловната логика и делува како посредник помеѓу frontend-от и базата на податоци. Развиено со користење на Java Spring Boot, се справува со барањата на API од frontend-от, ги потврдува влезните податоци и ја извршува потребната деловна логика. Backend-от, исто така, обезбедува безбедност на податоците и управува со сложени операции. Таа е во интеракција со базата на податоци за да ги поврати, складира или ажурира податоците по потреба, обезбедувајќи му ги на frontend-от потребните информации за прикажување.

Базата на податоци го формира податочниот слој на архитектурата, складирајќи ги сите постојани информации што ги бара апликацијата. Користејќи го PostgreSQL, се управува со структурирани податоци како што се корисници, производи и нарачки. Базата на податоци е дизајнирана да обезбеди конзистентност на податоците, интегритет и ефикасно пребарување преку оптимизирана шема.

Оваа апликација се придржува до моделот MVC за да одржува чиста поделба на зависностите. Моделот ги претставува податоците и деловната логика на апликацијата, опфаќајќи ги ентитетите, складиштата и услугите во backend делот кои комуницираат со базата на податоци. Прегледот е управуван од frontend делот, кој ги прикажува корисничките интерфејси и ги прикажува податоците добиени од backend-от. Контролерот делува како посредник, обработувајќи HTTP барања и ги насочува до соодветните услуги за понатамошно ракување.

Работниот тек започнува со интеракција на корисникот со интерфејсот на frontend-от. Овие интеракции генерираат HTTP барања (GET, POST, PUT, DELETE) кои се испраќаат до backend-от преку RESTful API. Backend делот ги обработува барањата во своите контролери, извршувајќи ја потребната бизнис логика преку услугите, кои пак комуницираат со базата на податоци за да преземат или манипулираат со податоците. Базата на податоци ги извршува потребните прашања и ги враќа резултатите во backend делот. На крај, backend-от испраќа одговор назад до frontend-от, кој го ажурира корисничкиот интерфејс врз основа на примените податоци.

Следејќи ја оваа архитектура, апликацијата постигнува модуларност, приспособливост и леснотија на одржување, обезбедувајќи стабилно и корисничко искуство како што е прикажано на Слика 2 [8].



**MVC Design Pattern**



Слика 2: Овој дијаграм ја објаснува шемата MVC каде што контролерот се справува со барањата, комуницира со моделот за податоци и го праќа приказот кон корисникот.

## Имплементација на решението

### База на податоци

Дизајнот на базата на податоци е од суштинско значење за успешно функционирање на системот. Базата на податоци на NeuraCartix вклучува различни табели, како што се корисници, продукти, картичка и нарачки. Релациите помеѓу табелите се дефинирани преку надворешни клучеви кои обезбедуваат поврзаност помеѓу истите [6]. Базата на податоци се состои од следните табели:

- **users** (табела која се користи за чување на сите корисници на системот, во која се чуваат информации како име, презиме, е-пошта, хешираната лозинка, корисничко име и улогата на корисникот);
- **token** (табела која се користи за чување на токените за автентикација на корисниците. Се чува токенот, типот на токенот, дали е откажан, дали е истечен и надворешен клуч до табелата users кој покажува на кој корисник припаѓа токенот);
- **product** (табела за чување на продуктите, во која се чува името на продуктот, описот, сликата за истиот, како и неговата цена);
- **shopping\_cart** (табела која е наменета да служи како кошничка за секој корисник, по што располага со надворешен клуч до табелата за корисници);

- **product\_in\_shopping\_cart** (меѓу табела која ги поврзува продуктите и кошничката, се чуваат надворешни клучеви кои покажуваат кон табелата за продукти и табелата за кошничка)
- **orders** (табела наменета за нарачките во која е поврзана со надворешен клуч со табелата за корисникот се со цел да се обележи која нарачка на кој корисник припаѓа)
- **product\_orders** (меѓу табела која ги поврзува продуктите и нарачките, се чуваат надворешни клучеви кои покажуваат кон табелата за продукти и табелата за кошничка, како и количината на секој од продуктите)

## Backend

### Models (модели)

Во мојата Java Spring Boot апликација, моделите за корисници, токен, производи, кошничка и нарачките ги претставуваат основните ентитети на системот. Тие се означени како ентитети бидејќи кореспондираат со табелите во базата на податоци и ја дефинираат структурата на податоците во апликацијата. Користејќи ја анотацијата `@Entity`, овие модели овозможуваат беспрекорна интеракција со базата на податоци, користејќи ја JPA за мапирање на објекти на релациони податоци. Овој пристап ги поедноставува CRUD операциите, наметнува јасен модел на доменот и ги поддржува врските OneToMany, ManyToOne, OneToOne и ManyToMany, подобрувајќи ја одржливоста и приспособливоста на кодот.

### JPA Repository (репозиториуми)

JPA репозиториумите на NeuraCartix ја овозможуваат интеракцијата со базата на податоци преку едноставен и декларативен пристап. Користејќи ги интерфејсите што наследуваат од `JpaRepository` или `CrudRepository`, репозиториумите ги обработуваат основните операции за податоците, како што се зачувување, ажурирање, бришење и пребарување, без потреба од пишување SQL-код. Покрај тоа, тие поддржуваат креирање прилагодени методи со користење на конвенции за именување и анотации како `@Query`. Овој пристап го одвојува слојот за пристап до податоци од бизнис-логиката, овозможувајќи поконзистентен и одржлив код [2].

## Services (сервиси)

Во оваа апликација, сервисниот слој ја обработува бизнис-логиката и обезбедува посредување помеѓу контролерите и ЈРА репозиториумите. Со користење на анотацијата `@Service`, овој слој осигурува дека апликациските правила и процесите се централизирани и лесно се управуваат. Сервисите ја апстрахираат сложеноста на трансакциите и овозможуваат повторна употреба на логиката во различни делови на апликацијата. Оваа архитектура ја подобрува модуларноста, тестабилноста и ги минимизира дуплирањата на кодот.

Сервисниот слој ја имплементира основната деловна логика за поддршка на непрекорна интеракција со предниот дел преку дефинираните крајни точки. „`AuthenticationService`“ се справува со регистрацијата и најавувањето на корисниците, обезбедувајќи безбедни и ефикасни процеси за автентикација на корисникот. „`OrderService`“ обезбедува функционалности за управување со нарачките, вклучително и експортирање на сите нарачки за надворешна употреба како и генерирање фактури за одредени нарачки. „`ProductService`“ поддржува сеопфатно управување со производи, овозможувајќи додавање, пронаоѓање, уредување, бришење производи како и интеракција со чет-ботот. „`ShoppingCartService`“ управува со операциите на количката, овозможувајќи им на корисниците да ги видат деталите за нивната кошничка и да додаваат производи во нивната кошничка, менувајќи ја количината на секој од нив како и нивно отстранување од самата кошничка. „`UserService`“ го олеснува управувањето со корисниците, вклучително преземање на сите кориснички податоци и увоз на масовни кориснички информации преку прикачувања на датотеки. Овие услуги се дизајнирани да се интегрираат беспрекорно со контролорите, обезбедувајќи ефикасно и модуларно функционирање на апликацијата.

## Controllers (контролери)

Контролерите на NeuraCartix го сочинуваат слојот за презентација и го управуваат корисничкиот влез. Со користење на анотациите `@RestController` или `@Controller`, тие ги примаат HTTP барањата, ги обработуваат преку сервисниот слој и враќаат соодветни одговори, во вид на објекти (DTO – data transfer objects), JSON или HTML. Контролерите се одговорни за рутирање на барањата преку анотации како `@GetMapping`, `@PostMapping`, `@PutMapping` и `@DeleteMapping`, овозможувајќи читлив и структуриран код. Тие

обезбедуваат јасен интерфејс за корисничката интеракција и промовираат добро одделување на одговорностите во апликацијата.

Во мојата backend апликација, ги имплементирав следните крајни точки во моите контролери за справување со различни функционалности. „AuthenticationController“ вклучува крајни точки како што се `/Auth/login` за најавување на корисникот и `/Auth/register` за регистрација на корисникот. „OrderController“ управува со крајните точки како `/order/getAllOrders` за да ги врати сите нарачки, `/order/createInvoice/{id}` за генерирање фактури и `/order/exportAllOrders` за експортирање на сите нарачки. Во „ProductController“, крајните точки вклучуваат `/products` за да ги преземат сите производи, `/products/{productId}` за да добиете одреден производ според неговиот ID, `/products/add` за додавање нов производ, `/products/edit` за уредување на постоечки производи и `/products/delete/{id}` за бришење производ, исто така е овозможена и крајната точка за интеракција со чет-ботот чија патека е `/products/chat`. „ShoppingCartController“ обезбедува `/ShoppingCart/shopping-cart-info` за преземање на деталите за количката за најавениот корисник и `/Products/add-to-shopping` за додавање производи во количката. И на крај, „UserController“ вклучува `/User` за да ги преземе сите корисници и `/User/importUsers` за увоз на корисници преку прикачување датотеки. Овие крајни точки обезбедуваат беспрекорна интеракција помеѓу предниот и задниот слој на апликацијата [3].

### Stripe (сервис за наплата)

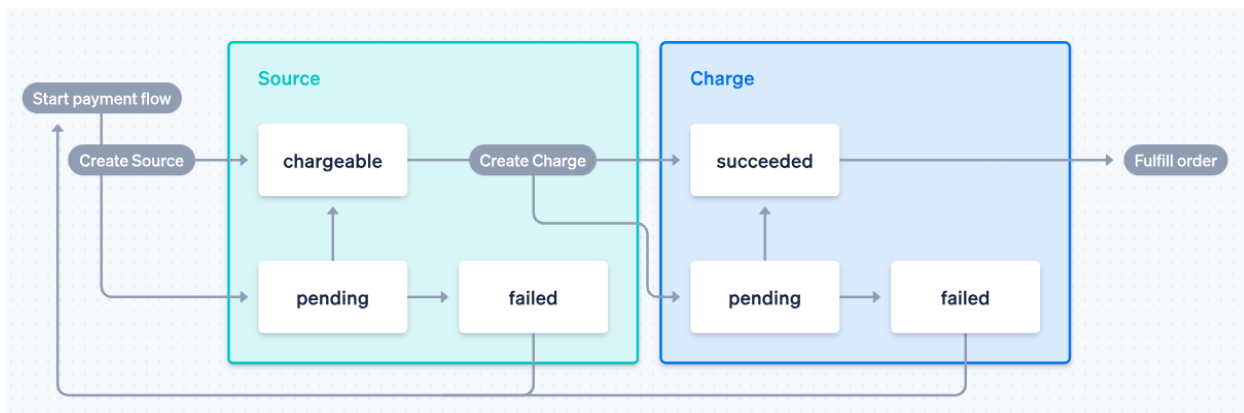
Stripe е популарна порта за плаќање што овозможува сигурна и ефикасна обработка на онлајн плаќање. Во оваа апликацијата, интеграцијата на Stripe вклучува неколку клучни чекори. Прво, развивачот регистрира сметка на платформата Stripe за да ги добие клучевите API, кои се од суштинско значење за автентикација на барањата до Stripe API. Овие клучеви обично се чуваат безбедно во датотеките за конфигурација на апликацијата. Јадрот на интеграцијата вклучува создавање на намера за плаќање на backend-от. Намерата за плаќање претставува сесија за плаќање, наведувајќи детали како износот, валутата и поддржаните начини на плаќање. Ова обезбедува сигурна обработка на плаќањето, правејќи го таинствено при испраќање кон frontend-от.

На страната на клиентот, библиотеката Stripe.js се користи за да се потврди плаќањето користејќи ја клиентската сесија обезбедена од backend-от. Овој пристап осигурува дека

чувствителните информации за плаќање, како што се деталите за картичката, се ракуваат безбедно од инфраструктурата на Stripe, намалувајќи го товарот на усогласеност со строгите прописи за плаќање.

За да се заврши работниот тек, Stripe го известува backend-от за промените на статусот на плаќање преку веб-куки (hooks). Тоа се крајни точки на серверот конфигурирани да управуваат со настани како што се успешни плаќања, неуспешни обиди или рефундирање. Со обработка на овие настани на веб-кука (hooks), апликацијата може да ги ажурира своите записи и да ја активира соодветната деловна логика.

Интеграцијата ја нагласува безбедната комуникација помеѓу клиентот, серверот и API на Stripe, со HTTPS како основен услов. Дополнително, клучевите API и чувствителните конфигурации мора да се складираат безбедно, често во променливи на околината. Овој дизајн гарантира дека процесот на плаќање останува сигурен, ефикасен и усогласен со индустриските стандарди (Слика 3) [9].



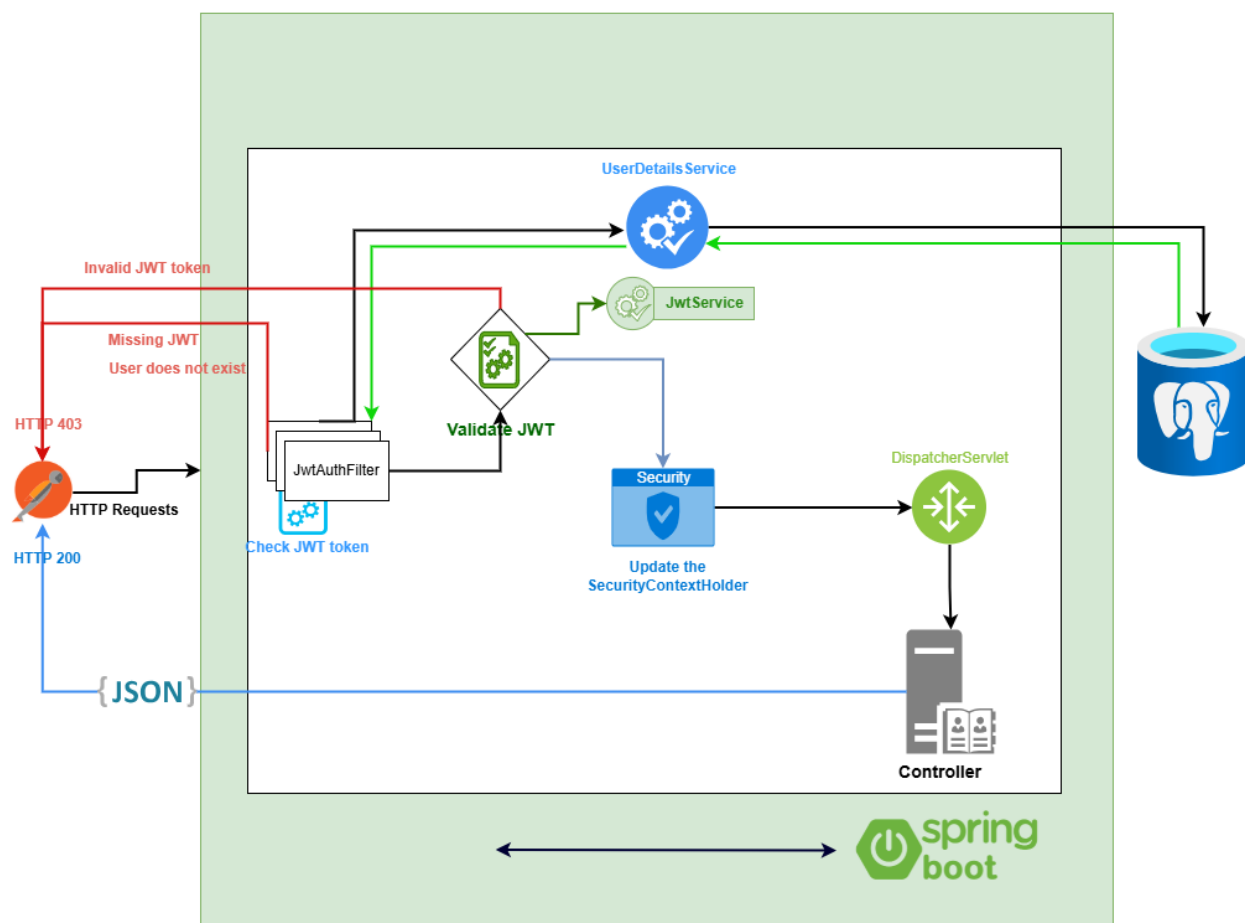
Слика 3: Процесот на плаќање направен со помош на stripe

### Authentication (автентикација и авторизација)

Автентикацијата за оваа апликација е направена со помош на JWT (JSON Web Token) кој обезбедува сигурна комуникација преку потврдување на барањата засновани на токен. Како што е илустрирано на дијаграмот (Слика 4), кога ќе се прими барање за HTTP, JwtAuthFilter го пресретнува за да провери дали има JWT. Ако токенот недостасува или е неважечки, филтерот одговара со HTTP 403 (Забрането). Ако е присутен валиден JWT, JwtAuthFilter го потврдува токенот со помош на JwtService, кој го декодира и го потврдува неговиот интегритет. Токенот потоа се користи за да се повлечат корисничките податоци од Услугата



за детали за корисникот, потврдувајќи го идентитетот на корисникот. По успешната валидација, SecurityContextHolder се ажурира со деталите на автентифицираниот корисник, дозволувајќи му на DispatcherServlet да го насочи барањето до соодветниот контролер. Контролерот потоа го обработува барањето и комуницира со базата на податоци доколку е потребно, враќајќи соодветен JSON одговор. Овој механизам гарантира дека само автентифицираните корисници можат да пристапат до безбедни крајни точки, со што се подобрува безбедноста на апликацијата [10].



Слика 4: Овој дијаграм го прикажува процесот на JWT автентикација на Spring Boot, потврдување на токени и ажурирање на безбедносниот контекст пред да се справи со HTTP барањата.

## Frontend

Frontend делот на апликацијата NeuraCartix е изграден со помош на React, со добро организирана и модуларна структура за да се обезбеди одржување и приспособливост. Главниот директориум на изворниот код е папката „src“, која е поделена на неколку подпапки, од кои секоја служи за одредена цел.

Папката „axios“ ја содржи централизираната конфигурациска датотека за Axios, наречена „axios.jsx“. Оваа датотека е одговорна за поставување на основната URL-адреса, сите пресретнувачи и заглавија, обезбедувајќи конзистентна API комуникација низ апликацијата.

Во папката „hoc“ е сместена датотеката „AuxWrapper.jsx“, која е компонента со повисок ред (НОС) дизајнирана да ги обвиткува детските компоненти. Ова овозможува управување со распоред за повеќекратна употреба и инкапсулација на заедничката логика споделена од повеќе компоненти.

Папката „pages“ е главниот директориум каде што се дефинирани сите страници на апликацијата, категоризирани по функционалност. Во подпапката „Auth“, има посебни компоненти за најавување и регистрација, именувани како „LoginPage.jsx“ и „RegisterPage.jsx“, соодветно, секоја придружена со своја CSS-датотека за стилизирање. Подпапката „Checkout“ ја содржи датотеката „CheckoutForm.jsx“, која се справува со процесот на наплата, заедно со неговите соодветни стилови. Подпапката „ForbiddenPage“ е посветена за прикажување на порака која означува одбиен пристап за неовластени корисници, имплементирана во „ForbiddenPage.jsx“. Подпапката „Home“ ја дефинира главната целна страница на апликацијата во „Home.jsx“. Слично на тоа, подпапката „Orders“ ја вклучува датотеката „Orders.jsx“ за прикажување на нарачките од корисниците. Заедничките компоненти како што се header-от и footer-от се наоѓаат во подпапката „Shared“, со нивната соодветна логика и стилови инкапсулирани во датотеките „Header.jsx“ и „Footer.jsx“. И на крај, подпапката „ShoppingCart“ ја содржи компонентата „ShoppingCartTerm.jsx“, која е одговорна за прикажување на поединечни ставки во количката.

Папката „Product“ содржи компоненти поврзани со управување и прикажување информации за производот. Подпапката „AddEditProductModal“ ги вклучува логиката и стиловите за додавање или уредување производи, додека подпапката „ProductsList“ е

одговорна за прикажување листа на производи. Компонентата „ProductTerm“, сместена во сопствената потпапка, прикажува детални информации за поединечни производи. Дополнително, во овој директориум има компонента „ChatBox“, која служи како чет-бот наменет за давање на препораки на корисниците, соодветни на нивните барања и преференци.

Папката „Users“ вклучува компоненти за управување со корисници, како што е датотеката „Users.jsx“, која се справува со функционалноста поврзана со корисникот. Постои и компонента „ImportUsersModal“ наменета за прикачување на кориснички податоци.

Глобалното управување со состојбите се имплементира со помош на Redux, при што конфигурацијата е зачувана во папката „redux“. Датотеката „store.js“ ја поставува продавницата Redux, додека датотеката „userSlice.js“ ја дефинира состојбата и дејствата поврзани со корисникот, како што се автентикација и управување со профилот.

Папката „repository“ е посветена на управување со интеракциите на API. Секоја подпапка во неа одговара на одреден домен на апликацијата. На пример, „authenticationRepository“ управува со најавување, одјавување и валидација на токени; „ordersRepository“ се справува со повици на API за преземање и креирање нарачки, „productRepository“ е одговорен за сите операции поврзани со производите, „rolesRepository“ се занимава со податоци за контрола на пристап засновани на улогите, „shoppingCartRepository“ управува со барањата за API поврзани со количката за купување и „usersRepository“ се справува со повици поврзани со API, како што се преземање и ажурирање кориснички профили.

Оваа структура гарантира дека апликацијата е модуларна, со јасна поделба на грижите. Страниците и споделените компоненти се организирани на начин кој промовира повторна употреба, додека Redux ефикасно управува со глобалната состојба. Употребата на репозиториумите за интеракции на API дополнително ја подобрува одржливоста на логиката на API повиците во апликацијата [4, 5].

## Интеграција со Open AI

OpenAI претставува водечка компанија за истражување и распоредување фокусирана на развој на напредни технологии за вештачка интелигенција. Неговите водечки модели, како што е GPT (Generative Pre-trained Transformer), се дизајнирани за разбирање и генерирање

на природен јазик. Овие модели можат да извршуваат широк спектар на задачи, вклучувајќи одговарање на прашања, генерирање креативна содржина, сумирање на текст и многу повеќе, преку искористување на обемна претходна обука за различни збирки на податоци. OpenAI ги обезбедува своите модели преку API, овозможувајќи им на програмерите да ги интегрираат способностите за вештачка интелигенција во нивните апликации. API им овозможува на корисниците да испраќаат потсетници (внесување базирано на текст) и да примаат одговори врз основа на задачата што ја имаат [7].

Во мојата апликација, го интегрирав API на OpenAI за да препорачам производи за шминка врз основа на барањата на корисниците. Еве како ја имплементирав врска:

#### 1. Поставување на крајната точка и овластување

- Крајната точка на OpenAI API за остварување на конекција е <https://api.openai.com/v1/chat/completions>.
- Комуникацијата бара клуч API за овластување. Клучот е вклучен во заглавјето на авторизација како посебен токен.

#### 2. Креирање на услугата

Направив сервис со името ChatGPService за да се справи со API комуникацијата. Овој сервис испраќа кориснички влезни податоци и податоци за производите до OpenAI и го обработува одговорот генериран од вештачката интелигенција (Слика 5).

#### 3. Конструирање на барањето

Барањето вклучува:

- Заглавија: за одредување на типот на содржина (апликација/json) и овластување.
- Тело: Содржи модел (gpt-4o-mini), кориснички влез и структурирана порака што се справува со одговорот на вештачката интелигенција.

```

HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);
headers.setBearerAuth(API_KEY);

ObjectMapper mapper = new ObjectMapper();

Map<String, Object> requestBody = new HashMap<>();
requestBody.put("model", "gpt-4o-mini");
String userPrompt = String.format("""
    For the following list of makeup credits recommend at most 3 suitable for the user requirements and provide
    an explanation for your choice as if you are explaining to the user why the chosen products are suitable.
    Return the result according to this JSON schema {"$schema":"http://json-schema.org/draft-07/schema",
    "type":"object","properties":{"productIds":{"type":"array","items":{"type":"integer"},"description":
    "An array of product IDs represented as integers."},"explanation":{"type":"string","description":
    "A string providing an explanation."},"required":["productIds","explanation"],"additionalProperties":false}.
    Do not ``json and ``.

    Products:
    %s

    User Requirements:
    %s
    """, mapper.writeValueAsString(products), userInput);
requestBody.put("messages", new Object[]{
    Map.of( k1: "role", v1: "system", k2: "content", v2: "Act as an expert in recommendation of makeup products."),
    Map.of( k1: "role", v1: "user", k2: "content", userPrompt),
});

```

Слика 5: Дел од кодот од ChatGPTService конкретно прикажан делот со креирањето на requestBody

#### 4. Испраќање на Барањето

За испраќање на барањето е искористен Spring's RestTemplate како што е прикажано на Слика 6.

```

HttpEntity<Map<String, Object>> entity = new HttpEntity<>(requestBody, headers);
RestTemplate restTemplate = new RestTemplate();

```

Слика 6: Имплементација на RestTemplate

#### 5. Обработка на одговорот од моделот

Одговорот на Open AI API се анализира за да се извлечат релевантните податоци, како што се препорачаните ID (идентификациски броеви) на производите и објаснувањето. Го десеријализирав одговорот на вештачката интелигенција во прилагоден објект ChatDto за структурирано ракување (Слика 7).

#### 6. Справување со грешки

Услугата вклучува робусно справување со грешки за управување со исклучоци за време на комуникацијата со API, осигурувајќи дека апликацијата останува стабилна (Слика 7).

```

try {
    ResponseEntity<Map> response = restTemplate.exchange(API_URL, HttpMethod.POST, entity, Map.class);
    Map<String, Object> responseBody = response.getBody();

    if (responseBody != null) {
        Map<String, Object> firstChoice = ((List<Map<String, Object>>) responseBody.get("choices")).get(0);

        String content = (String) ((Map<String, Object>) firstChoice.get("message")).get("content");

        System.out.println(content);

        ChatDto chatDto = null;
        try {
            chatDto = mapper.readValue(content, ChatDto.class);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return chatDto;
    }
} catch (Exception e) {
    e.printStackTrace();
}

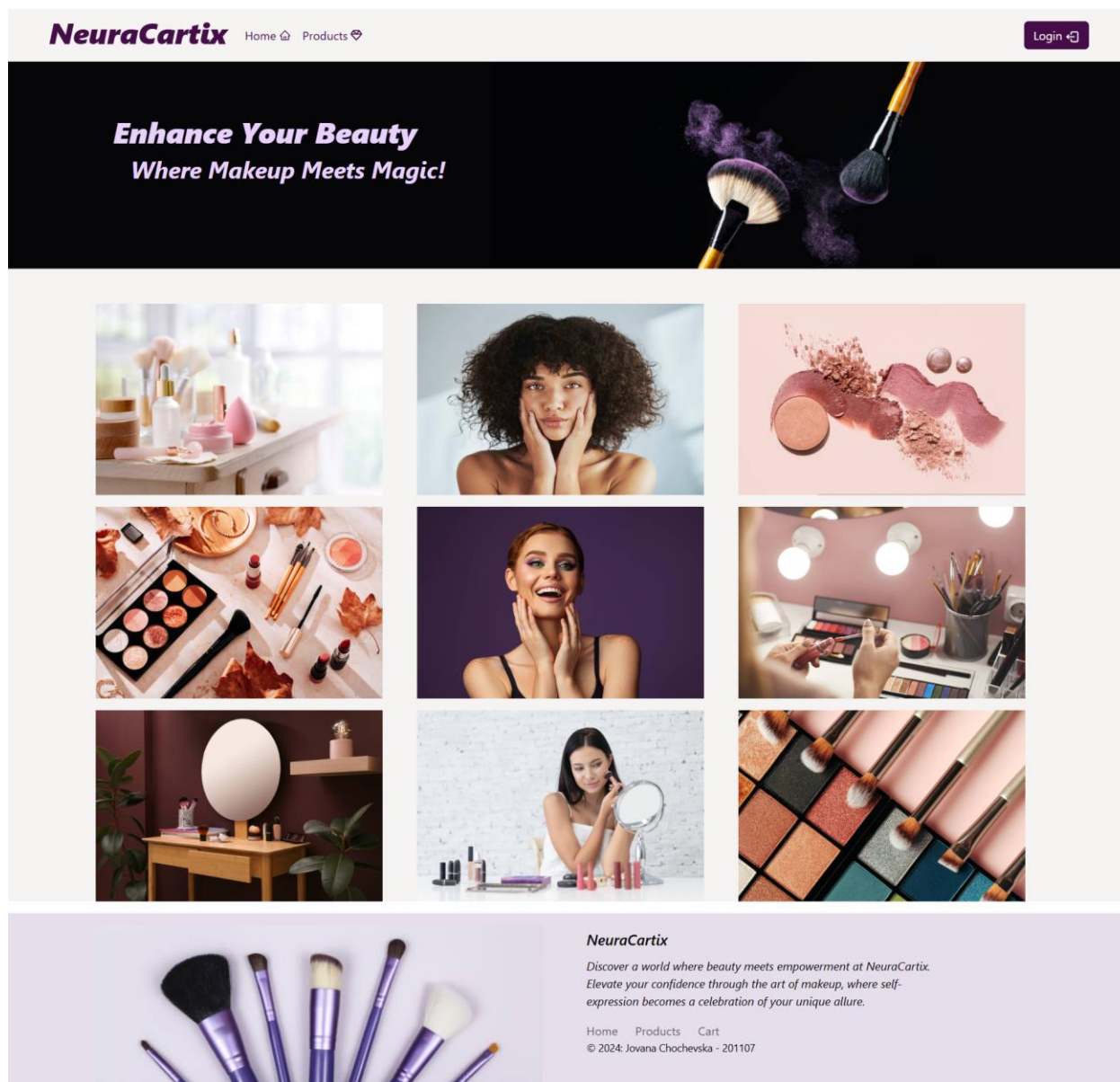
```

Слика 7: Овој фрагмент од кодот покажува како апликацијата испраќа барање до OpenAI API, го обработува одговорот и вклучува справување со грешки за управување со потенцијални исклучоци за време на комуникацијата со API.

Со интегрирање на OpenAI API мојата апликација обезбедува динамично интелигентни препораки, подобрувајќи го корисничкото искуство. Овој сервис го претвора барањето на корисникот во интелигентен персонализиран одговор кој содржи препорачани продукти и објаснување.

## Кориснички интерфејс

Кога корисникот за прв пат ќе пристапи до апликацијата тој е насочен кон почетната страна (Слика 8). Од оваа страна без да биде најавен тој има пристап само до каталогот на продукти, кој може само да го разгледува но не и да додаде некој продукт во кошничка.



Слика 8: Изглед на почетната страна („Home page“) на апликацијата

За да може корисникот да пристапи до својата картичка како и до опцијата за чет-бот за препораки тој мора да биде најавен. Најавата се прави при кликање на копчето „Login“ кое се наоѓа во навигацијата, по што се отвара екран за логирање, каде корисникот го внесува своето корисничко име и лозинка (Слика 9). Доколку корисникот нема направено свој профил тој може да кликне на копчето „Register“ кое ќе го пренасочи на страната за регистрација (Слика 10).

**NeuraCartix** Home Products Login

### Log in

Please login to your account

Username

Password

Log in

Don't have an account? [Register](#)

[Back to the main page](#)

Слика 9: Изглед на „Login“ страницата

**NeuraCartix** Home Products Login

### Register

Create a new account

First name Last name

Username Password

Email

Register

Already have an account? [Log in](#)


[Back to the main page](#)

Слика 10: Изглед на „Register“ страницата

Откако корисникот ќе се најави тој е пренасочен кон страницата за производи каде што овој пат му се овозможени делот со додавање на производи во кошничка како и полето за чет-бот за препораки (Слики 11, 12).




List of all products



**Scarlet Desire Lipstick**  
A bold red shade that commands attention with a creamy, long-lasting finish.

\$ 15.19


Add to cart



**Soft Touch Beauty Blender**  
A sponge that blends foundation effortlessly for a smooth, airbrushed look.

\$ 14.89


Add to cart



**Rosy Glow Blush**  
A soft, buildable formula that adds a natural flush and lifeness to your cheeks.

\$ 18.49


Add to cart



**Gleaming Eyeshadow**  
A versatile formula that glides on smoothly for bold, subtle or sparkly looks.

\$ 16.29


Add to cart



**Flawless Liquid Skin**  
Smooth and weightless foundation with medium to full coverage.

\$ 38.19


Add to cart



**Precision Define Lipliner**  
A creamy liner that shapes and enhances your lips with lasting color.

\$ 12.49


Add to cart



**Lash Amplify Mascara**  
Adds dramatic volume and curl to every lash for a striking effect.

\$ 26.59


Add to cart



**Velvet Touch Powder**  
A lightweight powder that evens out skin tone with a velvety texture.

\$ 22.79


Add to cart



**Precision Line Eyeliner**  
Smudge-proof and ultra-black for perfectly defined lines every time.

\$ 12.79


Add to cart



**Perfect Match Powder**  
Silky and buildable for natural-looking coverage that lasts all day.

\$ 22.49


Add to cart



**Pro Blend Beauty Blender**  
Precision-cut for versatile blending, from contouring to concealing.

\$ 13.99


Add to cart



**Luxe Lash Boost Mascara**  
Intensely pigmented for sky-high length and definition.

\$ 25.39


Add to cart



**Silky Blend Eyeshadow**  
Adds sparkle and dimension to your eyes with vibrant, glittery shades.

\$ 18.99


Add to cart



**Brighten Up Concealer**  
Lightweight yet full coverage for hiding imperfections and dark circles.

\$ 18.29


Add to cart



**Contour Pro Palette**  
Sculpt and define your features with this silky, blendable contouring kit.

\$ 36.49


Add to cart



**Luminous Glow Highlighter**  
A shimmering powder that enhances your features with a radiant glow.

\$ 26.79


Add to cart



**Rosy Bliss Lipstick**  
A cheerful pink hue that adds a pop of playful color to your smile.

\$ 16.39


Add to cart



**Radiant Glow Foundation**  
A hydrating formula that provides dewy, full coverage finish.

\$ 28.89


Add to cart



**Glow Maker Highlighter Brush**  
Designed for effortless application and blending of your favorite highlighters.

\$ 16.39


Add to cart



**Arch Define Brow Pencil**  
A dual-ended tool for shaping, defining, and filling in your brows with ease.

\$ 14.59


Add to cart



**Glossy Charm Nail Polish**  
A high-shine, chip-resistant formula for a salon-worthy manicure.

\$ 13.89


Add to cart



**Pro Finish Foundation Brush**  
A densely packed brush for seamless application of liquid foundations.

\$ 18.29


Add to cart



**Velvet Matte Eyeshadow**  
Soft, blendable shades for a flawless matte eye look.

\$ 15.69

Add to cart

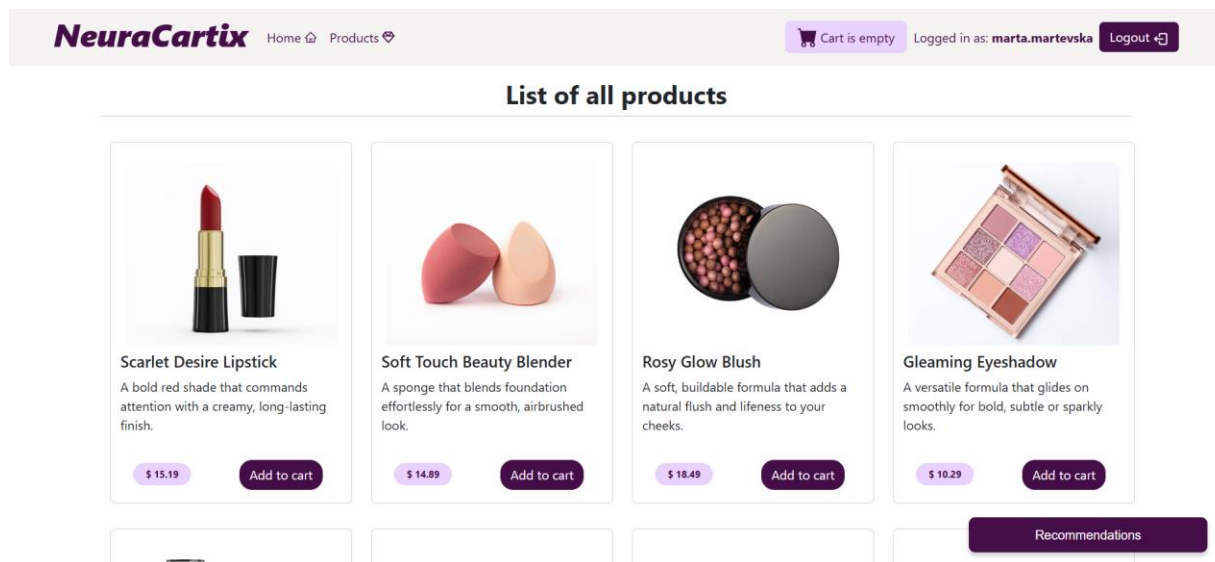


**Bold Flick Liquid Eyeliner**  
A waterproof formula that makes creating sharp wings a breeze.

\$ 18.49

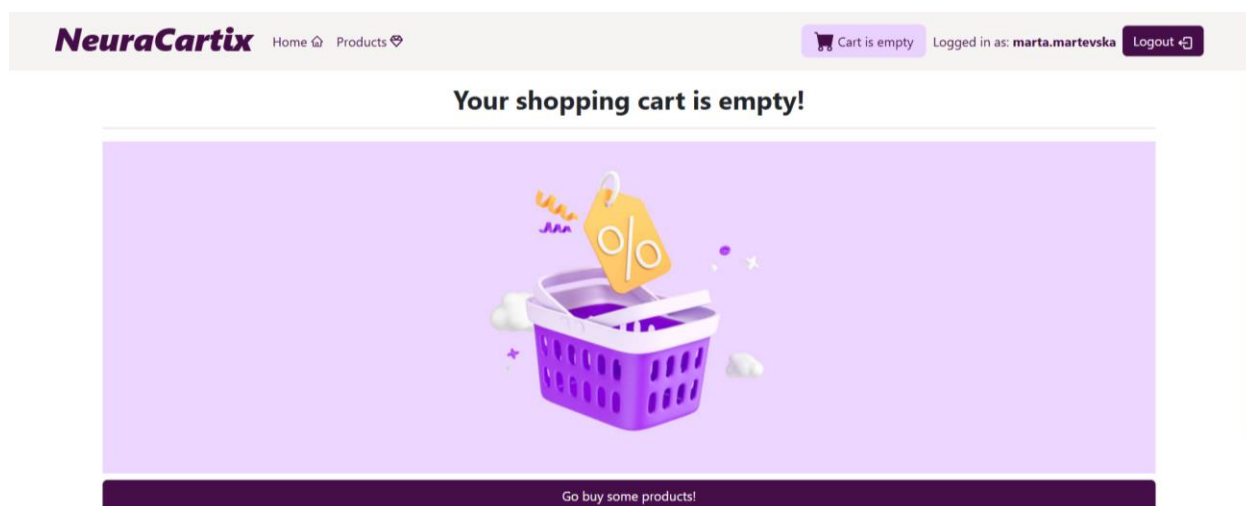
Add to cart

Слика 11: Целокупниот зглед на каталогот со производи кога корисникот е во улога на „user“

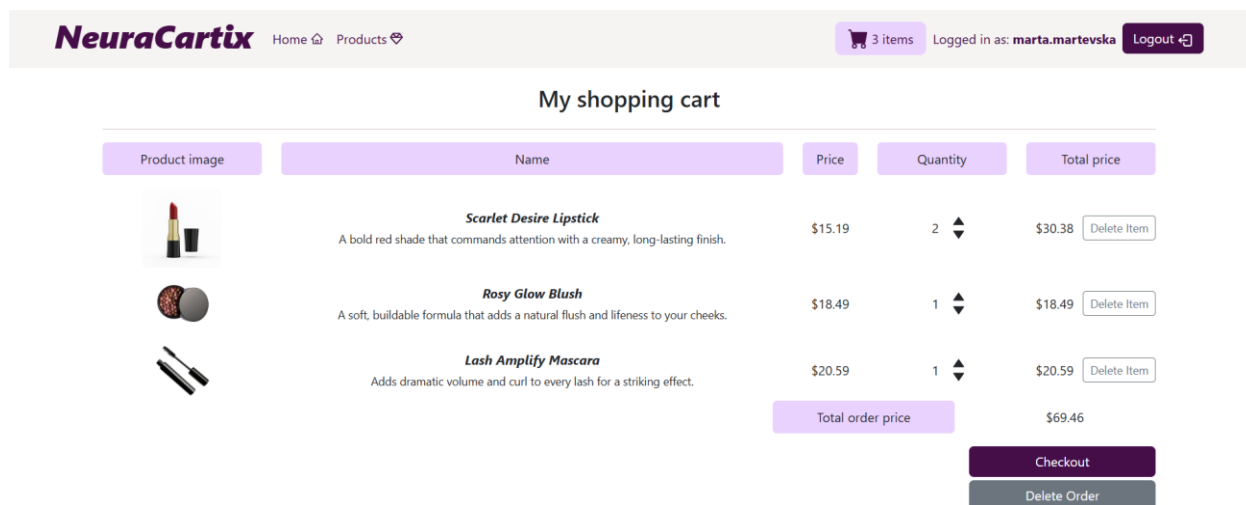


Слика 12: Изглед на каталогот со производи кога корисникот е во улога на „user“

Иницијално кошничката на корисникот е празна (Слика 13) и тој може соодветно по свој избор да ја пополни со производи од самиот каталог со кликање на копчето „Add to cart“. Откако корисникот ќе ги одбере посакуваните производи тој има можност да ја зголеми количината на секој од продуктите, да избрише даден продукт од кошничката или пак комплетно да ја избрише целата нарачка (Слика 14).

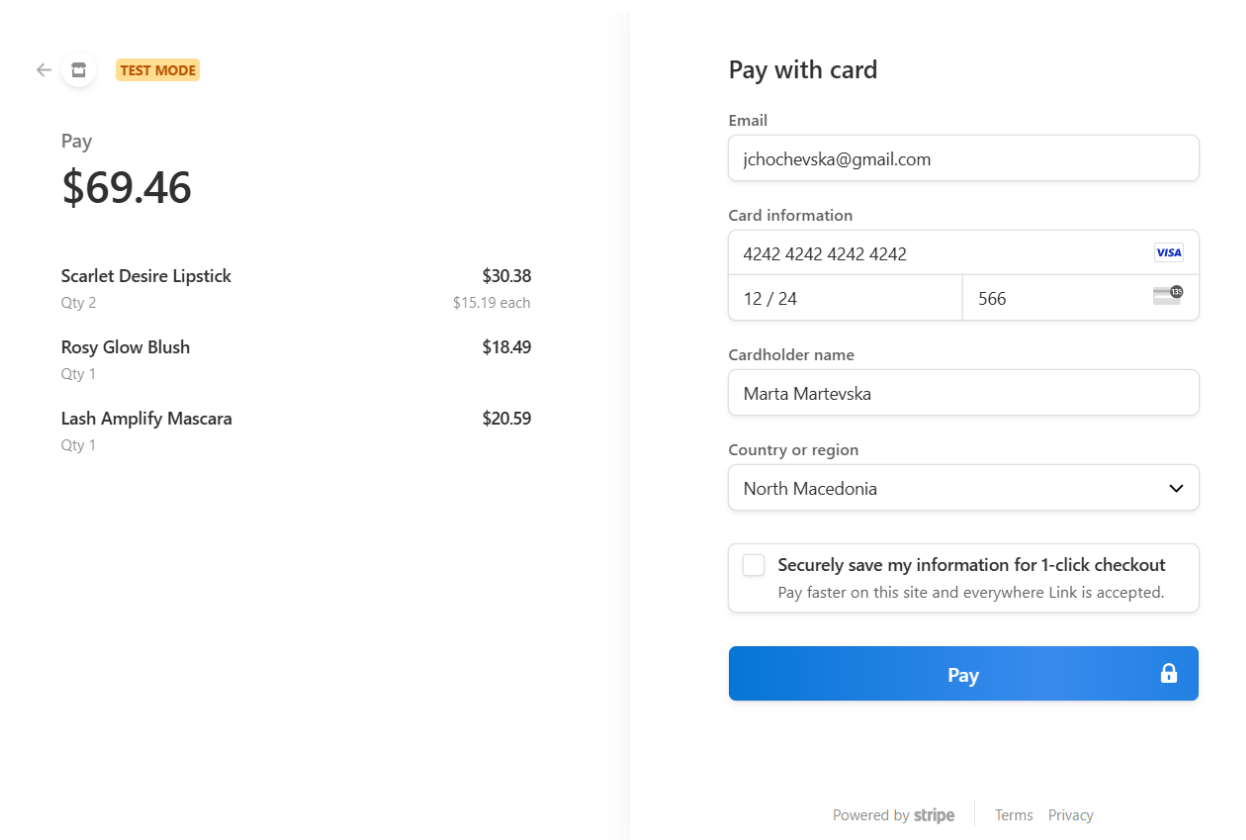


Слика 13: Изглед на кошничката за производи на корисникот кога не е пополнета со производи



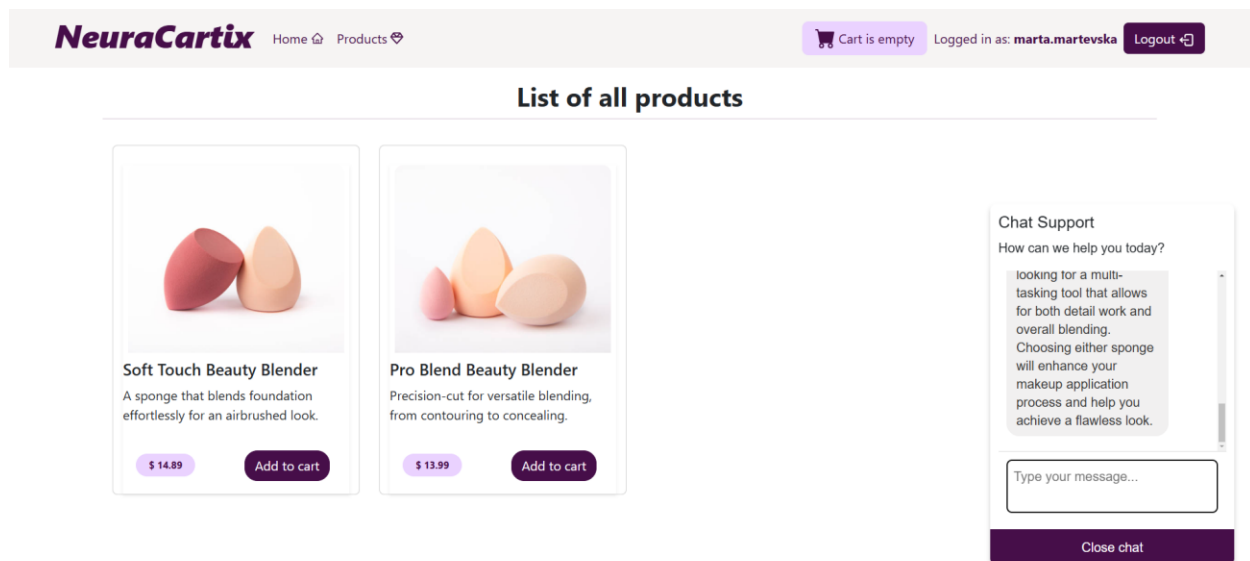
Слика 14: Изглед на кошничката за производи на корисникот кога е пополнета со производи и се овозможени сите соодветни функционалности

Кога истиот ќе заврши со креирање на својата нарачка притиска на копчето „Checkout“ кое го пренасочува на страна за плаќање каде ги внесува соодветните податоци (Слика 15).

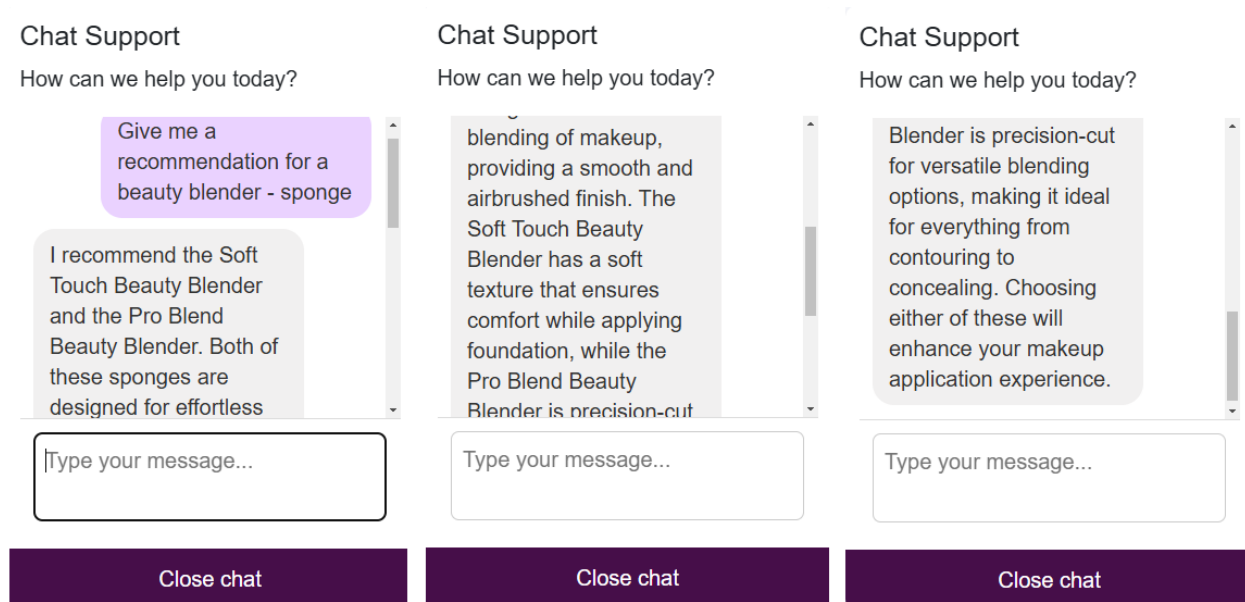


Слика 15: Изгледот на екранот на кој се пренасочува корисникот кога ќе сака да направи „checkout“

Освен функционалностите поврзани со картичката на корисникот му е достапен и чет-бот наменет за препораки. При кликање на копчето „Recommendations“ се отвара поле за чет каде корисникот поставува прашање во врска со продуктите и како одговор чет-ботот ги издвојува соодветните продукти и воедно му дава порака во која му објаснува зошто баш тие продукти се погодни и ги исполнуваат барањата на корисникот (Слики 16, 17).

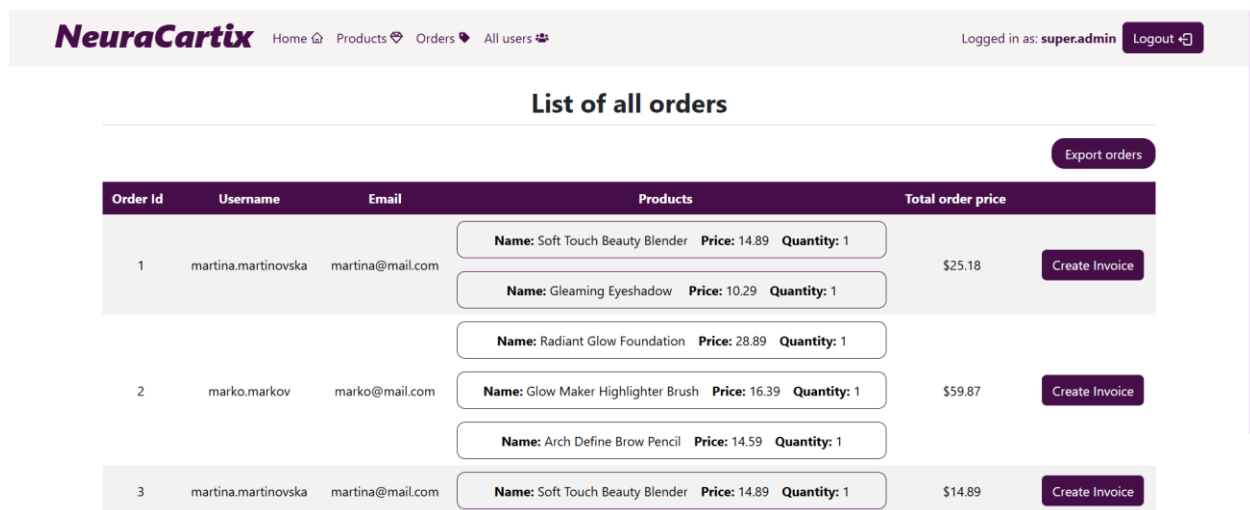


Слика 16: Изгледот на страницата за продукти откако ќе биде направено филтрирањето од страна на ботот



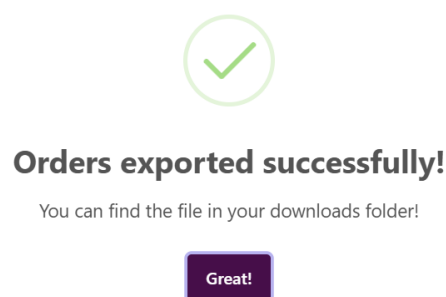
Слика 17: Деталниот одговор на чет-ботот поврзан со барањето на корисникот

Сето ова се однесува на корисникот во улога на „User“ но доколку се најавиме во улога на администратор („Admin“) тогаш ги немаме опциите за интеракција со чат-ботот како и можноста за додавање на продукти во картичка туку можеме да пристапиме до екраните кои располагаат со сите податоци за нарачките и корисниците (Слики 18, 23). Што се однесува до екранот за нарачки можеме да ја видиме табелата во која се прикажани сите направени нарачки за кои можеме да направиме експортирање во „Excel“ фајл (Слики 19, 20). Исто така можеме да направиме еден вид на фактура во „Word“ фајл која се однесува за секоја нарачка посебно (Слики 21, 22).



Order Id	Username	Email	Products	Total order price	
1	martina.martinovska	martina@mail.com	Name: Soft Touch Beauty Blender Price: 14.89 Quantity: 1	\$25.18	Create Invoice
			Name: Gleaming Eyeshadow Price: 10.29 Quantity: 1		
			Name: Radiant Glow Foundation Price: 28.89 Quantity: 1		
2	marko.markov	marko@mail.com	Name: Glow Maker Highlighter Brush Price: 16.39 Quantity: 1	\$59.87	Create Invoice
			Name: Arch Define Brow Pencil Price: 14.59 Quantity: 1		
3	martina.martinovska	martina@mail.com	Name: Soft Touch Beauty Blender Price: 14.89 Quantity: 1	\$14.89	Create Invoice

Слика 18: Изглед на екранот наменет за приказ на нарачките



Слика 19: Потврда за успешно експортирање на сите нарачки

	A	B	C	D	E	F	G	H	I	J
1	Order Id	Customer	Customer	Customer	Customer	Product-1	Product-2	Product-3		
2	1	Martina	Martinovs	martina.m	martina@	Soft Touch	Gleaming Eyeshadow			
3	2	Marko	Markov	marko.ma	marko@m	Radiant Gl	Glow Mak	Arch Define Brow Pencil		
4	3	Martina	Martinovs	martina.m	martina@	Soft Touch Beauty Blender				
5										
6										

Слика 20: Приказ на содржината на експортираните нарачки




## Invoice created successfully!

You can find the invoice in your downloads folder

Great!

Слика 21: Потврда за успешно креирање на фактура


**Neura Cartix**

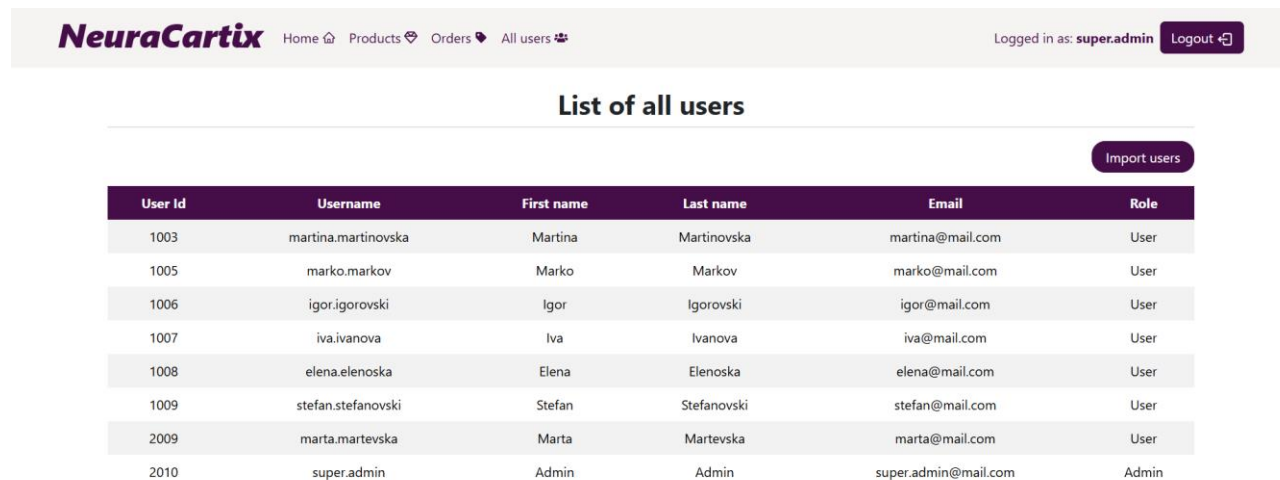
## Order Details

Order id:	2
Username:	marko.markov
Email:	marko@mail.com

Products in order	Radiant Glow Foundation with quantity of: 1 and price of : \$28.89 Glow Maker Highlighter Brush with quantity of: 1 and price of : \$16.39 Arch Define Brow Pencil with quantity of: 1 and price of : \$14.59
	<b>Total order price</b> <b>59.87</b>

Слика 22: Приказ на изгледот на генерираната фактура

Освен екранот кој служи за менаџирање на нарачките го имаме екранот на кој ни се прикажани податоците за сите корисници кои имаат направено свој профил (Слика 23). Тука имаме опција за додавање на корисници преку „Excel“ фајл во кој соодветно се внесуваат корисничкото име, името, презимето, електронската пошта и лозинката на корисникот (Слики 24, 25, 26). Оваа функционалност е корисна доколку се прави миграција од еден систем на друг или веќе имаме постоечки информации за нашите корисници.



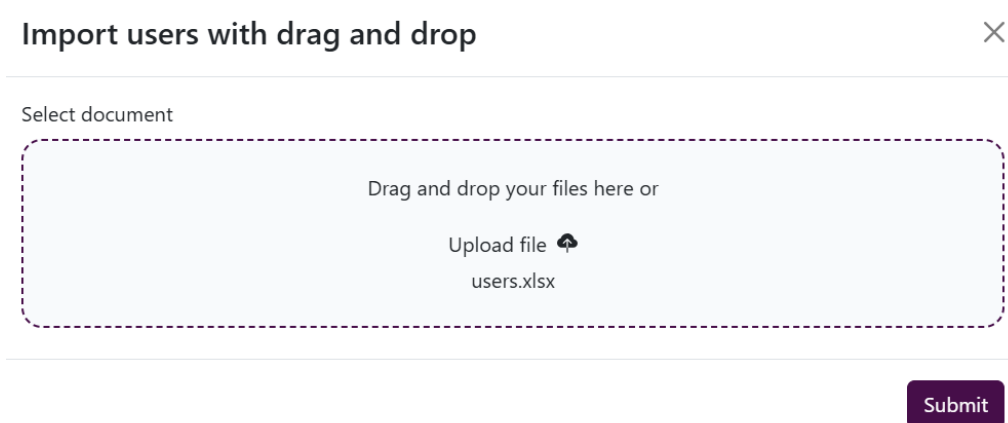
**NeuraCartix** Home Products Orders All users Logged in as: **super.admin** Logout

### List of all users

Import users

User Id	Username	First name	Last name	Email	Role
1003	martina.martinovska	Martina	Martinovska	martina@mail.com	User
1005	marko.markov	Marko	Markov	marko@mail.com	User
1006	igor.igorovski	Igor	Igorovski	igor@mail.com	User
1007	iva.ivanova	Iva	Ivanova	iva@mail.com	User
1008	elena.elenoska	Elena	Elenoska	elena@mail.com	User
1009	stefan.stefanovski	Stefan	Stefanovski	stefan@mail.com	User
2009	marta.martevska	Marta	Martevska	marta@mail.com	User
2010	super.admin	Admin	Admin	super.admin@mail.com	Admin


Слика 23: Изглед на екранот наменет за приказ на листата на корисници



### Import users with drag and drop

Select document

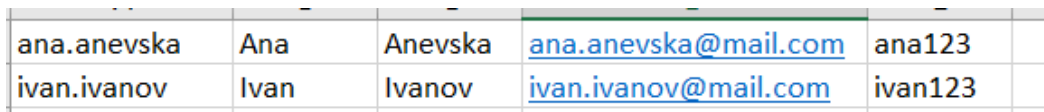
Drag and drop your files here or

Upload file 

users.xlsx

Submit

Слика 24: Приказ на модалниот прозорец каде се прикачува посакуваниот фајл со податоците за корисниците кои сакаме да ги додадеме



ana.anevska	Ana	Anevska	<a href="mailto:ana.anevska@mail.com">ana.anevska@mail.com</a>	ana123
ivan.ivanov	Ivan	Ivanov	<a href="mailto:ivan.ivanov@mail.com">ivan.ivanov@mail.com</a>	ivan123

Слика 25: Приказ на содржината на фајлот со корисници кој го прикачуваме

List of all users

Import users

User id	Username	First name	Last name	Email	Role
1003	martina.martinovska	Martina	Martinovska	martina@mail.com	User
1005	marko.markov	Marko	Markov	marko@mail.com	User
1006	igor.igorovski	Igor	Igorovski	igor@mail.com	User
1007	iva.ivanova	Iva	Ivanova	iva@mail.com	User
1008	elena.elenoska	Elena	Elenoska	elena@mail.com	User
1009	stefan.stefanovski	Stefan	Stefanovski	stefan@mail.com	User
2009	marta.martevska	Marta	Martevska	marta@mail.com	User
2010	super.admin	Admin	Admin	super.admin@mail.com	Admin
2011	ana.anevska	Ana	Anevska	ana.anevska@mail.com	User
2012	ivan.ivanov	Ivan	Ivanov	ivan.ivanov@mail.com	User

Слика 26: Изглед на екранот наменет за приказ на листата на корисници откако ќе направиме додавање на нови корисници




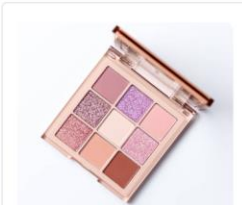
Освен тоа администраторот има опција да манипулира со производите. Односно да ги избрише, ажурира или пак да додава нови продукти (Слика 27). Бришењето се прави со кликање на копчето „Delete“. Ажурирањето се прави со кликање на копчето „Edit“ каде се отвара модален прозорец во кој сите полиња се пополнети со веќе постоечките информации за продуктот (Слика 28). За крај додавањето на продуктите се прави со кликање на копчето „Add new product“ каде се отвара истиот модален прозорец но овој пат со празни полиња кои ние треба да ги пополниме (Слика 29).

**NeuraCartix**
Home
Products
Orders
All users

Logged in as: **super.admin**
Logout

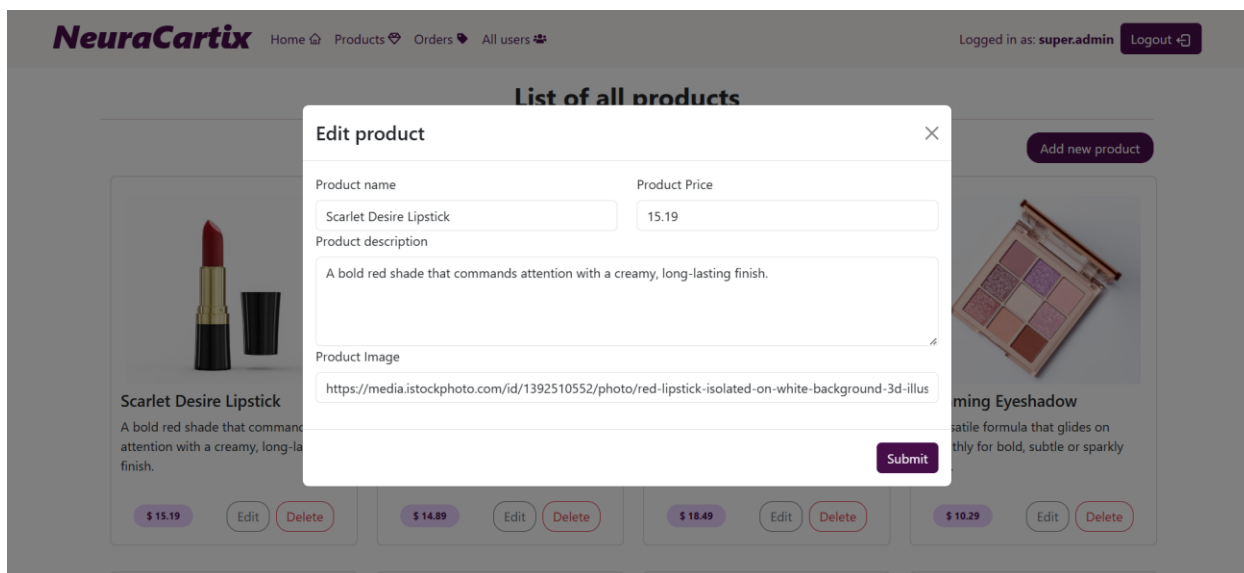
List of all products

Add new product

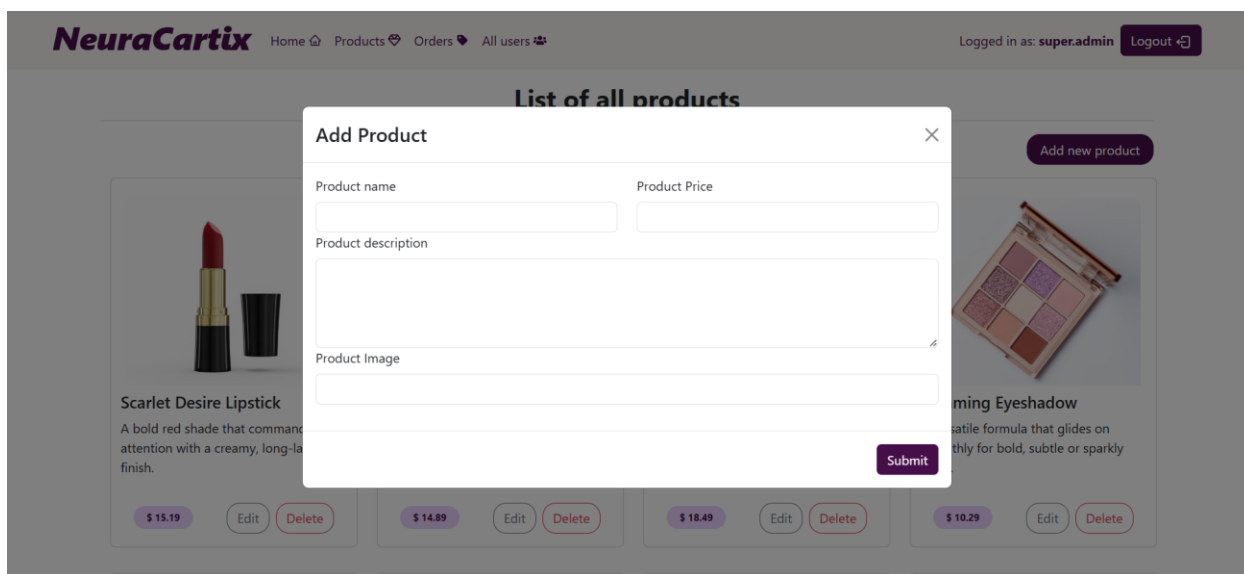
 <p><b>Scarlet Desire Lipstick</b></p> <p>A bold red shade that commands attention with a creamy, long-lasting finish.</p> <p>\$ 15.19</p> <p>Edit Delete</p>	 <p><b>Soft Touch Beauty Blender</b></p> <p>A sponge that blends foundation effortlessly for a smooth, airbrushed look.</p> <p>\$ 14.89</p> <p>Edit Delete</p>	 <p><b>Rosy Glow Blush</b></p> <p>A soft, buildable formula that adds a natural flush and lifeness to your cheeks.</p> <p>\$ 18.49</p> <p>Edit Delete</p>	 <p><b>Gleaming Eyeshadow</b></p> <p>A versatile formula that glides on smoothly for bold, subtle or sparkly looks.</p> <p>\$ 10.29</p> <p>Edit Delete</p>
--	---	---	---

Слика 27: Изглед на каталогот со продукти кога корисникот е во улога на „admin“





Слика 28: Приказ на изгледот на модалниот дијалог кој служи за ажурирање на веќе постоечки продукт



Слика 29: Приказ на изгледот на модалниот дијалог кој служи за додавање на нов продукт

## Подобрувања и нови функционалности

Развојот на оваа апликација за е-трговија поставува солидна основа, но има бројни можности да се прошири нејзината функционалност, да се подобрат нејзините перформанси и да се создаде уште попривлично корисничко искуство. Во иднина, еден од клучните приоритети е докеризирање на апликацијата. Со имплементирање на контејнеризација, процесот на распоредување ќе стане значително порационализиран, обезбедувајќи конзистентност во средини за развој, тестирање и производство. Овој чекор исто така ќе го

поедностави управувањето со зависностите и ќе го олесни скалирањето на апликацијата како што расте побарувачката на корисниците. Друго големо подобрување е транзицијата на апликацијата во архитектура базирана на микросервиси. Овој пристап ќе овозможи системот да се раздели на помали, независно распоредливи услуги, што ќе го олесни скалирањето на одредени компоненти по потреба, изолирањето на грешките и ажурирањето на функционалностите без да влијае на целиот систем. Таквата трансформација ќе ја подобри приспособливоста и флексибилноста на апликацијата, гарантирајќи за задоволувањето на потребите на растечката база на корисници. За понатамошна поддршка на растот на платформата, хостирањето на апликацијата на сигурна облак инфраструктура ќе биде значителен чекор. Ова ќе обезбеди висока достапност, побрзо време на вчитување и глобална пристапност.

Надвор од техничкиот напредок, подобрувањето на корисничкото искуство би било од голема корист. Ќе биде воведена нова функција за поврзување на апликацијата со упатства за шминкање, нудејќи уникатна понуда на вредностите. Оваа функција ќе обезбеди посветен дел каде што корисниците ќе можат да пристапат до видео содржини што покажуваат како се користат одредени производи, нивните придобивки и нивните намени. Со инкорпорирање на овие упатства, платформата не само што ќе го зајакне ангажманот на корисниците, туку ќе служи и како едукативна алатка, помагајќи им на клиентите да донесат поинформирани одлуки за купување. Овој персонализиран и интерактивен пристап има за цел да изгради доверба, да ја поттикне лојалноста и да ја подигне платформата надвор од едноставна страница за е-трговија до сеопфатен екосистем за купување и учење.

Со имплементирање на овие подобрувања и дополнителни функции, апликацијата е подготвена да стане робусна, скалабилна и корисничко-центрична платформа која се усогласува со динамичните потреби на модерната е-трговија и дава исклучителна вредност и на крајните корисници и на администраторите.

## Заклучок

Оваа апликација претставува сеопфатно решение кое го премостува јазот помеѓу беспрекорните искуства за купување преку Интернет и ефикасното административно управување. Со комбинирање на функции како безбедна обработка на плаќање, следење на нарачки, управување со корисници и генерирање фактури, платформата ги задоволува потребите и на крајните корисници и на администраторите. Интеграцијата на напредни технологии, како што е чет-ботот со препораки, напојуван со OpenAI API, го нагласува иновативниот пристап преземен за персонализирање на искуството за купување и подобрување на задоволството на корисниците.

Преку своите робусни функционалности и внимателен дизајн, апликацијата го демонстрира потенцијалот на современите технологии во преобликувањето на пејзажот на е-трговија. Покрај тоа, предложените идни подобрувања, вклучително и контејнеризација, транзиција кон микросервисна архитектура, хостирање и интегрирање на упатства за шминка, нудат јасен патоказ за континуиран раст и подобрување. Овие планирани достигнувања ќе обезбедат приспособливост, доверливост и збогатено корисничко искуство, позиционирајќи ја платформата како конкурентно и адаптивно решение во динамичниот свет на онлајн малопродажбата.

Овој проект покажува како добро дизајниран и напреден пристап може да создаде влијателна платформа за е-трговија која ги задоволува тековните барања останувајќи прилагодлива на идните предизвици и можности.

## Користена литература

- [1] <https://spring.io/web-applications>
- [2] <https://spring.io/projects/spring-data-jpa>
- [3] <https://spring.io/projects/spring-data-rest>
- [4] <https://react.dev/learn>
- [5] <https://react-bootstrap.netlify.app/docs/getting-started/introduction>
- [6] <https://www.postgresql.org/about/>
- [7] <https://www.baeldung.com/spring-boot-chatgpt-api-openai>
- [8] <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
- [9] <https://docs.stripe.com/>
- [10] <https://jwt.io/introduction>