Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

Навчально науковий інститут інформаційних технологій та робототехніки

(повна назва факультету)

Кафедра комп’ютерних та інформаційних технологій і систем

(повна назва кафедри)

**Пояснювальна записка**

**до дипломного проєкту (роботи)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_магістр\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(освітньо–кваліфікаційний рівень)

на тему \_\_\_\_\_\_\_Розробка інтелектуальної інформаційної системи продажу автозапчастин\_\_\_\_\_\_

Виконав: студент 6 курсу, групи 601–ТН

спеціальності

122 Комп’ютерні науки

(шифр і назва напряму)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Маслов А. В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_Янко А. С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Полтава – 2022 року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**МАГІСТЕРСЬКА РОБОТА**

**спеціальність 122 «Комп’ютерні науки»**

**на тему**

**«Розробка інтелектуальної інформаційної системи продажу автозапчастин»**

**Студента групи 601–ТН Маслова Антона Валерійовича**

Керівник роботи

кандидат технічних наук,

доцент Янко А.С.

Завідувач кафедри

Полтава – 2022

**РЕФЕРАТ**

Кваліфікаційна робота бакалавра: 83с., 22 рисунків, 3 додатки, 18 джерел.

**Об’єкт дослідження**: діяльність інтелектуальної інформаційної системи інтернет торгівлі.

**Мета роботи**: розроблення веб–ресурсу для Інтернет з інтелектуальним підбором автозапчастинам.

**Методи**: проектування та розробка бази даних для функціонування інтелектуальної інформаційної системи Інтернет торгівлі, програмна реалізація інтерфейсу користувача та розробка інтерфейсу для адміністрування веб–ресурсу.

**Ключові слова**: автоматизована інформаційна система, модуль, модель, база даних.

**ANNOTATION**

Qualification work of master's degree: 83 p., 22 drawings, 3 appendices, 18 sources.

**Object of study**: operation of the Internet trade information system with an intelligent information search system.

**The goal of the work**: development of a web resource for the Internet with an intelligent selection of auto parts.

**Methods**: design and development of a database for the functioning of the intelligent information system of Internet trade, software implementation of the user interface and development of the interface for the administration of the web resource.

**Keywords**: web resource, cross–browser, design.

ЗМІСТ

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИСВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ 5](#_Toc122009036)

[ВСТУП 6](#_Toc122009037)

[РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ МАГАЗИНУ АВТОЗАПЧАСТИН 8](#_Toc122009038)

[1.1 Опис предметної області 8](#_Toc122009039)

[1.2 Порівняння з аналогами 11](#_Toc122009040)

[1.3 Програмні рішення 15](#_Toc122009041)

[1.3.1 Масштабованість 15](#_Toc122009042)

[1.3.2 Масштабування веб–сайтів Python / Django 16](#_Toc122009043)

[1.3.3 Масштабування готовими рішеннями 16](#_Toc122009044)

[1.3.4 Продуктивність та оптимізація 17](#_Toc122009045)

[1.3.5 Продуктивність через фрейм ворки такі як Shopify та Magneto. 18](#_Toc122009046)

[1.3.6 Коли не слід розробляти інформаційну систему на Python 18](#_Toc122009047)

[1.3.7 Шаблони фреймворків 19](#_Toc122009048)

[1.3.8 Обмежений бюджет 19](#_Toc122009049)

[1.3.9 Підсумок 19](#_Toc122009050)

[1.3.10 SEO 20](#_Toc122009051)

[РОЗДІЛ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ 23](#_Toc122009052)

[2.1 Інструментальні засоби проектування 23](#_Toc122009053)

[2.2 Інструментальні засоби розробки 25](#_Toc122009054)

[2.2.1Мова програмування Python 25](#_Toc122009055)

[2.2.2 Веб–фреймворк Django 26](#_Toc122009056)

[2.2.3 Засоби розробки інтерфейсу 28](#_Toc122009057)

[2.2.4 Вибір СУБД 30](#_Toc122009058)

[2.3 Аналіз вимог на розробку і проектування 34](#_Toc122009059)

[2.3.1 Аналіз варіантів використання 34](#_Toc122009060)

[2.3.2 Проектування моделі бази даних 35](#_Toc122009061)

[2.3.3 Проектування користувальницького інтерфейсу 38](#_Toc122009062)

[РОЗДІЛ 3 РОЗРОБКА WEB–САЙТУ ДЛЯ ПРОДАЖУ АВТОЗАПЧАСТИН 41](#_Toc122009063)

[3.1 Принципи розробки 41](#_Toc122009064)

[3.2 Розробка моделі даних 44](#_Toc122009065)

[3.3 Структура сторінок веб–додатки 46](#_Toc122009066)

[3.4 Розробка користувальницького інтерфейсу 48](#_Toc122009067)

[3.5 Адміністративна панель 53](#_Toc122009068)

[РОЗДІЛ 4 ТЕСТУВАННЯ 58](#_Toc122009069)

[ВИСНОВОК 63](#_Toc122009070)

[CПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 64](#_Toc122009071)

[ДОДАТОК А РЕЗУЛЬТАТИ ТЕСТУВАННЯ 66](#_Toc122009072)

[ДОДАТОК Б ЛІСТИНГ КОДУ ДЛЯ РЕАЛІЗАЦІЇ ДОМАШНЬОЇ СТОРІНКИ 68](#_Toc122009073)

[ДОДАТОК В ФРАГМЕНТ ЛІСТИНГУ КОДУ ДЛЯ РЕАЛІЗАЦІЇ СТИЛЕЙ ПРОЕКТУ 78](#_Toc122009074)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИСВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

**ER–діаграма** – Entity–relationship діаграма.

**UML** – Unified Modeling Language.

**JS** – Java Script.

**БД** – база даних.

**ВР** – веб–ресурс.

**ІС** – інформаційна система.

**CMS** – Система управління контентом.

**СУБД** – Система управління базами даних.

**ООП** – об’єктно–орієнтоване програмування.

**SSMS** – SQL Server Management Studio.

**QA** – quality assurance тестування.

# ВСТУП

Процес розвитку інформаційного суспільства позитивно впливає на всі сторони життя країни, суспільства та особистості. Не є винятком і система освіти. Напрямок інформатизації освіти фігурує в усіх без винятку національних планах розвитку інформаційного суспільства. У сучасних ВНЗ інформатизація відбувається у двох основних взаємопов’язаних напрямках. Перший напрям – це комп’ютеризація навчального процесу, модернізація навчального процесу з використанням сучасних інформаційних технологій. Другий напрям – комп’ютеризація систем управління ВНЗ шляхом автоматизації бізнес–процесів сучасних ВНЗ. Ця орієнтація є важливою та актуальною, оскільки спрямована на підвищення якості навчального процесу та зниження матеріальних витрат на організацію роботи Університету в умовах переходу до масової вищої освіти та Болонського процесу. Він пов’язаний з розробленням і впровадженням спеціалізованих програмних платформ комп’ютеризації університетського менеджменту.

Він передбачає розробку та впровадження спеціальної програмної платформи для комп’ютеризації університетського управління. Головною метою розвитку та впровадження МІСУ є підвищення якості освітнього процесу. Ця мета досягається завдяки безперервному моніторингу параметрів якості, забезпеченню достовірності та швидкості отримання інформації про всі аспекти організації навчального процесу з необхідним рівнем деталізації, щоб забезпечити ефективність та ефективність управління безпосередній вплив на виконання університетом своїх освітніх функціональних рішень. Тема випускної роботи – розробка модуля «Електронний документообіг у підрозділах ВНЗ» для управління автоматизованою інформаційною системою управління навчальними процесами. Інструменти автоматизації управління навчальними процесами сьогодні є дуже популярним рішенням. Така система може істотно спростити виконання великої кількості внутрішніх процесів, тим самим підвищити продуктивність тих, хто в цих процесах задіяний, і підвищити ефективність роботи всього навчального закладу. Враховуючи складність і розмір таких систем, їх часто поділяють на програмні модулі, кожен з яких відповідає за певний аспект управління навчальним процесом. Під модулем ми маємо на увазі програмний блок, який реалізує одну або декілька взаємопов’язаних функцій. Такий підхід дозволив суттєво скоротити час впровадження системи, зосередившись на доставці самостійних функціональних модулів цільовому персоналу університету (працівникам деканату, професорсько–викладацького складу тощо). Враховуючи наведену ситуацію, ми будемо керуватися цим принципом при розробці системи створення програмного модуля «електронний документообіг у вищих навчальних закладах» буде орієнтований саме на автоматизацію процедур електронного документообігу, а адміністративний модуль – на заповнення та підписання документів, що будуть використовуватися в інших модулях системи.

# РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ІНФОРМАЦІЙНОЇ СИСТЕМИ ІНТЕРНЕТ МАГАЗИНУ АВТОЗАПЧАСТИН

## **1.1 Опис предметної області**

Інформаційна система Інформаційна система інтернет–магазину автозапчастин забезпечує продаж, автоматичний підбір автозапчастин, різноманітні способи сортування. Система призначена для покращення продажів автомобільної продукції в глобальній мережі. Інформаційна система повинна задовольняти потреби користувачів інформаційної системи, тобто аналізувати потреби клієнтів і здійснювати маркетинг для потенційних користувачів. Поширеність автомобілів серед людей збільшується з кожним днем, і актуальність створення цієї інформаційної системи залежить від постійного обслуговування автомобілів, які постійно ламаються. Різниця між європейськими країнами та Україною в технічному обслуговуванні автомобілів полягає в тому, що українські водії зацікавлені самостійно ремонтувати та обслуговувати свій транспорт, на відміну від європейців, які звикли звертатися до спеціалізованих СТО, де слюсарі професіонали займаються профілактикою ремонтом та підбором автозапчастин відповідно.

Інформаційна система інтернет–магазину автозапчастин повинна бути доступною і зрозумілою як для професійних СТО, так і для звичайних водіїв, яким необхідно самостійно ремонтувати свій автомобіль. Наявність термінів автозапчастин має бути різноманітною, оскільки досвідчені автомобілісти можуть не використовувати справжню технічну назву запчастини, оскільки під час автомобільного буму назви більшості деталей так чи інакше набули популярності та замінили оригінальні терміни. Оскільки основною метою розробки інформаційної системи є доступність, елементи каталогу продукції повинні мати кілька назв, щоб полегшити пошук продукції. Інформаційні системи повинні надавати можливість:

* Підбір товарів і подальша обробка замовлень за допомогою таблиці «Прайс» і каталогу товарів.

Інформаційна система має надавати такі можливості як:

* підбір товарів і подальша обробка замовлень за допомогою таблиці «Прайс» і каталогу товарів. Така можливість необхідна для функціональної частини інформаційної системи;
* сплачуйте товари через глобальну мережу за допомогою відомих платіжних систем. Від різноманітності платіжних систем залежить простота вибору найкращої платіжної системи для покупців;
* обробка замовлень в реальному часі. Така можливість має надати потенційним покупцям інформацію в режимі реального часу про наявність товару на складі та, залежно від характеристик товару та адреси доставки замовника, розрахувати термін доставки та її вартість;
* реєстрація користувачів інформаційної системи. Можливість реєстрації надає користувачам такі переваги, як особистий кабінет, де можна переглядати замовлення. Таким чином постачальники автозапчастин можуть цілеспрямовано розміщувати рекламу та відповідну інформацію або проводити дослідження ринку;
* аутентифікація та авторизація користувачів. Система повинна автентифікувати користувача за обліковим записом користувача;
* надати потенційним покупцям повну інформацію про автомобільний товар (назва, виробник, матеріал, розміри, маса деталей, інформація про марку автомобіля);
* наявність розширених пошукових систем. Пошук за категорією (бренд, виробник, ціна) і назвою, ввівши частину слова або все слово;
* надайте клієнтам всю інформацію, необхідну для здійснення покупок у вашому інтернет–магазині;
* надайте можливість потенційним покупцям поставити запитання про роботу вашого інтернет–магазину. Користувачі інформаційних систем повинні мати можливість задавати запитання, якщо вони незрозумілі під час використання веб–сайту. Онлайн–підтримка відповість на всі запитання потенційних покупців і проконсультує, якщо виникнуть проблеми;
* забезпечити швидку та зручну систему замовлення автомобільної продукції. Мати кілька різних способів оплати продукції. Під час оформлення замовлення товару, покупцю має надаватися вибір яким чином він бажає провести оплату. Це може бути оплата товару одразу через платіжні системи або ж оплата при отримані у відділені пошти;
* система знижок для постійних клієнтів. Особистий кабінет користувача повинен містити систему лояльності, пропонувати бонуси або знижки покупцям, які часто користуються інформаційною системою;
* наявність розділів з історією замовлень. У цьому розділі користувач може переглянути всі свої попередні замовлення або відстежити стан доставки поточного замовлення;
* можливість залишати відгуки про товар. Покупці повинні мати можливість поділитися своїм досвідом купівлі конкретних автозапчастин. Після публікації відгуку клієнт отримає зворотній зв'язок від співробітників інформаційної системи інтернет–магазину;
* є система відновлення пароля. Ця функція потрібна, якщо користувач не пам'ятає пароль облікового запису. Наявність цієї функції дозволяє відновити доступ до записів і всіх записаних в них даних;
* можливість підписки на новини інтернет–магазину. За згодою Користувача він може отримувати останні новини про даний товар у зручний для себе час в інформаційній системі інтернет–магазину. Його можна надіслати електронною поштою або SMS–повідомленням на номер телефону користувач;
* політика конфіденційності. Забезпечити конфіденційність персональних даних покупців та інформації про їх замовлення;
* можливість редагування інформації про товар. Адміністратор інтернет–магазину матиме право додавати, видаляти та редагувати властивості автозапчастин;
* доступ до інформації про покупців. Ці права будуть належати виключно менеджеру інтернет–магазину для створення рахунків або створення спеціальних пропозицій;
* вміння відповідати на запитання клієнтів. Менеджери мають доступ до живого чату, щоб вести діалог із проблемними користувачами інформаційної системи.

Вимоги до дизайну та інтерфейсу інформаційної системи мають бути сучасними. Проблема з іншими інформаційними системами автозапчастин полягає в тому, що вони спеціально орієнтовані на чоловіків. На це вказує дизайн і колірна палітра каталогу товарів, всюди синій колір. Сьогодні жінки – це повноцінні водії, які також доглядають та обслуговують свої автомобілі, тому тематика сайту має бути універсальною та привабливою як для жінок, так і для чоловіків.

## **1.2 Порівняння з аналогами**

Величезною популярністю користується інтернет–магазин автозапчастин, з розвитком інформаційних технологій йти на автобазар за аксесуарами вже не важливо, тому водії вважають за краще купувати аксесуари через Інтернет. Існує велика кількість інформаційних систем продажу запчастин, але кожна з них має свої плюси і мінуси, проаналізувавши які, можна створити найкращу інформаційну систему, яка відповідає всім потребам потенційного покупця.

**Інтернет–магазин «elit.ua»**

Авторитетний інтернет–магазин, в якому представлений широкий асортимент товарів від різних виробників. Цей інтернет–магазин має ряд переваг:

* у них великий вибір товарів, які надає компанія;
* на сайті компанії клієнти можуть замовити поштову посилку;
* детальний опис товару;
* компанія приймає різні форми оплати;
* готівкою, кредитними картками та електронними переказами;
* простота доступу до онлайн–аукціону.

До недоліків даної інформаційної системи треба віднести:

* дизайн не є естетично привабливим;
* обмежена адаптивність;
* перевантаження інформацією, яка не потрібна;
* маленький розмір шрифту.

Реалізація цієї інформаційної системи з використанням мови гіпертекстової розмітки HTML і каскадних таблиць стилів CSS і програмування на Java Script.

**інтернет–магазин «autohim.com.ua»**

Цей магазин має такі переваги:

* у них великий вибір автозапчастин , включаючи шини, акумулятори та інші автозапчастини;
* можливість порівнювати ціни різних компаній є однією з найважливіших переваг;
* є можливість консультації.

До недоліків можна віднести застарілий дизайн і відсутність адаптивності під смартфони.

Програмна реалізація цієї інформаційної системи використовує мову гіпертекстової розмітки HTML, каскадну таблицю стилів CSS і мову програмування Java Script.

**Інтернет–магазин «avtoskorost.com.ua»**

Інтернет–магазин, який може доставити до ваших дверей. Інтернет–магазин має такі переваги:

* велика кількість наданих автозапчастин;
* можливість замовлення поштовою посилкою;
* детальний опис товару;
* різні види платежів;
* доставка 24/7.

До недоліків сайту можна віднести:

* складність навігації в каталозі та результатах пошуку;
* відсутність можливості коментувати.

Інформаційна система виконана на базі системи WordPress CMS, що означає певні недоліки, такі як простий дизайн, іншими словами відсутність унікальності та функціональності.

Виходячи з перелічених недоліків, а також переваг існуючих аналогів, необхідно було реалізувати програмний засіб з такими властивостями:

* гуманізований дизайн;
* можливість обмеження функцій користувача відповідно до ролей;
* юзабіліті сайту;
* легка реєстрація;
* зручність навігації на сайті;
* простота управління адмінсайтом.

Сторінка з описом товару має бути найважливішою сторінкою у вашому інтернет–магазині. Тому вони повинні містити вичерпну інформацію, достатню для того, щоб розвіяти сумніви покупця щодо необхідності покупки: назва товару, повний і правдивий опис, якісні зображення, гарантія, короткий опис умов доставки, способу оплати та повернення, обміну. Залучений інтернет–магазин не наважувався згадувати про можливі повернення або обміни на веб–сайті, вважаючи, що така інформація збільшить кількість повернень і спонукатиме покупців до невигідних дій. Але на практиці все трохи інакше: зростає кількість продажів, а не повернень. Відомість магазину про обмін та повернення придбаного товару свідчить про довіру до своєї продукції.

Інтуїтивно все зрозуміло, але, як показує практика, багато хто забуває про них або вважає необов'язковими. Потрібна така інформація, як прямий міський номер телефону. Просто мати номер мобільного телефону недостатньо. Також необхідно вказати адресу офісу або пункту видачі, куди клієнти можуть прийти. На сторінці «Контакти» повинна бути карта сайту із зазначенням найближчих орієнтирів. Рекомендую розмістити інтерактивну карту з пошукової системи Google, створену за допомогою спеціального конструктора. Найпростіший спосіб – скористатися Google Maps Designer. Наявність такої інформації позитивно вплине на довіру користувачів і «довіру» пошукових систем.

Також не забувайте про реальну та актуальну інформацію про ціни та наявність товарів. Біля товару не повинна стояти ціна «0,00» або бути написано «Ціни зверніться до менеджера». Для кожного товару повинна бути встановлена ​​конкретна фактична ціна. Ви також повинні докладати зусиль, щоб інформація про наявність товару на сайті була точною.

Наявність позитивних відгуків на самому сайті та сторонніх джерелах сприятливо впливає на ставлення інтернет–магазину і часто стає вирішальним фактором при прийнятті рішення про покупку.

## **1.3 Програмні рішення**

Вам варто розробити веб–сайт електронної комерції з нуля чи скористатися наявним рішенням? Тривають дебати щодо найкращого підходу для бізнесу та того, який варіант є найбільш фінансовим. Є дійсні бали з обох сторін. Для одних компаній більш доцільним є готове виробництво, тоді як для інших розробка на замовлення є єдиним розумним варіантом. Що з моєю інформаційною системою? Щоб дізнатися, давайте почнемо з питання, яке я поставив собі, щоб знайти відповідь на це питання.

* + чи повинен мій веб–сайт обслуговувати тисячі чи навіть мільйони користувачів;
  + чи повинен мій веб–сайт витримувати великі навантаження та бути швидким;
  + чи вимагає мій проект унікальних можливостей.

Відповідь на всі ці питання – так, а це означає, що створювати інтернет–магазин на інших фреймворках не можна. Давайте створимо веб–сайт електронної комерції за допомогою Python. Щоб зрозуміти чому, давайте проаналізуємо розробку електронної комерції за допомогою Python на основі трьох важливих факторів: масштабованості, продуктивності та унікальних функцій.

### 1.3.1 Масштабованість. Коли справа доходить до платформ електронної комерції, важлива масштабованість. Роздрібні торговці хочуть залучити нових клієнтів і збільшити продажі. З ростом кількості клієнтів зростає і платформа.

### 1.3.2 Масштабування веб–сайтів Python / Django. За допомогою індивідуальної розробки ви можете масштабувати свій проект двома способами: вертикально або горизонтально.

Вертикальне розширення означає, що ми додаємо більше функцій до платформи шляхом оновлення інфраструктури проекту. Для вертикального масштабування платформи ми додали до платформи більше серверів баз даних, серверів кешу та веб–серверів. Django надає механізми для цього за замовчуванням.

Горизонтальне масштабування – це масштабування примірника Django. Іншими словами, ми додали більше машин. Горизонтальне масштабування необов’язкове; використовуйте, коли вертикального масштабування недостатньо.

Під час масштабування важливо зробити все правильно з самого початку. З самого початку потрібно аналізувати кількість користувачів. Роблячи це, ми можемо визначити, який метод масштабування підходить для програми інформаційної системи.

Вертикальне масштабування є найбільш рентабельним, оскільки є можливість заплатити Amazon Web Services (AWS) за додавання додаткових серверів і платити розробникам за підтримку цих серверів. Коли ви масштабуєте свою платформу, вам доведеться заплатити за сервери та фактичну розробку програмного забезпечення.

### 1.3.3 Масштабування готовими рішеннями. Розширення на платформах електронної комерції дещо відрізняються. Наприклад, за допомогою Shopify ви можете створити веб–сайт із будь–яким планом (Basic Shopify, Shopify, Advanced Shopify) і масштабувати свій магазин, змінюючи плани.



Рисунок 1.1 – Ціна за підписку Shopify на місяць

«Gastronom» має можливість перейти на Shopify Plus, масштабоване корпоративне рішення, оскільки платформа продовжує розвиватися. Вартість тарифного плану Enterprise починається приблизно з 2000 доларів на місяць, що означає, що такий проект коштує щонайменше 24 000 доларів на рік.

### 1.3.4 Продуктивність та оптимізація. За допомогою Django ви можете створити високопродуктивну інформаційну систему з нуля. Існує багато методів і інструментів Django, які можуть допомогти зробити ваш код ефективнішим, швидшим і використовувати менше системних ресурсів. Ці інструменти призначені для оптимізації кешування, продуктивності бази даних і продуктивності HTTP.

Крім того, Django також надає такі інструменти, як django–debug–toolbar, які можуть отримати уявлення про те, де код є неефективним, щоб зрозуміти, де оптимізувати.

Офіційна документація Django Software Foundation. Там ви знайдете огляд інструментів і методів Django для покращення продуктивності онлайн–магазину та мінімізації споживання пам’яті.

Всесвітньо відомі компанії роками використовують Python, довівши його продуктивність і масштабованість. Гарними прикладами є такі платформи електронної комерції, як Wish та Instacart. До некомерційних сайтів, які використовують Python/Django, належать Disqus, Instagram, Pinterest, YouTube, Mozilla, The Washington Post, Spotify, Eventbrite та Dropbox.

### 1.3.5 Продуктивність через фрейм ворки такі як Shopify та Magneto. Висока продуктивність не є їх сильною стороною. Основна причина – експансія сторонніх розробників. З фреймворками ймовірніше, що для додавання потрібної вам функції знадобляться сторонні розширення. Проблема в тому, що вони можуть займати багато місця, збільшуючи час завантаження та знижуючи продуктивність. Таким чином, чим більше на веб–сайті програм сторонніх розробників, тим гірше він працюватиме.

Проаналізуємо швидкість відгуку веб–сторінки. За даними Littledata, середня швидкість завантаження сторінки Magento становить 2,8 секунди, а Shopify – 2,6 секунди. Великі електронні магазини мають менший час завантаження. У середньому магазини Magento повністю завантажуються за 6 секунд, тоді як магазини Shopify займають 6,8 секунди.

### 1.3.6 Коли не слід розробляти інформаційну систему на Python. У деяких випадках використання індивідуальна розробка може бути не найкращим варіантом. Давайте розглянемо деякі випадки, коли інтернет–магазин можна розробити за допомогою платформи електронної комерції.

Якщо метою є створення магазину без фактичного кодування, єдиним варіантом може бути готова платформа. Однак це підходить лише для невеликих веб–сайтів із невеликою кількістю функцій і без налаштування.

За допомогою готового програмного забезпечення ви можете створити просту інформаційну систему без знання програмування, але вам потрібно буде закодувати будь–які розширення, розширені шаблони та налаштування. Візьмемо Shopify як приклад. Він дозволяє редагувати HTML і CSS, надаючи вам контроль над своїми проектами. Для цього потрібні принаймні базові навички HTML/CSS.

Більшість фреймворків надають необхідні функції для створення та керування веб–сайтами електронної комерції. Наприклад, у BigCommerce є система керування вмістом (CMS), функція оплати, керування замовленнями та запасами, виконання замовлень і доставка, знижки, керування поверненнями та інтеграція додатків. З OpenCart власники магазинів можуть насолоджуватися необмеженою кількістю продуктів і категорій, рейтингами та оглядами, мультивалютною та багатомовною підтримкою, інформаційною панеллю адміністратора, інтеграцією платіжного шлюзу, інтеграцією доставки та звітами про продажі.

### 1.3.7 Шаблони фреймворків. Готові рішення часто пропонують велику кількість шаблонів (як безкоштовних, так і платних), але обмежені можливості налаштування. Як правило, налаштування теми вимагає технічних навичок. Потрібне базове редагування макета без HTML/CSS.

### 1.3.8 Обмежений бюджет. Якщо гроші є проблемою, тоді ви можете використовувати фреймворки для створення інформаційних систем. Спочатку вам не потрібно багато вкладати, щоб створити платформу. Але кожного разу, коли вам потрібно додати функції, застосувати доповнення або налаштувати функції, з вас стягуватиметься плата.

Потрібно розуміти, що готові платформи не завжди коштують дешево. Візьмемо Shopify як приклад. Вам часто доводиться покладатися на програми сторонніх розробників, щоб додати функціональність, що означає доплату.

### 1.3.9 Підсумок. Python – це повністю власне рішення, тобто під час розробки веб–сайту можна розглянути все, від дизайну до функціональності. Використовуючи Python, ви можете створити велику та високопродуктивну програму. Розробка може бути величезною витратою бюджету, але це одноразова інвестиція такого розміру. Також будуть певні витрати, коли ви вирішите запровадити нові функції та внести зміни в майбутньому.

І навпаки, з розробником електронної комерції ви платите менше за налаштування свого веб–сайту, але потім продовжуєте платити за його використання.

Сьогодні готові платформи легко налаштовуються та містять обов’язкові функції для платформ електронної комерції. Однак, якщо виникне необхідність створити масштабовану платформу за допомогою готового рішення, ми, ймовірно, заплатимо більше, ніж розробимо її з нуля.

### 1.3.10 SEO. SEO (Search Engine Optimization) – це комплексна розробка та просування веб–сайту для отримання перших позицій у результатах пошукової системи (SERP) за вибраними запитами для збільшення трафіку та подальшого отримання прибутку.[5]

Незважаючи на те, що SEO має найвищу рентабельність інвестицій з усіх маркетингових зусиль електронної комерції, більшість онлайн–магазинів збирають разом мало уваги до пошукових систем.

З іншого боку, пошукова оптимізація вимагає лише попередніх зусиль – коли ви налаштуєте свій інтернет–магазин, інформаційна система може здійснювати продажі автоматично, без постійних витрат. Безкоштовний повторний трафік із високою конверсією. Саме цього потрібно досягти на завершальному етапі створення інтернет–магазину. Ключові тактики, які слід включити в стратегію SEO електронної комерції.

Безкоштовний повторний трафік із високою конверсією. Саме цього потрібно досягти на завершальному етапі створення інтернет–магазину.

Основні тактики, які слід включити в стратегію SEO електронної комерції

Найкращі стратегії SEO електронної комерції включають:

* + - дослідження ключових слів, щоб знайти типи ключових слів, за якими шукають ваші клієнти;
    - архітектура сайту на основі дослідження ключових слів;
    - seo на сторінці за допомогою мета–тегів і стратегічної оптимізації ключових слів у вмісті;
    - технічне оптимізацію пошукових систем, яке допомагає пошуковим системам ефективно сканувати сайт;
    - місцеве оптимізацію пошукових систем допомагає залучити місцевий органічний трафік (якщо у вас є звичайний магазин);
    - контент–маркетинг для залучення більшої кількості органічних відвідувачів;
    - побудова посилань, щоб підвищити авторитет вашого сайту.

Вимірюйте успіх SEO за допомогою таких інструментів, як Google Analytics і Ahrefs.

Пошукова оптимізація (SEO) – це мистецтво й наука оптимізації вашого веб–сайту за певними ключовими словами, щоб отримати кращий рейтинг у результатах пошуку, таких як Google.

Наука і мистецтво, тому що хоча багато відомо про технічні аспекти SEO, це стосується творчого досвіду користувача та дизайну. Але SEO–оптимізація зрештою полягає в одному: створенні найкращих результатів для ваших цільових ключових слів.

Мета Google – ранжувати результати пошуку, які відповідають на всі запитання шукача настільки добре, що їм не потрібно повертатися до Google за іншою відповіддю.

Забезпечте кращий досвід користувача завдяки швидшому веб–сайту, кращому мобільному досвіду, більш інтуїтивно зрозумілому інтерфейсу тощо.

Дослідження Outbrain показує, що пошук є рушієм номер один трафіку для контентних сайтів, перевершуючи соціальні мережі більш ніж на 300%.

Крім того, дослідження SEMuash 13 вертикалей електронної комерції виявило, що в 5 з них (музика, книги, меблі, дім і сад, електроніка) домінував звичайний пошук, причому на всі 13 звичайних і прямих пошуків припадало 80% усього трафіку.[\*]

У результатах пошуку Google покаже кілька оголошень, а потім звичайний список. Однак більшість кліків спрямовується на органічні результати. (Звичайно, це залежить від обсягу та ключових слів оголошення, але в більшості випадків це так.)

Оскільки приблизно 95% людей не пройдуть першу сторінку, рейтинг на першій сторінці є єдиним способом отримати реальні результати. (Знову ж таки, це буде різним, але здебільшого правда.)

Згідно з даними сервісу Ahrefs, інструменту, який аналізує запити веб–користувачів з усього світу, ключове слово "кінець кулі" отримує в середньому близько 11 000 пошукових запитів на місяць. Припустимо, що 35% кліків приходять до першого результату (середнє значення ключового слова), рейтинг №1 за цим ключовим словом отримає 3950 взаємодій.

Тепер припустімо, що коефіцієнт конверсії становить 10% (досить низький для ключового слова з високим наміром придбати). Перше місце за цим ключовим словом призведе до додаткових 385 продажів на місяць!

Це просто ключове слово. Більшість сторінок ранжуються за більш ніж одним ключовим словом, а більшість веб–сайтів ранжуються за більш ніж однією сторінкою.

Наявність добре налаштованої системи обміну повідомленнями може генерувати тисячі додаткових продажів на місяць лише з невеликими додатковими зусиллями щодо SEO.

# РОЗДІЛ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

## **2.1 Інструментальні засоби проектування**

Нотаційні діаграми UML і інструмент CASE Draw.io використовуються для розробки автоматизованих систем продажів.

Уніфікована мова моделювання UML – це мова загального призначення для графічного опису моделей, призначена для візуалізації, специфікації, проектування та документування компонентів програмного забезпечення, взаємодії системних об’єктів, бізнес–процесів та інших систем. UML необхідний для підтримки процесу моделювання програмного забезпечення з використанням об’єктно–орієнтованих методів, для організації концепцій і зв’язків між концепціями програмного забезпечення, а також для відображення проблем розширення складних систем. Модель графічної мови UML використовується на всіх етапах життєвого циклу програмного забезпечення, від бізнес–аналізу до підтримки системи. Різні організації застосовують UML на власний розсуд залежно від використовуваної технології або предметної області [2].

Базуючись на основних рисах об’єктно–орієнтованого аналізу та методів проектування, відносно повна модель складної системи представляє відповідну кількість взаємопов’язаних представлень, кожне з яких представляє аспект структури або поведінки системи. Ієрархія – це принцип, за яким моделі складних систем упорядковані, щоб забезпечити можливість процесу побудови моделей на різних рівнях деталізації або абстракції, не виходячи за межі фіксованого представлення. Загальновизнано, що мова складається з правил поєднання слів і словників для отримання значущих структур. Вона відрізняється тим, що мовні словники складаються з графічних елементів. Таким чином, кожна графічна нотація має власну відповідну семантику, так що модель, створена одним розробником, може бути чітко зрозуміла іншим розробником, а також програмними засобами, які інтерпретують UML. Словник мови UML складається з трьох типів блоків:

* суті – основні елементи моделі, так звані абстракції;
* відносини – частини, які з’єднують різні сутності;
* діаграми – блоки, представляють інтерес сутностей.

Діаграми UML додають підтримку для проектів, тим самим полегшуючи процес розробки документації для програмних систем. UML можна застосовувати до життєвого циклу аналізу бізнес–систем і розробки додатків [2].

Інструмент ERwin CASE використовується для розробки баз даних веб–додатків.

Інструмент CASE ERwin – це інструмент, який використовується для розробки структури бази даних. ERwin поєднує в собі графічний інтерфейс Windows і спеціальні засоби, необхідні для побудови ER–діаграм, також містить різні редактори для створення фізичних і логічних описів моделей даних і прозору підтримку провідних реляційних СУБД. ERwin дозволяє виконувати або створювати зворотне проектування баз даних.

Сімейство продуктів ERwin – це набір інструментів концептуального моделювання даних із використанням методології IDEF1X. Завдяки ERwin можливе проектування схем бази даних, опис і генерація мови цільової СУБД (DB2, Oracle, Microsoft SQL Server і т.д.) і зворотне проектування існуючих баз даних. Існує декілька конфігурацій ERwin, усі зосереджені на найпоширеніших інструментах розробки програм 4GL. Інтеграція з популярними інструментами для розробки клієнтів програм, дозволяючи автоматично генерувати код програми [3].

## **2.2 Інструментальні засоби розробки**

Інтернет–магазин використовує веб–фреймворк Django. Django – це безкоштовна відкрита платформа для веб–додатків, написана мовою програмування Python. Фреймворк підтримується Django Software Foundation. Розглянемо методики розробки докладніше.

### 2.2.1Мова програмування Python. Python – мова програмування, що відноситься до мов загального призначення високого рівня, яка спеціально розроблена для підвищення продуктивності розробників програмних продуктів, а також для читання коду. Лаконічний синтаксис є особливістю ядра Python. Проте стандартна бібліотека Python містить велику кількість корисних функцій [4].

Python є втіленням кількох парадигм програмування, з яких добре відомі: структурний підхід, функціонально–об’єктно–орієнтований, імперативний та аспектно–орієнтований. Код організований у функції та класи, які можуть бути об’єднані в модулі (які, у свою чергу, об’єднані в пакети) [4].

Процедурний підхід до об’єктно–орієнтованого та функціонального написання:

Переваги Python:

* відкритий розвиток;
* надавати методи швидкого прототипування та динамічної семантики;
* завдяки високо уніфікованому механізму імпорту та програмному інтерфейсу багато корисних бібліотек і мовних розширень можна легко використовувати у своїх власних проектах;
* модульна механіка добре проаналізована і може використовуватися послідовно;
* усе свідчить про те, що все в Python є об’єктом у сенсі ООП, але в той же час об’єктні методи не нав’язуються програмісту.

### 2.2.2 Веб–фреймворк Django. Django – це безкоштовний фреймворк веб–додатків Python, який використовує шаблон проектування MVC. Model–View–Controller (MVC, «Model–View–Controller») – це підхід, що використовує кілька шаблонів проектування, за допомогою яких модель програми, інтерфейс користувача та взаємодія з користувачем розділені на три окремі компоненти, модифікуючи один із компонентів таким чином. має мінімальний вплив на інші компоненти. Цей підхід проектування часто використовується для побудови архітектурної основи при переході від теорії до реалізації в конкретній предметній області [5].

Основною метою використання цієї концепції є відокремлення бізнес–логіки (моделі) від її візуалізації (подання, перегляду). Завдяки такому поділу збільшується можливість повторного використання.

Найбільш корисне застосування цієї концепції, коли користувачам потрібно переглядати ті самі дані одночасно в різних контекстах або з різних точок зору.

Зокрема, виконайте такі завдання:

* кілька типів можна приєднати до однієї моделі, не впливаючи на реалізацію моделі (наприклад, деякі дані можуть бути представлені одночасно у формі електронних таблиць, гістограм і кругових діаграм);
* не впливаючи на виконання перегляду, можна змінити реакцію на дії користувача (клацання мишкою по кнопці, введення даних), для цього достатньо використовувати інший контролер.

Деякі розробники спеціалізуються лише на одній із цих областей: або розробляють графічний інтерфейс, або розробляють бізнес–логіку (таким чином, можна усвідомити, що програміст, який працює над розробкою бізнес–логіки (моделі), не усвідомлює, що представлення в ньому буде використовуватися повністю).

Концепція MVC дозволяє вам розділити дані, презентацію та обробку дій користувача на три окремі компоненти. модель (англ. model).

Модель надає знання: дані та методи використання цих даних, відповідаючи на запити, змінюючи їх стан. Інформація про те, як візуалізувати ці знання, не включена.

Презентація, перегляд (View англ.). Відповідає за відображення (візуалізацію) інформації. Як правило, презентацією виступає форма (вікно) з графічними елементами. У Django шаблони визначені як презентації.

Контролер (англ. Controller). Забезпечує зв'язок між користувачем і системою: контролює дані, введені користувачем, і використовує моделі та представлення для досягнення бажаної реакції [5].

Фреймворк розділений на незалежні компоненти. Модель нічого не знає про такі речі, як HTTP–запити, і ви навіть не можете випадково змінити дані в шаблонах. Всі модулі можна використовувати незалежно від всієї системи, переносити з одного проекту в інший і підключати готові рішення. Існує велика кількість готових модулів, які можна додати в проект.

Для роботи з базами даних Django використовує власну ORM, де модель даних описується класом Python і на його основі генерується схема бази даних. Це опис даних у базі даних, який еквівалентний коду SQL CREATE TABLE [6].

Цей шаблон використовується для відображення сторінки. Тут ви можете знайти багато шаблонних тегів, які спрощують логічну структуру та створюють різні фільтри.

* крім того, Django постачається з багатьма попередньо написаними базовими речами, які в тій чи іншій формі існують майже в кожній сучасній веб–програмі;
* сесія (достатньо підключити потрібні модулі до програми, request.session є в кожному запиті, в ньому можна зберігати будь–які дані);
* реєстрація об’єктів моделі даних, авторизація, система дозволів, генерація паролів, відправка повідомлень електронною поштою.

Кешування (щоб не звертатися до бази щоразу, коли дані потрібно рідко змінювати, їх можна кешувати).

Django дотримується принципу DRY (Do not repeat yourself) – не повторюйтеся, тому програмісти намагаються не повторювати вже використаний код, тому кількість рядків зменшується [7].

Django використовує потужний інструмент під назвою успадкування схеми. Значення спадковості полягає в розвитку скелетної моделі «батько–дитина». Домашня сторінка містить загальний розділ і описує блоки, які перевизначені в шаблоні успадкування.

Django може легко вирішити проблеми безпеки додатків. Багато вбудованих механізмів запобігають використанню небезпечних розчинів. За замовчуванням Django екранує всі значення перед додаванням їх до бази даних, а перед цим він екранує HTML–код [7].

### 2.2.3 Засоби розробки інтерфейсу. Інтерфейс інтернет–магазину розроблений з використанням трьох основних елементів: HTML + CSS + JS. Це підключення використовується для будь–якої веб–програми.

HTML (Hypertext Markup Language) – стандартизована мова розмітки документів у Всесвітній павутині. CSS (Cascading Style Sheets) – формальна мова, яка використовується для опису зовнішнього вигляду документів, написаних за допомогою мови розмітки HTML [8].

HTML визначає структуру вмісту та його значення, визначаючи такі речі, як заголовки, абзаци чи зображення. CSS використовується для розробки зовнішнього вигляду вмісту, наприклад, використання шрифтів або кольорів.

Ці дві мови – HTML і CSS незалежні один від одного і повинні залишатися такими. CSS не слід писати в документах HTML і навпаки. Загалом, HTML завжди буде представляти вміст, тоді як CSS завжди буде визначати його дизайн [9].

JavaScript – мова програмування з кількома парадигмами. Підтримуються об’єктно–орієнтований, імперативний і функціональний стилі. є реалізацією мови ECMAScript. JavaScript часто використовується як вбудована мова для програмного доступу до об’єктів програми. Він найбільш широко використовується в браузерах як мова сценаріїв для додавання інтерактивності веб–сторінок [10].

Для реалізації системи автоматизації велосипедних вантажів використовується HTML5 + CSS3 + jQuery..

HTML5 є останньою версією HTML і має багато переваг, головним чином:

– перевірку форми тепер можна виконати класичним способом HTML;

– покращена семантика HTML5 дозволяє швидше та легше розрізняти нижній колонтитул, верхній колонтитул, панелі навігації;

– використання елементів HTML5, щоб зробити семантичне ядро ​​кожної сторінки більш видимим (відтепер для реалізації таких ідей використовувати лише стандартний код);

– покращена інтеграція з вмістом SVG (тому не потрібно встановлювати такі плагіни, як Flash, під час використання нових елементів);

– вставляти музику, графіку, відео та зображення без необхідності використання сторонніх програм, оскільки такі функції розроблені в самому браузері.

Розробники можуть значно покращити швидкість відгуку сторінки завдяки зменшенню розміру файлів cookie [8].

CSS3 відрізняється від попередніх версій тим, що він підтримує багато нових властивостей, таких як закруглені кути, градієнти, прозорість, тіні, трансформації, плавні переходи тощо [9].

jQuery – це бібліотека JavaScript, зосереджена на взаємодії між JavaScript і HTML. Бібліотека jQuery допомагає легко отримати доступ до будь–якого елемента DOM, отримати доступ до атрибутів і вмісту елементів DOM і маніпулювати ними. Бібліотека jQuery також надає зручний API для роботи з AJAX [11].

Ми використовуємо бібліотеку Leaflet для гри та взаємодії з картою.

Leaflet – це бібліотека з відкритим кодом, написана на JavaScript для відображення карт на веб–сайтах. Підтримує більшість мобільних і настільних платформ, які підтримують HTML5 і CSS3.

### 2.2.4 Вибір СУБД. СУБД повинна забезпечувати реляційну модель обробки даних. Сама модель передбачає певний тип зв’язку між сутностями з різних таблиць. Цей тип СУБД повинен мати специфічні структури (таблиці) для зберігання та використання даних. У таблиці кожен стовпець може містити різні типи даних. Кожен запис складається з багатьох атрибутів (стовпців) і має унікальний ключ, що зберігається в одній таблиці – усі ці дані пов'язані один з одним, як описано в реляційній моделі.

Розглянемо наступні СУБД.

SQLite – база даних, яку легко вбудовувати в програми. Оскільки система є файловою, вона надає досить широкий набір інструментів для роботи з нею порівняно з мережевими СУБД. Використовуючи цю СУБД, ви можете отримати прямий доступ до файлів (де зберігаються дані) замість портів і сокетів у мережевій СУБД. Ось чому SQLite дуже швидкий і також потужний завдяки технології бібліотеки сервісів.

Переваги SQLite:

* структура файлу – вся база даних складається з одного файлу, тому її легко перенести на іншу машину;
* ідеально підходить для розробки та тестування – під час розробки програм часто виникає потреба в розширеннях (SQLite надає все необхідне для цих цілей, оскільки складається лише з одного файлу та бібліотеки, написаної на C).

Недоліки SQLite:

* безкористувацькі системи – більші СУБД включають системи управління правами доступу користувачів (зазвичай, використання цієї функції не настільки важливо, тому що ця СУБД використовується для невеликих додатків);
* немає можливості підвищити продуктивність – знову ж таки, за дизайном, важко вичавити більшу продуктивність з цієї СУБД [11].

MySQL – найпоширеніша повноцінна серверна СУБД. MySQL – це потужна, вільно розповсюджувана СУБД, яку можна успішно використовувати на різноманітних веб–сайтах і веб–додатках. Навчитися користуватися цією СУБД дуже легко, оскільки ви можете легко знайти багато інформації в Інтернеті.

Хоча він не реалізує всі функції SQL, MySQL надає досить багато інструментів розробки додатків. Оскільки це серверна СУБД, програми доступу до даних, на відміну від SQLite, використовують сервіс MySQL.

Переваги MySQL:

* простота використання – Встановлення MySQL дуже просте. Існують різні програми, такі як GUI, які дуже спрощують роботу з базами даних;
* розширеній функціонал – MySQL підтримує більшість функцій SQL;
* безпека – велика кількість функцій забезпечують безпеку та підтримуються за замовчуванням;
* масштабованість – MySQL може легко обробляти великі обсяги даних і її легко розширювати;
* швидкість – спрощення кількох критеріїв дозволяє MySQL значно підвищити продуктивність.

Недоліки MySQL:

* відомі обмеження – За своєю конструкцією MySQL має деякі функціональні обмеження, які іноді необхідні в особливо вимогливих програмах;
* проблеми з надійністю – через певні методи обробки даних MySQL іноді не така надійна, як інші СУБД [11].

PostgreSQL є найбільш спеціалізованою з усіх трьох СУБД. Він вільно поширюється і максимально відповідає стандарту SQL. У міру випуску нових версій PostgreSQL намагається своєчасно використовувати всі переваги стандарту ANSI/ISO SQL.

Що відрізняє PostgreSQL від інших СУБД, так це його підтримка популярних об’єктно–орієнтованих та/або реляційних підходів до баз даних. Наприклад, комплексна підтримка надійних транзакцій, а саме атомарність, послідовність, ізоляція, міцність. Завдяки потужним технологіям PostgreSQL дуже ефективний. Паралелізм досягається не блокуванням операцій читання, а реалізацією керування кількома паралелізмами (MVCC). PostgreSQL легко розширює власні процедури, які називаються збереженими процедурами. Ці функції спрощують використання повторних операцій.

Хоча PostgreSQL не такий популярний, як MySQL, існує чимало програм, які можуть легше використовувати PostgreSQL, незважаючи на його дуже потужні функції. Встановити цю СУБД тепер дуже просто за допомогою стандартного менеджера пакетів операційної системи.

Переваги PostgreSQL:

* програмне забезпечення з відкритим кодом відповідає стандарту SQL;
* велика кількість надбудов – незважаючи на велику кількість вбудованих функцій, є багато надбудов, які дозволяють розробляти та керувати даними для цієї СУБД;
* розширення – можливість розширення функціональності за рахунок збереження її програми;
  + - об'єктно–орієнтована – PostgreSQL є не тільки реляційною СУБД, але й об'єктно–орієнтованою.

Недоліки PostgreSQL:

* + - продуктивність – завдяки простим операціям читання PostgreSQL може значно сповільнити сервер і бути повільнішим за своїх конкурентів, таких як MySQL;
    - популярність – незважаючи на значне співтовариство, ця СУБД за своєю природою не може похвалитися популярністю;
    - хостинг – через вищезазначені фактори іноді важко знайти хостинг, який підтримує цю СУБД [11].

Розглядаючи запропоновані СУБД, можна зробити висновок, що найкращим рішенням для розробки є SQLite.

Недолік даної СУБД для системи розробки несерйозний, оскільки користувальницька система управляється фреймворком Django, наявність цієї функції в СУБД не є обов'язковою, також не є важливим критерієм швидкість роботи інтернет–магазину. Переваги ще більш значні: вся база даних складається з одного файлу, і немає необхідності встановлювати різні сервіси, що сприяє економії місця на хостингу. Крім того, СУБД легко встановити та налаштувати, і, що більш важливо, вона масштабована.

## **2.3 Аналіз вимог на розробку і проектування**

### 2.3.1 Аналіз варіантів використання. У процесі проектування ми розглянемо основні вимоги до автоматизованої паспортної системи для магазинів. Дослідіть і сформулюйте функціональні вимоги у формі моделей прецедентів.

Варіант використання – це незалежне від реалізації багаторівневе представлення конкретної функціональності системи, що розробляється. Варіант використання розглядається як послідовність дій, які виконує система у відповідь на події, ініційовані зовнішніми об’єктами (активними особами, так званими акторами) [12]. Іншими словами, такий перегляд призначений для спрощення взаємодії з майбутніми користувачами системи, клієнтами, і особливо корисний для визначення необхідних характеристик системи.

Компіляція варіантів використання базується на аналізі вимог інформаційної системи електронного бізнесу. Основними гравцями в контексті програмного засобу є адміністратори та користувачі цієї системи.

Адміністратори мають повноваження додавати продукти, встановлювати характеристики продуктів, ціни та категорії, а також видаляти чи змінювати існуючі продукти. У міру просування продажів адміністратори можуть відстежувати статистику продажів, відвідувань, активності тощо.

Користувачі повинні мати можливість переглядати, шукати, сортувати та класифікувати продукти перед безпосередньою покупкою. Також має бути можливість переглядати огляди продуктів, що цікавлять, а також переглядати рейтинги вибраних продуктів.

На рисунку 2.1 показана діаграма варіантів використання в нотації UML.

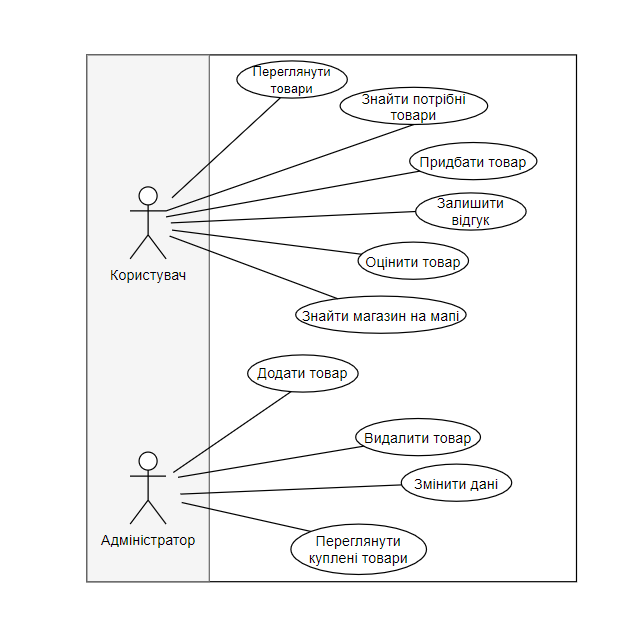


Рисунок 2.1 – Діаграма варіантів використання

Виділення основних варіантів використання полегшить процес подальшого проектування.

### 2.3.2 Проектування моделі бази даних. Для проектування моделі бази даних було використано середовище Draw.io та його метод IDEF1X, в якому передбачено два рівні представлення – логічний та фізичний.

Логічний рівень, як правило, виражає відображення конкретних фактів у реальному житті. Тоді як логічний рівень не враховує використання конкретної СУБД, не визначає типи даних (цілі чи дійсні) і не визначає індекси на таблицях [3].

Спроектуємо логічну модель бази даних інформаційної системи інтернет–магазину. Модель міститиме 12 таблиць, об’єднаних логічними з’єднаннями.

Для того щоб зберігати дані в базі необхідно створити такі таблиці:

* «Категорія товару» – використовується для зберігання даних про категорію: ідентифікатор категорії, назва;
* «Характеристики товару» – використовується для зберігання даних про різні атрибути товару: ідентифікатор характеристики, назва; – «Магазин» – для зберігання даних про магазин: ідентифікатор магазину, назва, адреса, координати;
* «Товар» – використовується для зберігання даних про різні товари: ідентифікатор товару, назва, ціна, дата додавання, зображення, гарантія, відображення, перегляд, опис, новинка, ідентифікатор категорії;
* «Товар – Характеристики товару» – допоміжна таблиця для зберігання даних про атрибути, притаманні одному товару, що дозволяє уникнути зв’язків «багато до багатьох»: ідентифікатор товару, ідентифікатор характеристики, значення;
* – «Product–Stores» – допоміжна таблиця для зберігання даних про магазини, доступні для товару, що дозволяє уникнути зв’язків «багато–до–багатьох»: product–id, store–id, quantity;
* «Профіль клієнта» – використовується для зберігання даних про клієнта: ім’я, email, пароль, номер телефону, стать, дата народження;
* «Список побажань» – допоміжна таблиця для зберігання даних про товари, які клієнти хочуть купити пізніше, уникає зв’язків «багато–до–багатьох»: ідентифікатор товару, ідентифікатор клієнта;
* «Історія продажів» – використовується для зберігання даних про продажі: ID клієнта, дата; – «Товарна позиція» – допоміжна таблиця для зберігання даних про продані товари, що дозволяє уникнути зв’язків «багато–до–багатьох»: ідентифікатор товару, ідентифікатор продажу, кількість;
* «Кошик для покупок» – це допоміжна таблиця, яка використовується для зберігання даних для прямої покупки доданих товарів, яка може уникнути зв’язків «багато–до–багатьох»: ID продукту, ID користувача, Кількість;
* «Відгуки» – Допоміжна таблиця для зберігання відгуків про товари, уникаючи зв'язків «багато–до–багатьох»: ідентифікатор товару, ідентифікатор користувача, дата, рейтинг, текст відгуку.

Логічна модель має наступний вигляд (рисунок 2.2):

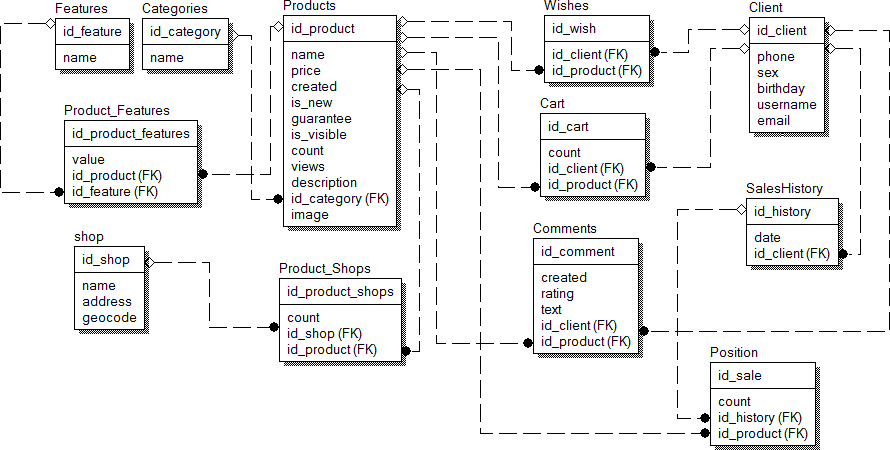


Рисунок 2.2 – Логічна модель даних

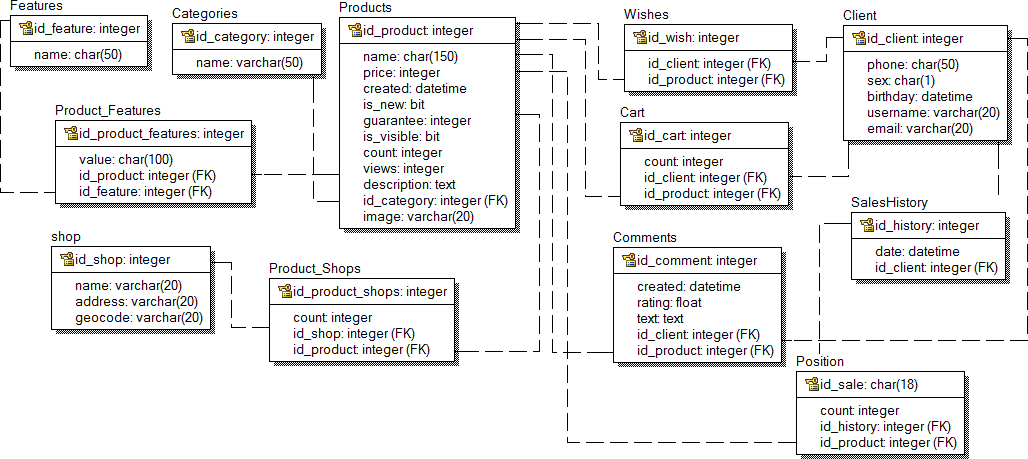
Фізичний рівень моделі складають імена об'єктів і типи даних, індекси. Фізична модель представлена на рисунку 2.3.

Рисунок 2.3 – Фізична модель даних

Фізична і логічна моделі дозволяють розглянути всі зв'язки і типи даних перед реалізацією, що полегшує створення бази даних.

### 2.3.3 Проектування користувальницького інтерфейсу. Інтерфейсами вважаються так звані канали зв'язку, за допомогою яких користувачі інформаційних систем взаємодіють з комп'ютерами.

Хороший користувальницький інтерфейс можна вважати таким, який користувачі не заперечуватимуть і фактично не помітять. Такий інтерфейс називають прозорим – ніщо не відволікає користувача від основного вмісту сторінки [13].

Користувачі, які відвідують інформаційні системи, в більшості випадків приходять з певною метою, і їх першочерговим завданням є швидкий і легкий доступ до потрібної інформації або товару, тому для розробки інтерфейсу дуже необхідно організувати простір підхід враховує всі особливості та потреби покупця.

Виділимо основні принципи, які слід враховувати при розробці призначених для користувача інтерфейсів.

Чіткість – головна мета будь–якого інтерфейсу. Для того, щоб ефективно використовувати розроблений інтерфейс, люди повинні легко знайти його, усвідомити, чому вони його використовують, зрозуміти, з яким об’єктом він надає взаємодію, уявити, що відбувається під час його використання, і, нарешті, мати можливість взаємодіяти з ним.

Перший контакт з інтерфейсом є критичним моментом, і в той же час дизайнери схильні забути про це. Щоб допомогти користувачам якомога швидше адаптуватися до інтерфейсу, дизайн повинен враховувати точку входу – момент, коли нічого не відбувається. Головна сторінка повинна вказувати напрямок руху. Більшість проблем взаємодії виникають на початку роботи, і користувачі досягнуть успіху, знаючи правила..

Функціональність важливіша за зовнішній вигляд. Люди почуваються найбільш комфортно, коли їх оточують об’єкти, поведінку яких вони можуть передбачити – інші люди, тварини, речі, програми. Коли хтось або щось поводиться так, як ми очікуємо, ми відчуваємо співчуття до них. При тривалій взаємодії з Інтернетом користувачі можуть стикатися з певними стереотипами, які є умовною інстинктивною системою. Тому зовнішній вигляд елементів дизайну повинен відповідати задуму користувача. Заходячи на сайт, людина заздалегідь передбачає, в якій частині сторінки буде розташована потрібна йому інформація або пункт меню. На практиці це означає, що будь–який користувач повинен зрозуміти, як це працює, подивившись на той чи інший елемент інтерфейсу.

Загальна композиція сторінки заснована на наступних принципах – текст відображається зліва направо і зверху вниз. Тому найважливіші елементи повинні розташовуватися в зоні початкового фокусування, щоб користувачі не відволікалися на другорядну інформацію. Тобто у верхній лівій частині сторінки. Найбільш змістовна інформація зазвичай знаходиться в центральній частині сторінки. Що менш важливо, допоміжна інформація має бути праворуч і внизу праворуч на сторінці. Дані одного типу на різних сторінках повинні розташовуватися в одному регіоні [13].

Існують інші способи привернути увагу користувачів не лише за допомогою просторового розташування, але й за допомогою помітного розміру. Зв’язок між розміром об’єкта простий: чим більший об’єкт, тим раніше користувач помітить його. Яскраві деталі легше сприймати, ніж темні, а кольорові шрифти також можуть привернути увагу користувачів.

При розробці дизайну інформаційної системи продажу автозапчастин слід враховувати особливості розгляду та принципи побудови інтерфейсу користувача.

# РОЗДІЛ 3 РОЗРОБКА WEB–САЙТУ ДЛЯ ПРОДАЖУ АВТОЗАПЧАСТИН

## **3.1 Принципи розробки**

Програма Django описує набір дій, які можуть відбуватися під час взаємодії сервера та клієнта. Ці дії можуть включати відображення веб–сторінок, виконання запитів до бази даних, запис даних на диск тощо.

Django використовує шаблон MVC, тому структура програми виглядатиме так. Файл models.py описує моделі даних як класи Python. Дані зберігатимуться в реляційній базі даних, але завдяки технології ORM дані використовуються як об’єкти моделі.

Інтерфейси – це зв’язки між шаблонами Django. Структура сторінки представлена ​​базовим шаблоном, який містить верхню частину сайту. Цей шаблон успадковується іншими, і вони замінюють один одного при переході з одного розділу в інший. Це досягається за допомогою системи успадкування шаблонів.

Сторінка заповнюється даними, отриманими з моделі. views.py міститиме всі можливі керуючі функції та класи. Вони контролюють, які шаблони використовувати та які модельні дані та методи використовувати. Діаграма компонентів представлена на рисунку 3.1.

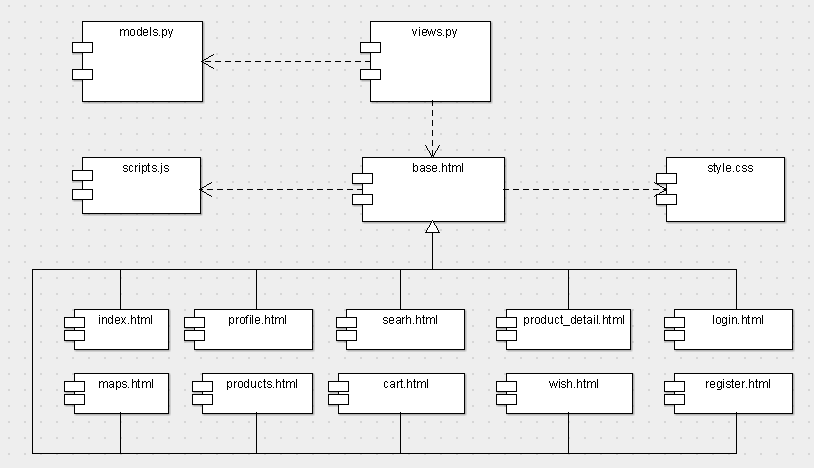


Рисунок 3.1 – Діаграма компонентів

Автоматизована система продажів складається з двох частин. По–перше, клієнтська частина відповідає за взаємодію з клієнтами. Це інтернет–магазин, який пропонує користувачам різноманітні можливості.

Незареєстровані користувачі можуть переглядати каталог товарів (всі товари, новинки, бестселери), сортувати товари (за рейтингом, ціною, результатами пошуку, відгуками) і використовувати пошук.

Для того, щоб придбати продукцію, вкажіть при реєстрації на сайті. Після цього користувач може додати товар у кошик, встановити необхідну кількість товару та завершити покупку. При необхідності продукти можна вилучити з кошика. У профілі можна скористатися історією покупок, а також змінити персональні дані або пароль. Якщо у вас немає достатньо грошей, щоб купити його, ви можете додати свій улюблений продукт до списку бажань, щоб не втратити його.

Діаграма діяльності для реєстрації користувача представлена на рисунку 3.2.



Рисунок 3.2 – Діаграма діяльності

Друга частина – адміністративна група. Доступ мають лише користувачі з правами адміністратора. За допомогою панелі адміністратори можуть заповнювати базу даних продуктами, категоріями, атрибутами та іншою необхідною інформацією, а також редагувати або видаляти дані.

У розділі Продажі можна переглянути історію покупок користувачів. У розділі магазину адміністратори можуть вказати географічні координати, щоб користувачі могли знайти на карті найближчий до улюбленого товару магазин.

Код додатку представлений в додатку Б.

## **3.2 Розробка моделі даних**

У Django таблиці бази даних представлені як моделі. Моделі є основним джерелом даних. Він містить набір полів і даних, що зберігаються в базі даних. Django приймає принципи ORM, тому він підтримує основні принципи ООП, такі як успадкування, поліморфізм та інкапсуляція.

Опис моделі даних схожий на SQL CREATE TABLE з такими характеристиками:

– назви таблиць автоматично створюються з назв програм і назв моделей у нижньому регістрі;

– автоматично додавати первинні ключі;

– django додає "\_id" до імен зовнішніх ключів. Структури бази даних описуються за допомогою класів Python. Кожен клас успадковує базовий клас Django "Model", який, у свою чергу, містить методи для додавання, видалення та зміни даних.

Діаграма класів представлена на рисунку 3.3.

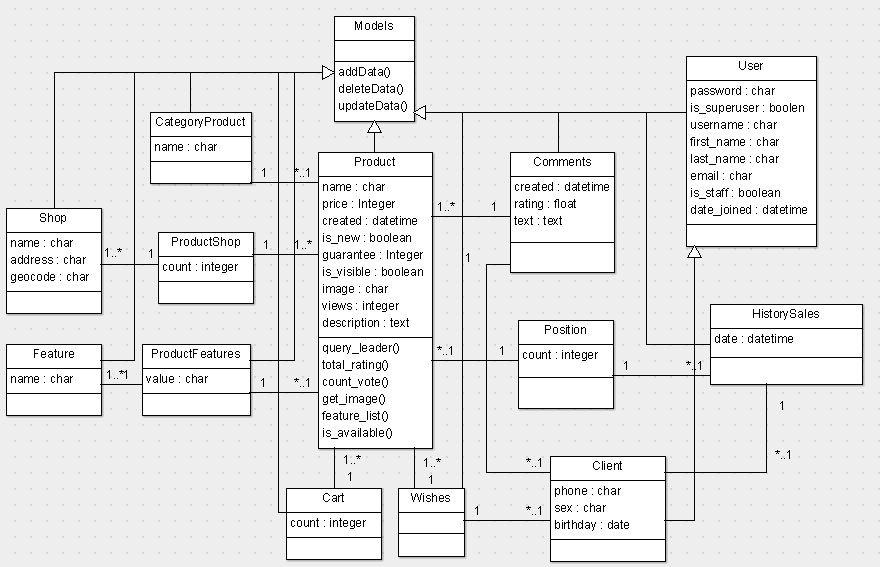


Рисунок 3.3 – Діаграма класів

Представлені класи:

* клас «CategoryProduct» – описує таблицю «Категорії товарів»;
* клас «Feature» – описує таблицю «Характеристики»;
* клас «Shop» – описує таблицю «Магазини»;
* клас «Product» – описує таблицю «Товари»;
* клас «ProductFeature» –опісиваеттабліцу «Товар–Характеристики товарів »;
* клас «ProductShop» – описує таблицю «Товар–Магазини»;
* клас «Client» – описує таблицю «Профіль користувача»;
* клас «ClientWish» – описує таблицю «Список бажань»;
* клас «HistorySales» – описує таблицю «Історія продажів»;
* клас «ClientCart» – описує таблицю «Кошик»;
* клас «Comment» – описує таблицю «Коментарі»;
* клас «Position» – описує таблицю «Позиція товару».

Крім моделей даних для роботи магазину знадобляться кілька властивостей і методів:

* query\_leader (cls, top), метод класу «Product», визначає лідерів з продажу;
* total\_rating (self), властивість об'єкта класу «Product», вираховує загальний рейтинг певного товару;
* count\_vote (self), властивість об'єкта класу «Product», визначає кількість голосів певного товару;
* get\_image (self), властивість об'єкта класу «Product», повертає url до зображення в html вигляді;
* feature\_list (self), метод об'єкта класу «Product», повертає список характеристик зазначеного товару;
* is\_available (self), метод об'єкта класу «Product», повертає Tuae, якщо товар є в наявності.

## **3.3 Структура сторінок веб–додатки**

Основним шаблоном є base.html. Він містить назву сайту, логотип, блог із можливістю пошуку та навігацію. Доступ до цих елементів необхідний з будь–якої частини сайту, тому під час візуалізації сторінки завжди використовуватиметься базовий шаблон. Він успадковується іншими шаблонами (рис. 3.4).

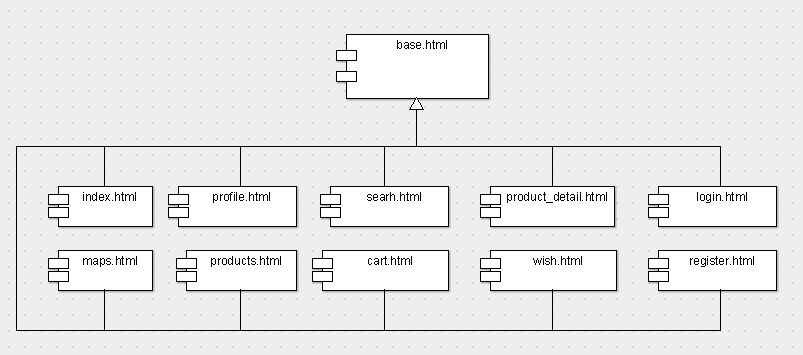


Рисунок 3.5 – Спадкування шаблонів

На рисунку представлені наступні шаблони:

* index.html, містить в собі структуру головної сторінки;
* products.html, є каталогом товарів, в залежності від параметрів вибірки відображає потрібні товари;
* product\_detail.html, сторінка с детальною інформацією про конкретний товар, характеристиками, рейтингом, відгуками та ін.;
* cart.html, користувацька корзина з можливістю змінювати кількість товарів, видаляти або оформити покупку;
* wish.html, список бажаних товарів користувача;
* login.html, сторінка авторизації користувача;
* register.html, сторінка реєстрації нового користувача;
* profile.html, профіль користувача, є можливість змінювати інформацію, змінити пароль, подивитися історію покупок;
* maps.html, сторінка з відображенням магазинів на карті;
* search.html, сторінка з результатами пошуку.

## **3.4 Розробка користувальницького інтерфейсу**

Враховуючи вимоги, висунуті в розділі дизайну, розроблено інтерфейс користувача.

Розглянемо домашню сторінку веб–додатку, наприклад: домашню сторінку інтернет–магазину, де відображаються акції та пропозиції, каталог товарів, які можна вибирати та сортувати за різними параметрами, сторінка з детальним описом товар, його характеристики та відгуки, сторінка, де зберігається товар. Карта магазину та кошик для покупки, де користувач розміщує всі товари перед покупкою.

Верхня половина сайту – це окремий шаблон, який містить функціональність, необхідну для будь–якої частини інтернет–магазину. Цей шаблон успадковується всіма іншими шаблонами.

Інтерфейс головної сторінки представлений на рисунку 3.6.

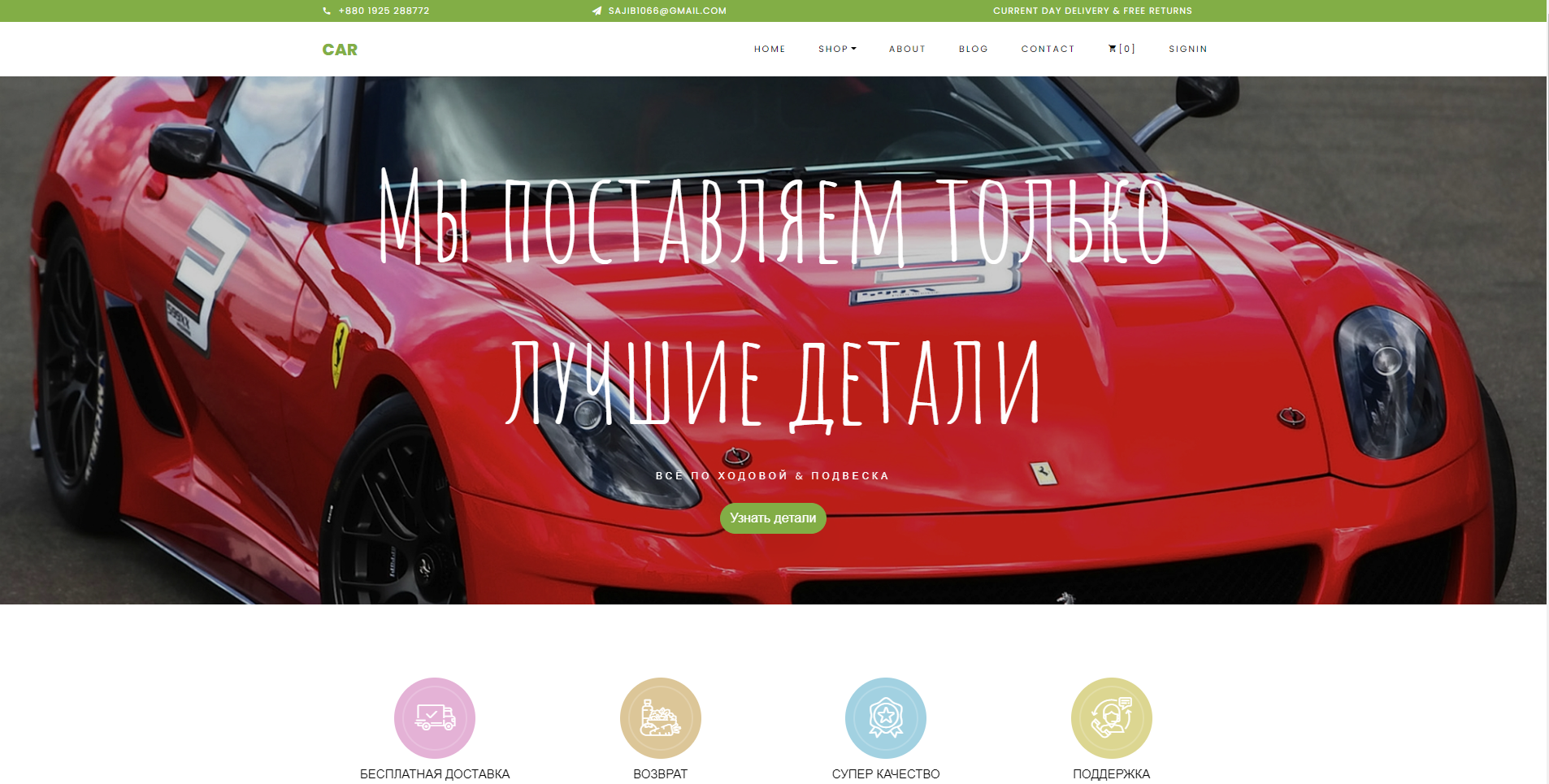


Рисунок 3.6 – Головна сторінка

Інтерфейс можна умовно розділити на три частини. Верхній – заголовок сайту, середній – слайдер, панель вибору категорії товарів і блоки для популярних товарів, нижній – інформація про магазин.

Верхня частина сайту містить основні функції та елементи навігації інтернет–магазину. Є інформація про товари в кошику, їх загальна вартість. Далі йде форма пошуку потрібного товару. Вище посилання на «Авторизація», «Список бажань», «Профіль», і кнопка для виходу з профілю. Є також кнопки, які ведуть до каталогу товарів, де ви можете ознайомитися з новими продуктами або найкращими продавцями у вашому магазині.

Центральну частину сайту займає анімований слайдер з цікавими пропозиціями та знижками. Панель вибору категорії продукту знаходиться над повзунком. Нижче представлені найпопулярніші товарні панелі в магазині. Товари можна відразу додати в кошик або переглянути детально.

Нижня половина сайту містить інформацію про магазин, номер телефону, адресу основного магазину та доступні способи оплати товару.

Інтерфейс каталогу товарів представлений на рисунку 3.7.

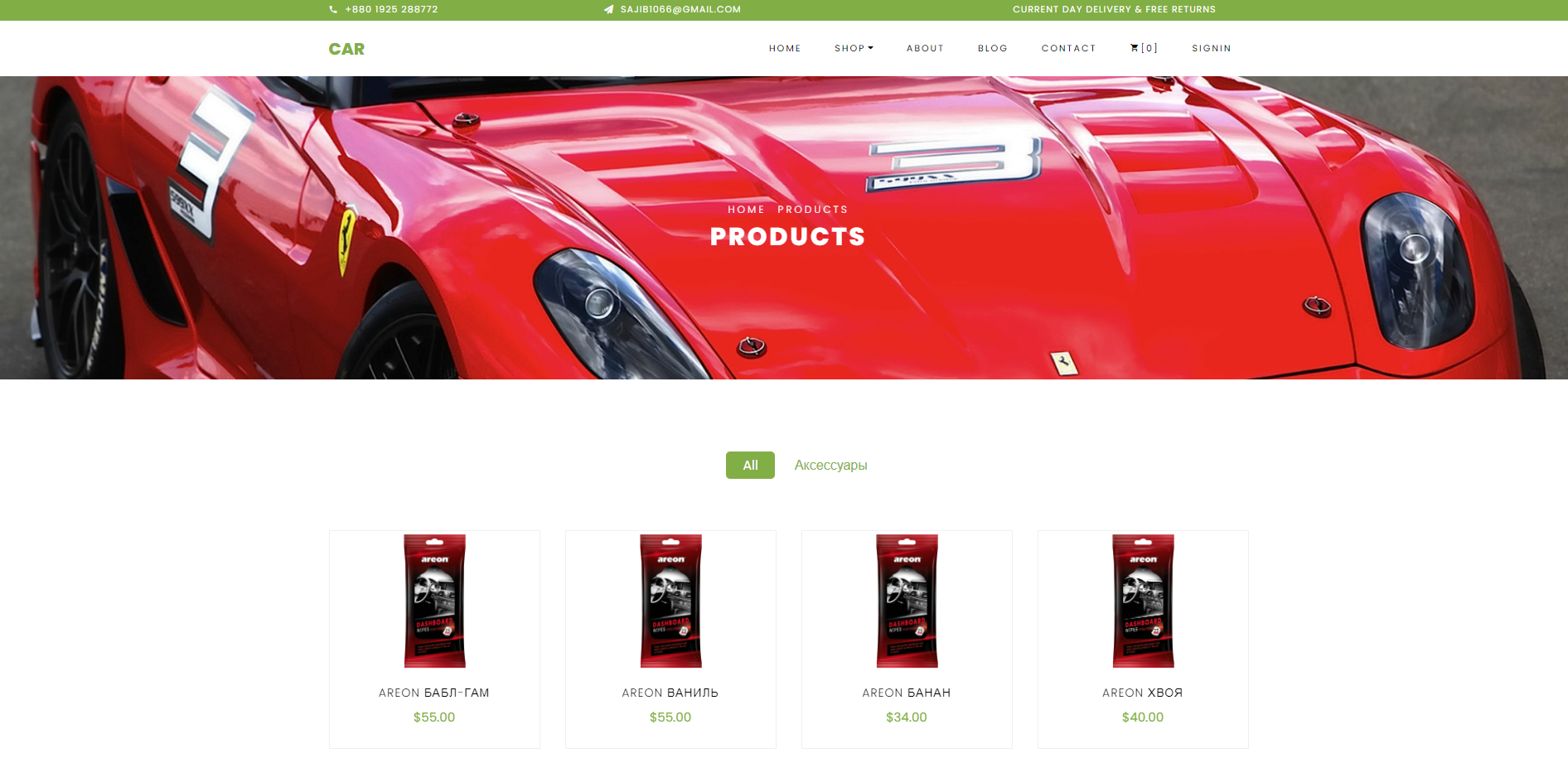


Рисунок 3.8 – Каталог товарів

Тут верхня і нижня частини інтерфейсу залишаються незмінними, а середня частина – це каталог товарів. Продукти відображаються по одній сторінці. Є меню для вибору категорій товарів. Новинки позначаються спеціальним знаком. Поточні значки представляють думки та відгуки про певний продукт. Товари, оцінені користувачами, мають оцінки.

Інтерфейс сторінки певного товару представлений на рисунку 3.9.

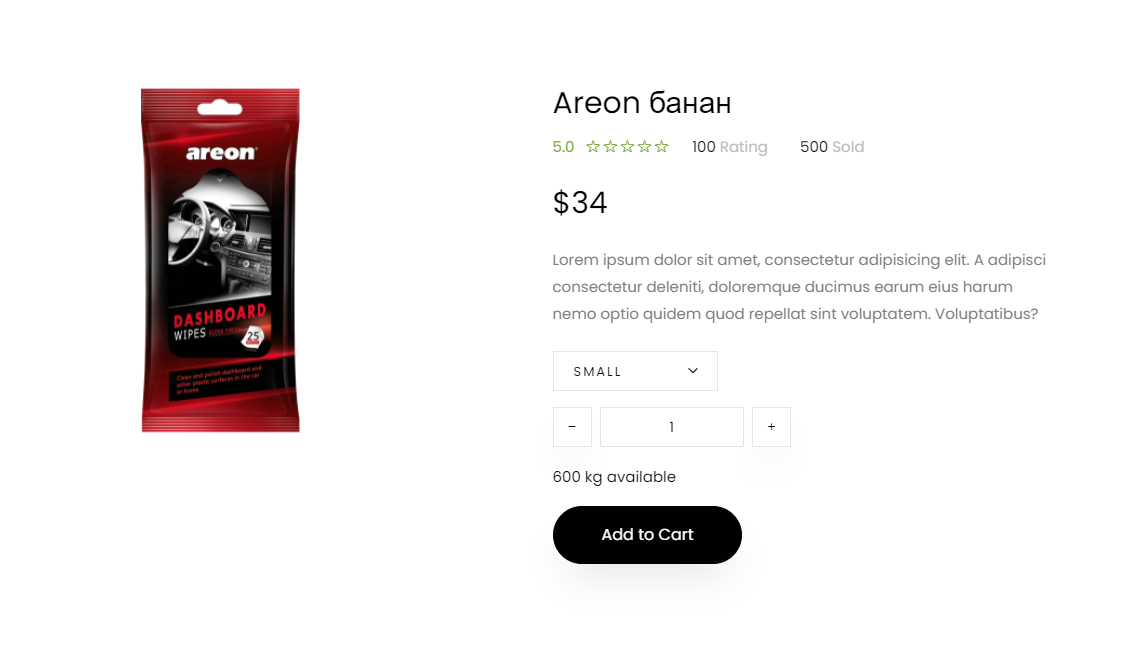


Рисунок 3.9 – Сторінка товару

Сторінки продуктів містять детальну інформацію про конкретний товар. У верхній частині відображається назва товару, зображення, ціна, рейтинг, гарантія та кнопка «Додати в кошик». Розмір виробу можна вибрати.

У розділі «Опис» є детальний опис товару, у розділі «Відгуки» ви можете переглянути відгуки інших покупців або залишити свої.

На рисунку 3.10 показано інтерфейс сторінки з картою міста та вкладками магазину..

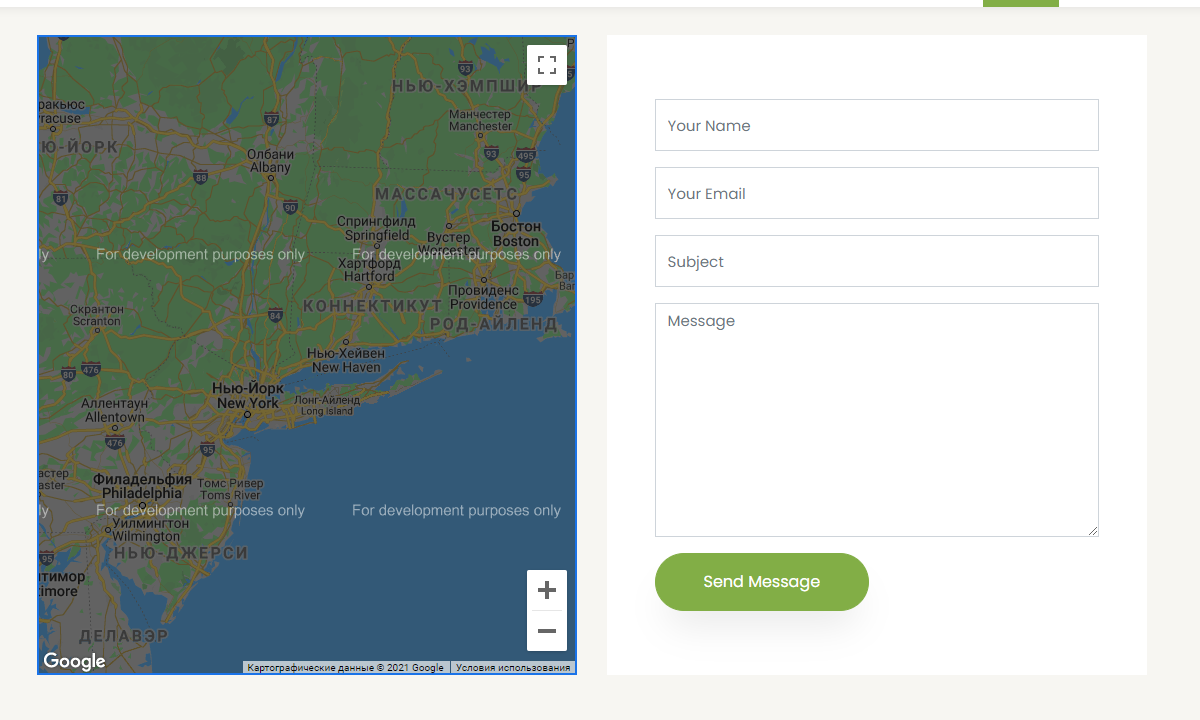


Рисунок 3.10 – Карта магазину

На карті будуть позначені магазини, в яких є обраний товар. Ви можете залишити повідомлення на правій панелі.

Інтерфейс сторінки з товарами в кошику представлена на рисунку 3.11.



Рисунок 3.11 – Кошик

У кошику ви можете змінити кількість товарів, видалити непотрібні товари та завершити покупку.

Інтелектуальний підбір автозапчастини на прикладі двірників для автомобіля. Для того щоб покупець був максимально впевнений у своєму виборі на допомогу приходить «інтелектуальний підбір».

Інтерфейс сторінки з інтелектуальним підбором двірників авто представлена на рисунку 3.12 та 3.13 та 3.14



Рисунок 3.12 – Інтелектуальний підбір



Рисунок 3.13 – Інтелектуальний підбір двірників



Рисунок 3.14 – Інтелектуальний підбір двірників по кузову авто

Реалізований він таким чином що покупцю потрібно крок за кроком надати інформацію про його авто, на основі обраних даних система додає можливі варіанти вибору, в решті решт покупець отримує лист с двірниками що точно будуть підходити саме його автомобілю.

Інтерфейс сторінки з товарами в кошику представлена на рисунку 3.15

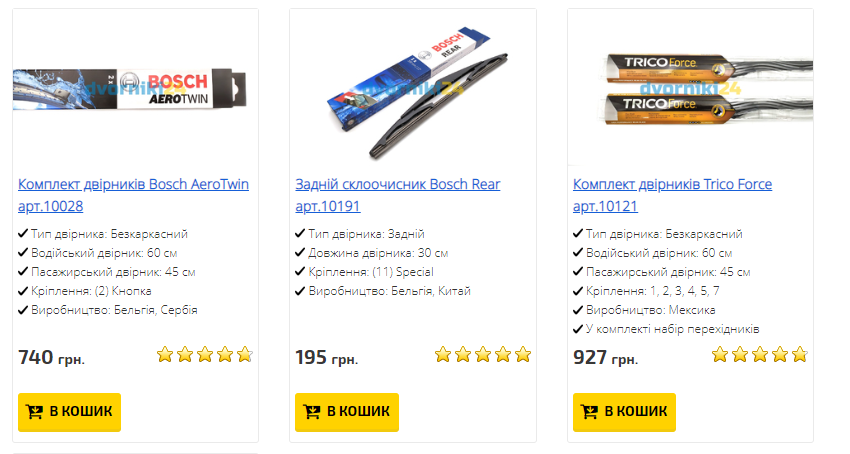


Рисунок 3.15 – Результат підбору

## **3.5 Адміністративна панель**

Панель адміністратора – це панель управління сайтом, якою може користуватися будь–який користувач без навичок програмування. Весь матеріал вноситься на сучасний веб–сайт за допомогою системи управління контентом (CMS).

Розглянемо приклад, щоб зрозуміти, що означає адміністратор для сайту. Перед капітаном космічного корабля знаходиться пульт з різноманітними кнопками, джойстиками та екранами. І він знає, яку кнопку натиснути, щоб комп'ютер видав інформацію про обстановку на борту, а корабель рушив вліво, або, наприклад, стрибнув у гіперпростір.

Панель адміністратора – це панель управління космічним кораблем. Основна відмінність веб–ресурсів на основі Django полягає у використанні зрозумілих систем керування базами даних, текстовими, графічними та іншими файлами.

Інформаційні системи без адміністратора також можуть використовувати базу даних і всі перераховані вище компоненти. Але, наприклад, щоб встановити зображення на сторінку в Django, вам потрібно зробити лише одне: натиснути відповідну іконку у вбудованому редакторі та слідувати інструкціям. Ризик помилки виключається на 99,9%.

Але щоб встановити зображення на кодові сторінки (без панелі адміністратора), вам доведеться виконати багато кроків:

1. Завантажте зображення на сервер.

2. Знайдіть відповідний розділ у базі даних або в html–коді сторінки, де ви хочете розмістити зображення.

3. За допомогою тегу без помилок запишіть код відображення зображення.

4. Відкрийте сторінку та перевірте відображення зображення. Поверніться до розділу коду та за потреби виправте помилки. Для редагування інформації в інформаційній системі необхідно мати доступ до панелі адміністрування, тобто мати обліковий запис з логіном і паролем, які необхідно ввести в полі авторизації.

Інтерфейс адмін–панелі з полем авторизації представлена на рисунку 3.16

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 3.16 – Поле авторизації

Адміністратором вважається особа, яка керує роботою інформаційної системи. Створення постів і додавання товарів на сайт здійснюється безпосередньо через панель адміністратора. Після авторизації адміністратор отримує доступ до зручного інтерфейсу управління інформаційною системою.

Інтерфейс адмін панелі представлена на рисунку 3.17

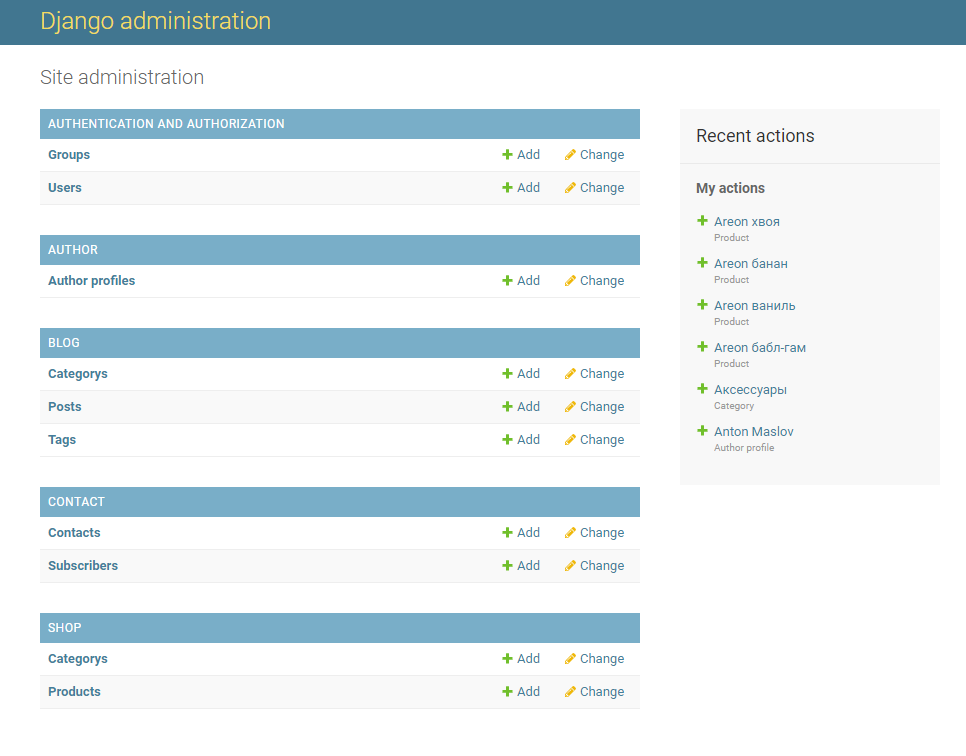


Рисунок 3.17 – Інтерфейс адміністративної панелі

У розділі «Товари» знаходиться перелік товарів для інформаційної системи. Адміністратори можуть редагувати, видаляти та додавати нові продукти. Додані продукти відображатимуться в інтерфейсі користувача після збереження.

Інтерфейс адмін панелі розділу «Товари» представлена на рисунку 3.18

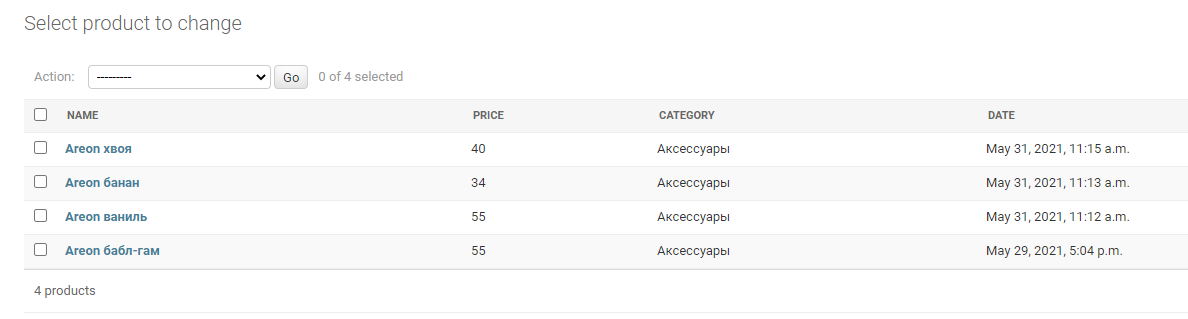


Рисунок 3.18 – Розділ «Товари»

Щоб додати новий товар, адміністратору необхідно заповнити поля товару, вказати ціну, завантажити фото товару, створити опис тощо.

Інтерфейс адмін панелі розділу «Товари» представлена на рисунку 3.19



Рисунок 3.19 – Розділ «Товари»

Після натискання кнопки Save, товар буде додано до бази даних та відображено в користувацькому інтерфейсі.

# РОЗДІЛ 4 ТЕСТУВАННЯ

Тестування програмного забезпечення – це процес виявлення помилок у програмному забезпеченні (ПО). Існуючі методи тестування програмного забезпечення не дозволяють однозначно і повністю усунути всі дефекти і помилки і визначити коректну працездатність аналізованої програми, особливо в закритих пропрієтарних програмах. Тому всі існуючі методи тестування діють у рамках формального процесу перевірки програмного забезпечення, що досліджується або розробляється [18].

Цей процес формальної перевірки або верифікації може виявитися вільним від дефектів з точки зору використовуваних методів, тобто немає способу точно встановити або гарантувати, що програмний продукт вільний від дефектів, враховуючи, що людина елемент існує протягом життєвого циклу програмного забезпечення на всіх етапах.

Існує багато способів вирішення завдань тестування та верифікації програмного забезпечення, але ефективне тестування складних програмних продуктів є дуже творчим процесом, який не обмежується прийняттям строго визначених процедур або створенням таких процедур.

Одним із методів тестування є метод функціонального тестування.

Функціональне тестування – це тестування програмного забезпечення, яке проводиться з метою перевірки можливості реалізації функціональних вимог, тобто проведення функціонального тестування дозволяє перевірити здатність інформаційної системи вирішувати завдання, необхідні користувачам за певних умов [18].

Функціональне тестування – один із ключових видів тестування, завданням якого є визначення відповідності розробленого програмного забезпечення початковим функціональним вимогам замовника.

Ключові переваги функціонального тестування:

– функціональне тестування програмного забезпечення повністю імітує реальне використання системи;

– дозволяє своєчасно виявляти системні помилки програмного забезпечення, що дозволяє уникнути багатьох проблем при його використанні в майбутньому;

– економія за рахунок виправлення помилок на початку життєвого циклу програмного забезпечення.

Проведемо перевірку працездатності системи автоматизації в магазині автозапчастин.

За результатами перевірки автоматизована система магазину автозапчастин працює нормально. Результати тестування наведено в Додатку А.

# ВИСНОВОК

У ході виконання випускної кваліфікаційної роботи було проведено аналіз предметної області, в рамках якого розглянуто існуючий веб–додаток для мерчандайзингу, на основі якого розроблялася структура та основні функції системи. Крім того, проведено функціональний аналіз вартості робіт та визначено остаточну вартість розробки.

Реалізовану функцію підбору автозапчастин за допомогою «штучного інтелекту», який надає чітку та коректну інформацію на результатах вибору користувача інформаційної системи.

Проаналізовано схему використання інформаційної системи електронної торгівлі автозапчастинами з урахуванням проектування та розробки програмних засобів. Тому була розроблена інформаційна система для магазину автозапчастин, яка дозволяє продавати товари в Інтернеті, надає зручний інтерфейс для управління магазином, інтегрується з картами, які допоможуть покупцям знайти найближчий магазин.

Тестування програмного забезпечення включає визначення методу тестування та проведення тестування. Результати функціональних тестів свідчать про правильність роботи системи.

Завдяки фреймворку Django розробку можна значно прискорити за рахунок компонентного підходу та реалізації принципу DRY. Компонентна архітектура спрощує визначення та підтримку єдиного стилю програми та правил поведінки.

Розроблена система може бути додатково вдосконалена шляхом впровадження в додаток можливостей аналізу продажів, наприклад, за допомогою кластеризації або асоціативних зв’язків.

# CПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Елементи інтернет–магазину [Електронний ресурс]. – URL:[http://www.web.starport.ua/ecommerce/pg3\_0.php](http://www.web.starport.ru/ecommerce/pg3_0.php)
2. Model View Controller [Електронний ресурс]. – URL: https://ua.wikipedia.org/wiki/Model–View–Controller
3. Фреймворк Django [Електронний ресурс]. – URL:[http://djbook.ua/rel1.8/intro/overview.html](http://djbook.ru/rel1.8/intro/overview.html)
4. Загальний огляд архітектури Django [Електронний ресурс]. – URL:<http://kutaloweb.com/jeff_forcier_glava_3/obschiy_obzor_arhitektury_django/>
5. Розробка сайтів з використанням HTML5 [Електронний ресурс]. – URL: https: //[www.prcomm–spb.ua/html–5.html](http://www.prcomm-spb.ru/html-5.html)
6. СSS3 властивості [Електронний ресурс]. – URL: https://webformyself.com/sss3–svojstva/
7. Сучасний підручник Javascript [Електронний ресурс]. – URL: https://learn.javascript.ua/
8. Порівняння систем управління базами даних [Електронний ресурс].

– URL: [http://devacademy.ua/posts/sqlite–vs–mysql–vs–postgresql/](http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/)

1. Діаграма варіантів використання [Електронний ресурс]. – URL:[http://www.intuit.ua/studies/courses/32/32/lecture/1004](http://www.intuit.ua/studies/courses/32/32/lecture/1004%20)
2. Основи побудови інтерфейсів [Електронний ресурс]. – URL:<http://www.info–system.ua/interface/begin_design_interface.html>
3. Принципи розробки інтерфейсу [Електронний ресурс]. – URL:<http://www.cmsmagazine.ua/library/items/usability/principles–of–user–interface–>design /
4. Функціонально–вартісний аналіз [Електронний ресурс]. – URL:[http://www.iso.ua/print/ua/document5954.phtml](http://www.iso.ua/print/ua/document5954.phtml%20)
5. Метрики коду програмного забезпечення [Електронний ресурс]. – URL: https: //[www.viva64.com/ua/a/0045/](http://www.viva64.com/ru/a/0045/)
6. Якісний аналіз програмного модуля на основі метрик коду [Електронний ресурс]. – URL: https://vunivere.ua/work18973
7. Тестування програмного забезпечення [Електронний ресурс]. – URL:[http://www.dpguap.ua/testing.htm](http://www.dpgrup.ru/testing.htm)

# ДОДАТОК А РЕЗУЛЬТАТИ ТЕСТУВАННЯ



Рисунок 4.1.1 –Дефекти функціонального тестування сайту



Рисунок 4.2.2 –Дефекти мобільного тестування сайту

# ДОДАТОК Б ЛІСТИНГ КОДУ ДЛЯ РЕАЛІЗАЦІЇ ДОМАШНЬОЇ СТОРІНКИ

{% extends 'base/base.html' %}  
{% load static %}  
  
{% block content %}  
<section id="home–section" class="hero">  
 <div class="home–slider owl–carousel">  
 <div class="slider–item" style="background–image: url({% static 'images/bg\_111.jpg' %});">  
 <div class="overlay"></div>  
 <div class="container">  
 <div class="row slider–text justify–content–center align–items–center" data–scrollax–parent="tuae">  
  
 <div class="col–md–12 ftco–animate text–center">  
 <h1 class="mb–2">Мы поставляем только лучшие детали</h1>  
 <h2 class="subheading mb–4">Всё по ходовой &amp; подвеска</h2>  
 <p><a href="#" class="btn btn–primary">Узнать детали</a></p>  
 </div>  
  
 </div>  
 </div>  
 </div>  
  
 <div class="slider–item" style="background–image: url({% static 'images/bg\_211.jpeg' %});">  
 <div class="overlay"></div>  
 <div class="container">  
 <div class="row slider–text justify–content–center align–items–center" data–scrollax–parent="tuae">  
  
 <div class="col–sm–12 ftco–animate text–center">  
 <h1 class="mb–2">100% Гарантия &amp; Товары от производителя</h1>  
 <h2 class="subheading mb–4">Всё по ходовой &amp; подвеска</h2>  
 <p><a href="#" class="btn btn–primary">Узнать детали</a></p>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
  
<section class="ftco–section">  
 <div class="container">  
 <div class="row no–gutters ftco–services">  
 <div class="col–md–3 text–center d–flex align–self–stretch ftco–animate">  
 <div class="media block–6 services mb–md–0 mb–4">  
 <div class="icon bg–color–1 active d–flex justify–content–center align–items–center mb–2">  
 <span class="flaticon–shipped"></span>  
 </div>  
 <div class="media–body">  
 <h3 class="heading">Бесплатная доставка</h3>  
 <span>On order over $100</span>  
 </div>  
 </div>  
 </div>  
 <div class="col–md–3 text–center d–flex align–self–stretch ftco–animate">  
 <div class="media block–6 services mb–md–0 mb–4">  
 <div class="icon bg–color–2 d–flex justify–content–center align–items–center mb–2">  
 <span class="flaticon–diet"></span>  
 </div>  
 <div class="media–body">  
 <h3 class="heading">Возврат</h3>  
 <span>Возможен возврат товара</span>  
 </div>  
 </div>  
 </div>  
 <div class="col–md–3 text–center d–flex align–self–stretch ftco–animate">  
 <div class="media block–6 services mb–md–0 mb–4">  
 <div class="icon bg–color–3 d–flex justify–content–center align–items–center mb–2">  
 <span class="flaticon–award"></span>  
 </div>  
 <div class="media–body">  
 <h3 class="heading">Супер качество</h3>  
 <span>Качество продуктов</span>  
 </div>  
 </div>  
 </div>  
 <div class="col–md–3 text–center d–flex align–self–stretch ftco–animate">  
 <div class="media block–6 services mb–md–0 mb–4">  
 <div class="icon bg–color–4 d–flex justify–content–center align–items–center mb–2">  
 <span class="flaticon–customer–service"></span>  
 </div>  
 <div class="media–body">  
 <h3 class="heading">Поддержка</h3>  
 <span>24/7 Поддержка</span>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
  
<section class="ftco–section ftco–category ftco–no–pt">  
 <div class="container">  
 <div class="row">  
 <div class="col–md–8">  
 <div class="row">  
 <div class="col–md–6 order–md–last align–items–stretch d–flex">  
 <div class="category–wrap–2 ftco–animate img align–self–stretch d–flex" style="background–image: url({% static 'images/category11.jpg' %});">  
 <div class="text text–center">  
 <h2>Запчасти</h2>  
 <p>Покупай качество</p>  
 <p><a href="#" class="btn btn–primary">Купить сейчас</a></p>  
 </div>  
 </div>  
 </div>  
 <div class="col–md–6">  
 <div class="category–wrap ftco–animate img mb–4 d–flex align–items–end" style="background–image: url({% static 'images/category–111.jpg' %});">  
 <div class="text px–3 py–1">  
 <h2 class="mb–0"><a href="#">Комплектующие</a></h2>  
 </div>  
 </div>  
 <div class="category–wrap ftco–animate img d–flex align–items–end" style="background–image: url({% static 'images/category–211.jpg' %});">  
 <div class="text px–3 py–1">  
 <h2 class="mb–0"><a href="#">Запчасти</a></h2>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div class="col–md–4">  
 <div class="category–wrap ftco–animate img mb–4 d–flex align–items–end" style="background–image: url({% static 'images/category–311.jpg' %});">  
 <div class="text px–3 py–1">  
 <h2 class="mb–0"><a href="#">Детали</a></h2>  
 </div>  
 </div>  
 <div class="category–wrap ftco–animate img d–flex align–items–end" style="background–image: url({% static 'images/category–411.jpg' %});">  
 <div class="text px–3 py–1">  
 <h2 class="mb–0"><a href="#">Аксесуары</a></h2>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
  
<section class="ftco–section">  
 <div class="container">  
 <div class="row justify–content–center mb–3 pb–3">  
 <div class="col–md–12 heading–section text–center ftco–animate">  
 <span class="subheading">Рекомендуемые товары</span>  
 <h2 class="mb–4">Наша продукция</h2>  
 <p>Доставка со всего мира, только оригинал</p>  
 </div>  
 </div>  
 </div>  
 <div class="container">  
 <div class="row">  
 {% if products %}  
 {% for product in products %}  
 <div class="col–md–6 col–lg–3 ftco–animate">  
 <div class="product">  
 <a href="{% url 'product–details' product.id %}" class="img–prod"><img class="img–fluid" src="{{ product.photo.url }}" alt="Colorlib Template">  
 <div class="overlay"></div>  
 </a>  
 <div class="text py–3 pb–4 px–3 text–center">  
 <h3><a href="{% url 'product–details' product.id %}">{{ product.name }}</a></h3>  
 <div class="d–flex">  
 <div class="pricing">  
 <p class="price"><span class="price–sale">${{ product.price }}</span></p>  
 </div>  
 </div>  
 <div class="bottom–area d–flex px–3">  
 <div class="m–auto d–flex">  
 <a href="#" class="add–to–cart d–flex justify–content–center align–items–center text–center">  
 <span><i class="ion–ios–menu"></i></span>  
 </a>  
 <a href="{% url 'cart' %}" class="buy–now d–flex justify–content–center align–items–center mx–1">  
 <span><i class="ion–ios–cart"></i></span>  
 </a>  
 <a href="#" class="heart d–flex justify–content–center align–items–center ">  
 <span><i class="ion–ios–heart"></i></span>  
 </a>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 {% else %}  
 <h2 style="margin: auto">Простите! Товар не доступен!</h2>  
 {% endif %}  
 </div>  
 </div>  
</section>  
  
<section class="ftco–section img" style="background–image: url({% static 'images/bg3111.jpg' %});">  
 <div class="container">  
 <div class="row justify–content–end">  
 <div class="col–md–6 heading–section ftco–animate deal–of–the–day ftco–animate">  
 <span class="subheading">Лучшая цена для тебя</span>  
 <h2 class="mb–4">Товар дня</h2>  
 <p>Доставка со всего мира, только оригинал</p>  
 <h3><a href="#">Омыватель стекла</a></h3>  
 <span class="price">$10 <a href="#">сейчас $5 за литр</a></span>  
 <div id="timer" class="d–flex mt–5">  
 <div class="time" id="days"></div>  
 <div class="time pl–3" id="hours"></div>  
 <div class="time pl–3" id="minutes"></div>  
 <div class="time pl–3" id="seconds"></div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
  
<section class="ftco–section testimony–section">  
 <div class="container">  
 <div class="row justify–content–center mb–5 pb–3">  
 <div class="col–md–7 heading–section ftco–animate text–center">  
 <span class="subheading">Отзывы</span>  
 <h2 class="mb–4">Наш довольный клиент говорит</h2>  
 <p>Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in</p>  
 </div>  
 </div>  
 <div class="row ftco–animate">  
 <div class="col–md–12">  
 <div class="carousel–testimony owl–carousel">  
 <div class="item">  
 <div class="testimony–wrap p–4 pb–5">  
 <div class="user–img mb–5" style="background–image: url({% static 'images/person\_1.jpg' %})">  
 <span class="quote d–flex align–items–center justify–content–center">  
 <i class="icon–quote–left"></i>  
 </span>  
 </div>  
 <div class="text text–center">  
 <p class="mb–5 pl–4 line">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>  
 <p class="name">Garreth Smith</p>  
 <span class="position">Marketing Manager</span>  
 </div>  
 </div>  
 </div>  
 <div class="item">  
 <div class="testimony–wrap p–4 pb–5">  
 <div class="user–img mb–5" style="background–image: url({% static 'images/person\_2.jpg' %})">  
 <span class="quote d–flex align–items–center justify–content–center">  
 <i class="icon–quote–left"></i>  
 </span>  
 </div>  
 <div class="text text–center">  
 <p class="mb–5 pl–4 line">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>  
 <p class="name">Garreth Smith</p>  
 <span class="position">Interface Designer</span>  
 </div>  
 </div>  
 </div>  
 <div class="item">  
 <div class="testimony–wrap p–4 pb–5">  
 <div class="user–img mb–5" style="background–image: url({% static 'images/person\_3.jpg' %})">  
 <span class="quote d–flex align–items–center justify–content–center">  
 <i class="icon–quote–left"></i>  
 </span>  
 </div>  
 <div class="text text–center">  
 <p class="mb–5 pl–4 line">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>  
 <p class="name">Garreth Smith</p>  
 <span class="position">UI Designer</span>  
 </div>  
 </div>  
 </div>  
 <div class="item">  
 <div class="testimony–wrap p–4 pb–5">  
 <div class="user–img mb–5" style="background–image: url({% static 'images/person\_1.jpg' %})">  
 <span class="quote d–flex align–items–center justify–content–center">  
 <i class="icon–quote–left"></i>  
 </span>  
 </div>  
 <div class="text text–center">  
 <p class="mb–5 pl–4 line">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>  
 <p class="name">Garreth Smith</p>  
 <span class="position">Web Developer</span>  
 </div>  
 </div>  
 </div>  
 <div class="item">  
 <div class="testimony–wrap p–4 pb–5">  
 <div class="user–img mb–5" style="background–image: url({% static 'images/person\_1.jpg' %})">  
 <span class="quote d–flex align–items–center justify–content–center">  
 <i class="icon–quote–left"></i>  
 </span>  
 </div>  
 <div class="text text–center">  
 <p class="mb–5 pl–4 line">Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.</p>  
 <p class="name">Garreth Smith</p>  
 <span class="position">System Analyst</span>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</section>  
  
<hr>  
  
<section class="ftco–section ftco–partner">  
 <div class="container">  
 <div class="row">  
 <div class="col–sm ftco–animate">  
 <a href="#" class="partner"><img src="{% static 'images/partner–1.png' %}" class="img–fluid" alt="Colorlib Template"></a>  
 </div>  
 <div class="col–sm ftco–animate">  
 <a href="#" class="partner"><img src="{% static 'images/partner–2.png' %}" class="img–fluid" alt="Colorlib Template"></a>  
 </div>  
 <div class="col–sm ftco–animate">  
 <a href="#" class="partner"><img src="{% static 'images/partner–3.png' %}" class="img–fluid" alt="Colorlib Template"></a>  
 </div>  
 <div class="col–sm ftco–animate">  
 <a href="#" class="partner"><img src="{% static 'images/partner–4.png' %}" class="img–fluid" alt="Colorlib Template"></a>  
 </div>  
 <div class="col–sm ftco–animate">  
 <a href="#" class="partner"><img src="{% static 'images/partner–5.png' %}" class="img–fluid" alt="Colorlib Template"></a>  
 </div>  
 </div>  
 </div>  
</section>  
  
<section class="ftco–section ftco–no–pt ftco–no–pb py–5 bg–light">  
 <div class="container py–4">  
 <div class="row d–flex justify–content–center py–5">  
 <div class="col–md–6">  
 <h2 style="font–size: 22px;" class="mb–0">Subcribe to our Newsletter</h2>  
 <span>Get e–mail updates about our latest shops and special offers</span>  
 </div>  
 <div class="col–md–6 d–flex align–items–center">  
 <form class="subscribe–form" method=POST>  
 <div class="form–group d–flex">  
 {% csrf\_token %}  
 {{ forms.email }}  
 <input type="submit" value="Subscribe" class="submit px–3">  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
</section>  
{% endblock %}

# ДОДАТОК В ФРАГМЕНТ ЛІСТИНГУ КОДУ ДЛЯ РЕАЛІЗАЦІЇ СТИЛЕЙ ПРОЕКТУ

/\*!  
 \* Bootstrap v4.2.1 (https://getbootstrap.com/)  
 \* Copyright 2011–2018 The Bootstrap Authors  
 \* Copyright 2011–2018 Twitter, Inc.  
 \* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)  
 \*/  
:root {  
 ––blue: #007bff;  
 ––indigo: #6610f2;  
 ––purple: #6f42c1;  
 ––pink: #e83e8c;  
 ––red: #dc3545;  
 ––orange: #fd7e14;  
 ––yellow: #ffc107;  
 ––green: #28a745;  
 ––teal: #20c997;  
 ––cyan: #17a2b8;  
 ––white: #fff;  
 ––gray: #6c757d;  
 ––gray–dark: #343a40;  
 ––primary: #007bff;  
 ––secondary: #6c757d;  
 ––success: #28a745;  
 ––info: #17a2b8;  
 ––warning: #ffc107;  
 ––danger: #dc3545;  
 ––light: #f8f9fa;  
 ––dark: #343a40;  
 ––breakpoint–xs: 0;  
 ––breakpoint–sm: 576px;  
 ––breakpoint–md: 768px;  
 ––breakpoint–lg: 992px;  
 ––breakpoint–xl: 1200px;  
 ––font–family–sans–serif: –apple–system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans–serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";  
 ––font–family–monospace: SFMono–Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace; }  
  
\*,  
\*::before,  
\*::after {  
 –webkit–box–sizing: border–box;  
 box–sizing: border–box; }  
  
html {  
 font–family: sans–serif;  
 line–height: 1.15;  
 –webkit–text–size–adjust: 100%;  
 –webkit–tap–highlight–color: rgba(0, 0, 0, 0); }  
  
article, aside, figcaption, figure, footer, header, hgroup, main, nav, section {  
 display: block; }  
  
body {  
 margin: 0;  
 font–family: –apple–system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans–serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";  
 font–size: 1rem;  
 font–weight: 400;  
 line–height: 1.5;  
 color: #212529;  
 text–align: left;  
 background–color: #fff; }  
  
[tabindex="–1"]:focus {  
 outline: 0 !important; }  
  
hr {  
 –webkit–box–sizing: content–box;  
 box–sizing: content–box;  
 height: 0;  
 overflow: visible; }  
  
h1, h2, h3, h4, h5, h6 {  
 margin–top: 0;  
 margin–bottom: 0.5rem; }  
  
p {  
 margin–top: 0;  
 margin–bottom: 1rem; }  
  
abbr[title],  
abbr[data–original–title] {  
 text–decoration: underline;  
 –webkit–text–decoration: underline dotted;  
 text–decoration: underline dotted;  
 cursor: help;  
 border–bottom: 0;  
 text–decoration–skip–ink: none; }  
  
address {  
 margin–bottom: 1rem;  
 font–style: normal;  
 line–height: inherit; }  
  
ol,  
ul,  
dl {  
 margin–top: 0;  
 margin–bottom: 1rem; }  
  
ol ol,  
ul ul,  
ol ul,  
ul ol {  
 margin–bottom: 0; }  
  
dt {  
 font–weight: 700; }  
  
dd {  
 margin–bottom: .5rem;  
 margin–left: 0; }  
  
blockquote {  
 margin: 0 0 1rem; }  
  
b,  
strong {  
 font–weight: bolder; }  
  
small {  
 font–size: 80%; }  
  
sub,  
sup {  
 position: relative;  
 font–size: 75%;  
 line–height: 0;  
 vertical–align: baseline; }  
  
sub {  
 bottom: –.25em; }  
  
sup {  
 top: –.5em; }  
  
a {  
 color: #007bff;  
 text–decoration: none;  
 background–color: transparent; }  
 a:hover {  
 color: #0056b3;  
 text–decoration: underline; }  
  
a:not([href]):not([tabindex]) {  
 color: inherit;  
 text–decoration: none; }  
 a:not([href]):not([tabindex]):hover, a:not([href]):not([tabindex]):focus {  
 color: inherit;  
 text–decoration: none; }  
 a:not([href]):not([tabindex]):focus {  
 outline: 0; }  
  
pre,  
code,  
kbd,  
samp {  
 font–family: SFMono–Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace;  
 font–size: 1em; }  
  
pre {  
 margin–top: 0;  
 margin–bottom: 1rem;  
 overflow: auto; }  
  
figure {  
 margin: 0 0 1rem; }  
  
img {  
 vertical–align: middle;  
 border–style: none; }  
  
svg {  
 overflow: hidden;  
 vertical–align: middle; }  
  
table {  
 border–collapse: collapse; }  
  
caption {  
 padding–top: 0.75rem;  
 padding–bottom: 0.75rem;  
 color: #6c757d;  
 text–align: left;  
 caption–side: bottom; }  
  
th {  
 text–align: inherit; }  
  
label {  
 display: inline–block;  
 margin–bottom: 0.5rem; }  
  
button {  
 border–radius: 0; }  
  
button:focus {  
 outline: 1px dotted;  
 outline: 5px auto –webkit–focus–ring–color; }  
  
input,  
button,  
select,  
optgroup,  
textarea {  
 margin: 0;  
 font–family: inherit;  
 font–size: inherit;  
 line–height: inherit; }  
  
button,  
input {  
 overflow: visible; }  
  
button,  
select {  
 text–transform: none; }  
  
button,  
[type="button"],  
[type="reset"],  
[type="submit"] {  
 –webkit–appearance: button; }  
  
button::–moz–focus–inner,  
[type="button"]::–moz–focus–inner,  
[type="reset"]::–moz–focus–inner,  
[type="submit"]::–moz–focus–inner {  
 padding: 0;  
 border–style: none; }  
  
input[type="radio"],  
input[type="checkbox"] {  
 –webkit–box–sizing: border–box;  
 box–sizing: border–box;  
 padding: 0; }  
  
input[type="date"],  
input[type="time"],  
input[type="datetime–local"],  
input[type="month"] {  
 –webkit–appearance: listbox; }  
  
textarea {  
 overflow: auto;  
 resize: vertical; }  
  
fieldset {  
 min–width: 0;  
 padding: 0;  
 margin: 0;  
 border: 0; }  
  
legend {  
 display: block;  
 width: 100%;  
 max–width: 100%;  
 padding: 0;  
 margin–bottom: .5rem;  
 font–size: 1.5rem;  
 line–height: inherit;  
 color: inherit;  
 white–space: normal; }  
  
progress {  
 vertical–align: baseline; }  
  
[type="number"]::–webkit–inner–spin–button,  
[type="number"]::–webkit–outer–spin–button {  
 height: auto; }  
  
[type="search"] {  
 outline–offset: –2px;  
 –webkit–appearance: none; }  
  
[type="search"]::–webkit–search–decoration {  
 –webkit–appearance: none; }  
  
::–webkit–file–upload–button {  
 font: inherit;  
 –webkit–appearance: button; }  
  
output {  
 display: inline–block; }  
  
summary {  
 display: list–item;  
 cursor: pointer; }  
  
template {  
 display: none; }  
  
[hidden] {  
 display: none !important; }  
  
h1, h2, h3, h4, h5, h6,  
.h1, .h2, .h3, .h4, .h5, .h6 {  
 margin–bottom: 0.5rem;  
 font–family: inherit;  
 font–weight: 500;  
 line–height: 1.2;  
 color: inherit; }  
  
h1, .h1 {  
 font–size: 2.5rem; }  
  
h2, .h2 {  
 font–size: 2rem; }  
  
h3, .h3 {  
 font–size: 1.75rem; }  
  
h4, .h4 {  
 font–size: 1.5rem; }  
  
h5, .h5 {  
 font–size: 1.25rem; }  
  
h6, .h6 {  
 font–size: 1rem; }  
  
.lead {  
 font–size: 1.25rem;  
 font–weight: 300; }  
  
.display–1 {  
 font–size: 6rem;  
 font–weight: 300;  
 line–height: 1.2; }  
  
.display–2 {  
 font–size: 5.5rem;  
 font–weight: 300;  
 line–height: 1.2; }  
  
.display–3 {  
 font–size: 4.5rem;  
 font–weight: 300;  
 line–height: 1.2; }  
  
.display–4 {  
 font–size: 3.5rem;  
 font–weight: 300;  
 line–height: 1.2; }