МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія управління відносинами з клієнтами автосервісу»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Проценко О.Б.**

Студент групи ІН.м – 02 — **Лаврик Д.С.**

СУМИ 2021

Сумський державний університет

(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*

Спеціальність *«122 -Комп'ютерні науки»*

Затверджую:

зав.кафедрою _____

_____

"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Лаврику Дмитру Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія управління відносинами з клієнтами автосервісу*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналітичний огляд CRM-систем. 2) Постановка задачі та*

формування завдань дослідження. 3) Аналіз існуючих методологій створення веб-додатку. 4) Вибір технології для створення інформаційної технології. 5) Розробка інформаційного та програмного забезпечення інформаційної технології управління відносинами з клієнтами автсервісу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
_____
_____
_____
_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналітичний огляд CRM-систем.		
2.	Постановка задачі та формування завдань дослідження		
3.	Аналіз існуючих методологій створення веб-додатку.		
4.	Вибір технології для створення інформаційної технології.		
5.	Розробка інформаційного та програмного забезпечення інформаційної технології управління відносинами з клієнтами автсервісу.		
6.	Оформлення пояснювальної записки до дипломної роботи		

Студент – дипломник _____

(підпис)

Керівник проекту _____

(підпис)

РЕФЕРАТ

Записка: 81 стор., 29 рис., 1 табл., 5 додатків, 18 джерел.

Об’єкт дослідження – інформаційні технології управління в бізнес-процесах.

Мета роботи – проектування та розробка інформаційної технології управління відносинами з клієнтами автосервісу, що допоможе покращити онлайн зв’язок «сервіс-користувач», за допомогою мови програмування Python та її фрейморків.

Метод дослідження – інформаційний аналіз, метод моделювання, методи розробки, що базуються на мові програмування Python.

Результати – проведено аналітичний огляд CRM-систем, уточнено особливості управління для автосервісу, розроблено алгоритм та програмно реалізовано веб-додаток CRM-системи для автосервісу з двофакторною аутентифікацією, а саме, реалізовано серверну частину на Python Flask та візуально обрамлено за допомогою Bootstrap Framework.

PYTHON FLASK, SQLALHEMY, CRM-СИСТЕМА, ВЕБ-ДОДАТОК, BOOTSTRAP FRAMEWORK, ДВОФАКТОРНА АУТЕНТИФІКАЦІЯ

ЗМІСТ

ВСТУП

На сьогоднішній день, конкуренція у сфері авто-бізнесу зростає дуже швидкими темпами, адже населення нашої планети продовжує стрімко рости. Майже кожен день у світі з’являються нові дилери продажу автомобілів, зростає кількість паркомісць, розвивається сфера обслуговування для автотранспортних засобів. Але щоб досягти успіху, бізнес-лідери виробили для себе певні кроки, які неохідно виконати, щоб досягти бажаного успіху:

- необхідність таргетованої реклами на сучасних платформах;
- достовірна інформація про послуги та сервіси, які надаються;
- підтримка існуючих клієнтів та залучення нових;
- якісне проведення операцій: дзвінки клієнтам, e-mail повідомлення тощо.

Саме даних кроків необхідно дотримуватися аби досягти успіху. Постає питання: де знайти необхідну інформацію про сервіс, який зацікавив кінцевого користувача? Відповідь – в інтернеті. Онлайн-сервіс дозволяє користувачам швидко та зручно обрати та здійснити оплату необхідної послуги у будь-який час, за рахунок чого послугообіг зростає. Онлайн-автосервіс дає змогу користувачеві завчасно замовити повну діагностику свого транспортного засобу, аби вподальшому уникнути ДТП. Як показує аналітика, ринок онлайн-автосервісів зростає з кожним роком, у тому числі у країнах що розвиваються, що є позитивним для розвитку їх економіки.

Актуальність роботи можливо пояснити з точки зору ведення сучасного бізнесу, адже кожна людина майже увесь свій день проводить за електронним пристроєм (комп’ютер, лептоп, смартфон тощо), а тому дзвінки для бронування вже не є такими актуальними.

Метою роботи є проектування та розробка веб-додатку для автосервісу, який допоможе покращити онлайн зв’язок «сервіс-користувач», за допомогою мови програмування Python.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 CRM-системи як основи сучасного бізнесу

Постійний розвиток інформаційних технологій сприяє розвитку сервісів, які забезпечують роботу за всіма спектрами інформаційних ресурсів, що відрізняються один від одного формами представлення та методами обробки складових їх інформаційних об’єктів. Найбільш відомими є сервіси із зберігання даних, електронної і голосової пошти, відео-сервісу, передачі повідомлень і блоків даних, організації і управління діалогом партнерів, надання з’єднань. Мережеві сервіси Інтернет змінюються, інтегруються і доповнюються та характеризують розвиток різних поколінь Інтернет. З ряду причин не можна ввести яку-небудь жорстку або визначену класифікацію.

Основна причина – унікальність кожного сервісу і одночасна невіддільність його від інших. Кожен сервіс характеризується властивостями, частина яких об’єднує його з однією групою сервісів, а інша частина з іншою групою [7]. Найбільш відповідним для класифікації сервісів Інтернет є поділ на сервіси інтерактивні, прямого звернення і відкладеного читання. Сервіси прямого звернення характерні тим, що інформація за запитом повертається негайно, проте від одержувача інформації не вимагається негайної реакції. Сервіси, де потрібна негайна реакція на отриману інформацію, тобто, отримувана інформація є, по суті справи, запитом, відносяться до інтерактивних сервісів, а сервіси відкладеного читання характеризуються відсутністю негайного повернення інформації та реакції. Для стандартних сервісів Інтернет характерні однакові принципи у побудові клієнтського і серверного програмного забезпечення, протоколів взаємодії, які реалізуються за міжнародними стандартами, а нестандартні сервіси, мають іноді несумісність з апаратним або програмним забезпеченням [1-3].

Незважаючи на значні інвестиції в системи управління відносинами з клієнтами (CRM), компанії продовжують відчувати втрати, а не прибуток. У той же час концепції «адаптивної поведінки» передових співробітників приділено мало уваги в літературі, пов'язаній з CRM-системами, в яких основними користувачами є співробітник. За допомогою CRM-систем можливо негайно отримати доступ до інформації про клієнтів і пропозицій послуг, що дозволить надавати персоналізовані послуги окремим клієнтам. Спираючись на теорію, розробляється модель адаптивної поведінки, керована системою CRM, яка пояснює, як CRM-системи полегшують роботу окремих співробітників, забезпечуючи адаптивну поведінку під час їх роботи. Крім того, формування поведінки CRM-систем після впровадження залежить від частого використання.

Перспективністю систем управління взаємовідносинами з клієнтами (CRM) є їх ефективність у документуванні актуальної для ринку інформації, управлінні контактами з клієнтами, а також сприянні орієнтованій на клієнта координації роботи співробітників у всій організації. Згідно зі звітом Gartner1, світовий ринок CRM-систем у 2017 році становив 39,5 мільярдів доларів США, що було найшвидшим зростаючим ринком програмного забезпечення та зайняв лідерство серед усіх ринків програмного забезпечення. Однак, незважаючи на значні інвестиції в CRM-системи, компанії продовжують відчувати втрати (наприклад, низьке використання та відсутність інтеграції з бізнес-процесами), а не прибуток [4].

У контексті обслуговування відповідальність за підтримання стосунків з клієнтами лягає на плечі передових співробітників, які відповідають за надання послуг і мають безпосередній контроль над обслуговуванням клієнтів. Впроваджуючи CRM-системи, компанії очікують, що їхні співробітники зможуть ефективніше задовольнити індивідуальних клієнтів, вести з ними персональні діалоги та персоналізувати продукти чи пропозиції послуг. Однак взаємодія між CRM-системами та співробітниками все ще виглядає слабкою.

Згідно зі статистикою за 2020 рік, ємність ринку сервісних послуг для легкового автотранспорту перевищила 100 млрд долларів, причому частка офіційних дилерів не перевищує 15%. Дані аналітики наголошують на прибутковості СТО у великих містах і показують потенціал ринку — автосервісам ще є куди зростати. Для ефективного керування автосервісом важливо гарно організувати бізнес-процеси та налагодити продажі, інакше при великому потоці клієнтів знижується якість сервісу та рівень їх задоволеності. Розберемося, як автоматизувати роботу СТО у 2021 році та що потрібно робити автосервісу для отримання стабільного прибутку.

1.2 Автоматизація та масштабування автобізнесу

Постійний потік клієнтів та використання кількох рекламних інструментів потребує впровадження CRM, що дозволяє автоматизувати внутрішні процеси компанії [13]. Інтеграція CRM - обов'язковий крок при масштабуванні автосервісу з гаражної СТО на повноцінну станцію технічного обслуговування. Автосервісу потрібна CRM-система, якщо потрібно:

- Облік об'ємного масиву неструктурованих даних;
- Організація роботи великого штату працівників;
- Автоматизація документообігу та продажу клієнтам;
- Аналітика та статистика в режимі реального часу;
- Підвищити лояльність клієнтів;
- Поліпшити конверсію продажу та допродажу;
- Систематизувати спілкування з клієнтами.

Впровадження CRM доцільне вже за наявності понад 3–4 співробітників у штаті компанії, а також при використанні кількох каналів реклами. Крім

того, CRM життєво необхідна для масштабування бізнесу, зокрема, якщо передбачається:

- Відкриття кількох філій СТО;
- Переїзд відділу продажів до офісу, віддаленого від автосервісу;
- Підключення до СТО автомийки, дітей-лінг-студії, магазину запчастин.;
- Ремонт, відновлення та продаж контрактних деталей для автосервісу.

Основною проблемою більшості СТО вважається складність у реєстрації та обслуговуванні клієнтів – менеджмент замовлень. Впровадження CRM дозволяє налаштувати вирву продажів, що розділяє взаємодію менеджера СТО з клієнтом на кілька етапів угоди [18]. Це дозволяє сегментувати процес реєстрації та обробки лідів, а також прописати чіткі посадові інструкції відділу продажу (або майстрів-приймачів) під кожний етап угоди. У функціонал продукту ззвичай включено:

- Планувальник завдань — система автоматично нагадає менеджерам зателефонувати клієнту, передати замовлення на склад, а також повідомить про завантаження боксів;
- Автозаповнення клієнтської картки — при появі нового ліду система переводить контакт до угоди та автоматично додає всі дані до CRM-картки у міру появи інформації;
- Система рекомендацій для менеджерів – алгоритм нагадає менеджерам запропонувати клієнту додаткові послуги, провести рекомендовані роботи (наприклад, регламентне ТО) для збільшення середнього чека.

Як результат — менеджери звільнені від системних процесів і весь час приділяють спілкуванню з клієнтами, що знижує кількість втрачених лідів і прискорює обробку угод.

При масштабуванні бізнесу дуже важливо отримати актуальні дані про бізнес-процеси компанії: завантаження боксів СТО, товарний залишок та заповнення складу, рівень навантаження менеджерів та автомеханіків. Аналітика в режимі онлайн дозволяє визначити проблемні ділянки бізнесу - виявити вузькі зони у вирві продажів, проаналізувати причини втрачених угод та незадоволеності клієнтів. Комплексна аналітика дозволяє отримати повний пакет даних про поточний стан бізнесу та запобігти зниженню конверсії та лояльності клієнтської бази [6].

CRM для автосервісу поєднує всі канали звернення клієнтів та дозволяє автоматизувати розсилки. Комунікативні можливості зазвичай представлені наступним функціоналом:

- Вбудований модуль телефонії — для СТО передбачена безкоштовна АТС із можливістю інтеграції кол-трекінгу, підключення багатоканального номера та SMS-сервісу для розсилок;
- Єдиний інтерфейс для комунікації — ліди з сайту, соцмереж та месенджерів відображаються в одному вікні, що не дозволяє менеджерам упустити угоду та спрощує комунікацію з клієнтами;
- Генератор повторного продажу — пакет роботів дозволяє налаштувати автоматичний обдзвон і розсилку за заданими сценаріями, що сприяє повторним операціям СТО та збільшенню лояльності клієнтів.

Пряма взаємодія з клієнтами та автозаповнення клієнтської бази дозволяє обробляти всі ліди, що надходять, і збільшує конверсію автосервісу. Згідно зі статистикою дослідницької агенції Nucleus Research, впровадження CRM для СТО дозволяє збільшити ефективність підприємства в середньому на 15%.

1.3 Постановка задачі

Головною метою роботи є розробка інформаційної вебсистеми CRM для автосервісу на Python Flask. Основними функціями даної системи є:

- оформлення заявки на ремонт автотранспорту;
- можливість перегляду замовлень користувачів та керуваннями ними;
- можливість доступу до системи не тільки з настільних комп’ютерів чи лептопу, а й з мобільних пристроїв;
- можливість реєстрації постійних клієнтів;
- перегляд власних замовлень користувачами.

Перед створенням даного сервісу необхідно виокремити основні частини розробки та досліджень, а саме:

- попередній аналіз існуючих додатків;
- дизайн інтерфейсу системи;
- розробка головної сторінки системи;
- впровадження бази данних до системи;
- розробка сторінки реєстрації та входу користувачів;
- розробка системи створення замовлень;
- впровадження системи адміністрування користувачів;
- реалізація двофакторної аутентифікації користувачів при реєстрації;
- тестування створеної системи.

2 СУЧАСНА МЕТОДОЛОГІЯ СТВОРЕННЯ ВЕБ-ДОДАТКУ

2.1 Фреймворк Flask

Flask — це мікровеб-фреймворк, написаний на Python. Його класифікують як мікрофреймворк, оскільки він не вимагає особливих інструментів чи бібліотек. У ньому немає рівня абстракції бази даних, перевірки форми чи будь-яких інших компонентів, де вже існуючі сторонні бібліотеки надають загальні функції. Однак Flask підтримує розширення, які можуть додавати функції програми так, ніби вони були реалізовані в самому Flask. Існують розширення для об’єктно-реляційних картографів, перевірки форм, обробки завантаження, різних технологій відкритої аутентифікації та кількох загальних інструментів, пов’язаних з фреймворком [7].

Підтримується установка за допомогою пакетного менеджера PyPI, версія 1.0 сумісна з Python 2.7, Python 3.3 та вище. Творець та основний автор — австрійський програміст Армін Ронахер, розпочав роботу над проектом у 2010 році [14].

Flask має модульний дизайн, який дозволяє адаптувати його для виконання багатьох завдань. "З коробки" розробник отримує наступні функції:

- вбудований сервер та дебаггер;
- шаблони Jinja2;
- підтримка безпечних кукіс;
- WSGI 1.0;
- Unicode;
- можливість підключення до будь-якої ORM.

Створювався фреймворк для підтримки розробників веб-застосунків, які отримали можливість вибирати розширення за смаком [9].

Обирати Flask для розробки необхідно, якщо ви зацікавлені в отриманні професійного досвіду та можливості навчання або хочете отримати більше контролю над тим, які компоненти використовуються (наприклад, які бази даних ви хочете використовувати та як взаємодіяти з ними). Flask найкращий вибір, тому що він досить простий у роботі. Flask підходить тим, кому потрібно більше кастомізації [18].

Для того, щоб зробити висновки про популярність цього фреймворку, було вирішено створити порівняльну таблицю, розташовану нижче. Flask був порівняний з такими фреймворками як – Django та CherryPy та ін. за такими критеріями:

- Група фреймворків: Full stack або Micro-framework;
- Зірки Github: загальна кількість зірок проекту, виставлених користувачам;
- Релізи Github: кількість релізів кожного проекту, що відображає активність роботи над проектом та його зрілість;
- Fork-і Github: кількість копій кожного проекту, що показує популярність використання проекту у власних роботах;
- Запитання Stack-overflow: кількість питань, заданих за певною темою.
- Вакансії: кількість вакансій, пов'язаних із технологіями або IT компетенціями.

Сама порівняльна таблиця:

Таблиця 2.1 – Порівняння фреймворків Python

Назва фреймворку	Группа фреймворку	Зірки на GitHub	Релізи на GitHub	Fork на GitHub	Питання на Stack-overflow	Вакансії
Flask	Micro	48747	256	13400	618	12
Dash	Micro	1237	216	25400	158	45
Django	Full	46728	268	20400	217030	47
CherryPy	Micro	1120	126	279	1300	0
Zope	Full	194	227	81	3	0

"Мікро" у фреймворку відноситься не тільки до простоти і невеликого розміру бази, але це також може означати той факт, що він не пропонує вам багато проектних рішень. Незважаючи на те, що Flask використовує щось подібне як шаблонизатор, ми не будемо приймати подібні рішення для вашого сховища даних або інших частин. Тим не менш, для нас термін "мікро" не означає, що вся реалізація має вписуватися в один файл.

Одним із проектних рішень у Flask є те, що прості завдання мають бути простими; вони не повинні займати багато коду, і це не повинно обмежувати вас. Тому ми зробили кілька варіантів дизайну, деякі люди можуть вважати це дивним і навіть дивним. Наприклад, Flask використовує локальні треди всередині об'єктів, так що ви не повинні передавати об'єкти в межах одного запиту від функції до функції, залишаючись у безпечному треді. Хоча це і дуже простий підхід, який дозволяє заощадити час, таке рішення може викликати деякі проблеми для занадто великих додатків, оскільки зміни в цих локальних об'єктах-тредах можуть відбутися будь-де в цьому треді. Для того, щоб вирішити ці проблеми, ми не стали приховувати від вас локальні треди-об'єкти, натомість ми охоплюємо їх і надаємо вам багато інструментів, щоб зробити роботу з ними настільки приємною, наскільки це можливо.

У Flask багато речей попередньо налаштовані, на основі загальної базової конфігурації. Наприклад, шаблони та статичні файли збережені в підкаталогах у межах вихідного дерева. Ви також можете змінити це, але зазвичай це не потрібно [8].

Основна причина чому Flask називається "мікрофреймворком" - це ідея зберегти ядро простим, але розширюваним. У ньому немає абстрактного рівня бази даних, немає валідації форм або всього, що вже є в інших бібліотеках. Однак, Flask підтримує розширення, які можуть додати необхідну функціональність і імплементує їх так, ніби вони вже були спочатку вбудовані. В даний час вже є розширення: форми валідації, підтримка завантаження файлів, різні технології автентифікації та багато інших.

Виходячи з виконаної роботи можна зробити висновок про те, що даний фреймворк має достатню велику спільноту і багато розробників, що працюють на Flask.

2.2 Фреймворк Bootstrap 5

Bootstrap – це відкритий та безкоштовний HTML, CSS та JS фреймворк, який використовується веб-розробниками для швидкої верстки адаптивних дизайнів сайтів та веб-додатків.

Фреймворк Bootstrap використовується в усьому світі не тільки незалежними розробниками, але іноді цілими компаніями. На Bootstrap створено багато різних сайтів.Основна сфера його застосування – це фронтенд розробка сайтів та інтерфейсів адмінок. Серед аналогічних систем (Foundation, UIkit, Semantic UI, InK та ін) фреймворк Bootstrap є найпопулярнішим [16].

Велику популярність Bootstrap пов'язано з тим, що він дозволяє верстати сайти в кілька разів швидше, ніж на чистому CSS і JavaScript. А в нашому світі час – це дуже цінний ресурс. Ще один аспект – доступність. Вона зводиться до того, що надає можливість навіть початківцю веб-розробнику (без глибоких знань і достатньої практики) створювати досить якісні макети [11].

Фреймворк Bootstrap – це набір набір CSS та JavaScript файлів. Щоб його використовувати ці файли, необхідно просто підключити до сторінки. Після цього вам стануть доступні інструменти фреймворку: колонкова система (сітка Bootstrap), класи і компоненти.

Bootstrap складається з:

- інструментів для створення макета (обгорткових контейнерів, потужної системи сіток, гнучких медіа-об'єктів, адаптивних утилітних класів);
- класів для стилізації базового контенту: тексту, зображень, коду, таблиць та figure;

- готових компонентів: кнопок, форм, горизонтальних і вертикальних навігаційних панелей, слайдерів, списків, що випадають, акордеонів, модальних вікон, підказок та ін.;
- утилітних класів для вирішення традиційних завдань, що найбільш часто виникають перед веб-розробниками: вирівнювання тексту, відображення та приховування елементів, завдання кольору, фону, margin і padding відступів, і т.д.

Переваги Bootstrap при його використанні для frontend розробки сайтів:

- висока швидкість створення якісної адаптивної верстки навіть початківцями веб-розробниками (досягається це завдяки використанню готових класів та компонентів, створених професіоналами);
- кросбраузерність та кросплатформенність (коректне відображення та робота сайту у всіх підтримуваних цим фреймворком браузерах та операційних системах);
- наявність великої кількості готових, добре продуманих компонентів, протестованих величезним співтовариством веб-розробників на різних пристроях;
- можливість налаштування під свій проект, досягається це за допомогою зміни SCSS змінних та використання міксинів (можна змінити кількість колонок, кольору, радіус заокруглень, відступи між колонками тощо);
- низький поріг входження; для роботи з фреймворком не обов'язково мати «глибокі» знання з HTML, CSS, JavaScript та jQuery (досить знати лише основи цих технологій);
- однорідність дизайну та його узгодженість між різними компонентами (у Bootstrap всі компоненти виконані в єдиному стилі);
- наявність величезної кількості спільнот та навчальних матеріалів; при бажанні це допоможе не тільки добре розібратися у фреймворку, але і знайти відповіді практично на будь-які питання, що виникають у вас [5].

Фреймворк Bootstrap – це проект із відкритим вихідним кодом, доступним на Github. Він має ліцензію MIT. Це означає, що його можна безкоштовно використовувати як у відкритих, так і комерційних проектах. Bootstrap дуже гарний інструмент для будови адаптивних сайтів, але його не завжди доцільно використовувати. Наприклад, не має сенсу використовувати його для:

- створення фронтендів проектів із унікальним дизайном;
- розробки проектів, у яких замовник готовий платити за проект на «чистому» CSS та JavaScript (у більшості випадків така розробка здійснюється в команді, в якій кожен її учасник виконує певний набір функцій);
- верстки особистих проектів, якщо у вас є достатньо часу і ваш рівень знань з HTML, CSS і JavaScript є достатнім, щоб це здійснити.

Bootstrap, як і більшість подібних фреймворків, має недоліки. Серед них можна відзначити такі:

- більший розмір кінцевих css і js-файлів проекту, ніж вони вийшли, якби ми все це створювали самостійно (це пов'язано з тим, що стилі фреймворку та його js-код містять універсальний код (на всі випадки життя), а за фактом конкретного проекту з цього може знадобитися лише частина);
- складність використання Bootstrap для верстки сайтів із унікальним дизайном, т.к. технологія в цьому випадку супроводжуватиметься значним переписуванням його коду і простим налаштуванням Bootstrap змінних тут вже не обійтися.

2.3 База даних SQLite

SQLite — компактна реляційна база даних, що вбудовується. Вихідний код бібліотеки передано до суспільного надбання. Є суто реляційною базою даних.

Слово "вбудований" означає, що SQLite не використовує парадигму клієнт-сервер. Тобто. двигун SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якою програма компонується і двигун стає складовою програми [15]. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку та спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси та дані) у єдиному стандартному файлі на тому комп'ютері, на якому виконується програма. Простота реалізації досягається рахунок того, що перед початком виконання транзакції запису весь файл, що зберігає базу даних, блокується; ACID - функції досягаються зокрема рахунок створення файлу журналу [11].

Декілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис до бази можна здійснити лише в тому випадку, якщо жодних інших запитів на даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу. Можна також ввести тайм-аут операцій. Тоді підключення, зіткнувшись із зайнятістю бази даних (БД), чекатиме N секунду перш, ніж відвалитися з помилкою SQLITE_BUSY [17].

У комплекті поставки йде також функціональна клієнтська частина у вигляді файлу sqlite3, що виконується, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, що дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі двигуна можна використовувати SQLite як на вбудовуваних системах, так і на виділених машинах з гігабайтними масивами даних.

Формат файлу бази даних є крос-платформним, що дозволяє без проблем використовувати ту саму базу на декількох операційних системах. Також є

можливість зберігання бази в пам'яті, без її запису на диск. Цей параметр використовується за промовчанням для консольної утиліти sqlite3, якщо не вказано ім'я файлу.

Особливості SQLite

- SQLite не вимагає окремого процесу сервера або системи для роботи (без сервера);
- SQLite постачається з нульовою конфігурацією, що означає відсутність необхідності в налаштуванні або адмініструванні;
- Повна база даних SQLite зберігається в одному крос-платформному диску;
- SQLite дуже маленький і легкий, менше 400KiB повністю налаштований або менше 250KiB з додатковими функціями, опущеними;
- SQLite є автономним, що означає відсутність зовнішніх залежностей;
- SQLite транзакції повністю сумісні з ACID, забезпечуючи безпечний доступ до кількох процесів або потоків;
- SQLite підтримує більшість функцій мови запитів, знайдених у стандарті SQL92 (SQL2);
- SQLite написаний на ANSI-C і надає простий та простий у використанні API.
-

SQLite використовує динамічне типування даних. Це означає, що тип стовпця не визначає тип значення, що зберігається в цьому полі запису. Тобто. у будь-який стовпець можна занести будь-яке значення [11].

Тип стовпця визначає як порівнювати значення (потрібно їх привести до єдиного типу при порівнянні, скажімо, всередині індексу). Але не зобов'язує заносити значення саме такого типу в стовпець.

Припустимо, ми оголосили стовпець як "A INTEGER". SQLite дозволяє занести в цей стовпець значення будь-якого типу (999, "abc", "123", 678.525).

Якщо значення, що вставляється, - не ціле, то SQLite намагається привести його до цілого. Тобто. рядок «123» перетвориться на ціле 123, а решта значень запишуться «як є».

Можливі типи полів: NULL, INTEGER, REAL, TEXT, BLOB.

2.4 Двофакторна аутентифікація

Двофакторна аутентифікація стала необхідною і важливою частиною безпеки облікового запису, тому що через зростання та розвиток кіберзлочинності паролі більше не є надійними. Після серії гучних витоків даних за останні десять років багато комбінацій імені користувача та пароля вже доступні для продажу в Dark Web [12]. Організації більше не можуть довіряти тому, що просте знання правильного пароля є достатньо надійним, щоб дозволити користувачеві отримати доступ до облікового запису.

Крім того, існують людські фактори та тенденції, які сприяють тому, що паролі стають вразливою стратегією аутентифікації:

- Слабкі паролі

Безпека часто виявляється поза увагою звичайного користувача. Складається помилкове враження, що як приватна особа вони навряд чи стануть метою кіберзлочинності. Це призводить до того, що користувачі обирають слабкі паролі, такі як "password123" або "123456".

- Втома паролів

У міру того, як кількість власників мобільних пристроїв продовжує зростати, а компанії продовжують зусилля з цифрової трансформації, кількість онлайн-акаунтів, які кожен користувач повинен пам'ятати, також зростатиме. Це збільшення кількості імен користувачів та паролів призводить до того, що користувачі повторюють той самий пароль у кількох облікових записах.

Двофакторна аутентифікація — це метод ідентифікації користувача в будь-якому сервісі (як правило, в Інтернеті) за допомогою запиту

аутентифікаційних даних двох різних типів, що забезпечує двошаровий, а отже, ефективніший захист облікового запису від несанкціонованого доступу. На практиці це зазвичай виглядає так: перший рубіж - це логін і пароль, другий - спеціальний код, що надходить SMS або електронною поштою. Рідше другий шар захисту запитує спеціальний USB-ключ або біометричні дані користувача. Загалом, суть підходу дуже проста: щоб кудись потрапити, потрібно двічі підтвердити той факт, що ви це ви, причому за допомогою двох «ключів», одним з яких ви володієте, а інший тримаєте в пам'яті [12].

Двофакторна аутентифікація у поєднанні з традиційною системою паролів забезпечує більш надійний захист, ніж використання облікових даних для входу. Саме завдяки наявності двофакторної аутентифікації багатьох атак за останні місяці можна було б запобігти.

Рішення необхідне для компаній будь-якого розміру, оскільки щодня працівники здійснюють вхід на кілька платформ. Насамперед двофакторну автентифікацію необхідно забезпечити для облікових записів з правами адміністратора та тих, хто має доступ до конфіденційної інформації. Це є потужним кроком до запобігання крадіжці даних та можливим фінансовим втратам [12].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Реалізація бази даних на SQLAlchemy for Python

SQLAlchemy - це набір інструментів Python SQL, який заснований на базі SQLite тадозволить використовувати в додатку повну міць та гнучкість SQL. Він надає повний набір певних шаблонів для збереження структури, призначених для ефективного та високопродуктивного доступу до баз даних. SQLAlchemy є чудовим рішенням для реалізації БД при написанні простого веб-додатку за допомогою фреймворку Python Flask (див. Додаток А).

База даних буде складатися з чотирьох таблиць, в які буде зберігатися уся інформація (Рис. 3.1). Це таблиці:

1) User;
2) Service;
3) Auto;
4) CurrentWork.

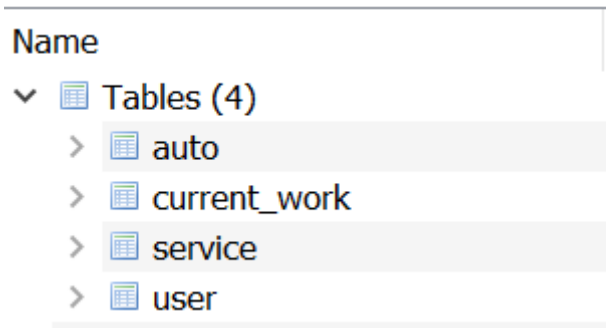

Рисунок 3.1 – Структура БД

Таблиця «User» представляє собою набір даних про користувачів сайту. У даній таблиці зберігається інформація про ім’я та прізвище юзера, його/її мобільний телефон та електронну пошту. У таблиці присутнє посилання на таблицю «Auto», в якій зберігається інформаціє про автомобіль даного

користувача. Також, за користувачем зберігається інформація про пароль – він хешується, аби зберегти приватність користувача, - та інформація про секретний ключ, який необхідний для подальшої авторизації користувача, використовуючи двофакторну аутентифікацію. Додавання користувачів до таблиці відбувається шляхом реєстрації на сайті. Таблиця складається з колонок, які представлені нижче:

1) id;
2) first_name;
3) last_name;
4) phone;
5) email;
6) password;
7) auto;
8) otp_secret.

Таблиця «Service» зберігає в собі інформацію про послуги, які пропонує СТО. Дана таблиця заповнюється власноруч адміністратором. В ній знаходиться наступна інформація про послуги: унікальний ідентификатор для кожної послуги, назва послуги, ціна на неї, стислий/повний опис послуги. Структура таблиці представлена нижче:

1) id;
2) service_title;
3) service_price;
4) service_comment.

Таблиця «Auto» містить інформацію про автомобілі користувачів. Інформація про авто вноситься виключно адміністратором після перевірки достовірності даних. До даної таблиці входить унікальний ідентифікатор авто та зовнішній ключ «user_id», який посилається на володара автомобіля, тобто користувача сайту. Також, представлена основна інформація про транспортний засіб (ТЗ): ВІН-код, марка та модель ТЗ, рік випуску, потужність

двигуна та тип коробки передач, а також – посилання на таблицю «Service» аби відслідковувати роботи з ТЗ. Структура таблиці:

1) id;
2) vin;
3) brand;
4) model;
5) year;
6) engine_volume;
7) trans_type;
8) user_id;
9) inservice.

Таблиця «CurrentWork» містить у собі інформацю про замовлення користувачів, які були виконані на сайті. До неї записується наступна інформація: унікальний ідентифікатор замовлення, назва замовлення, ім’я та прізвище замовника, телефонний номер, ціна на послугу, дата створення та дата закриття замовлення, зовнішній ключ «user_id», який містить унікальний ідентифікатор користувача (якщо такий зареєстрований на сайті), та зовнішній ключ «auto_id», який передає унікальний ідентифікатор автомобіля. Структура таблиці виглядає наступним чином:

1) id;
2) title;
3) first_name;
4) last_name;
5) phone;
6) price;
7) date_open;
8) data_close;
9) user_id;
10) auto_id.

3.2 Реалізація клієнтської частини за допомогою Bootstrap

Для реалізації клієнтської частини було обрано Bootstrap Framework, який включає в себе шаблони для швидкої верстки адаптивного дизайну сайту. Під час розробки, використовувався Bootstrap версії 5.1.

Під час розробки клієнтської частини програми, було використано метод, який дозволяє з усього веб-коду виділити ту ділянку, яка буде зустрічатися на усіх вкладках веб-додатку, аби зменшити частину коду, який повторюється (див. Додаток Б). Для усіх шаблонів, представлених у додатку використовувався метод «extend», який дозволяв вставляти новостворені сторінки до вже існуючого шаблону.

Навігаційне меню та футер сторінки було додано до основної частини коду, адже ця частина писутня на кожній сторінці веб-додатку. Також, було додано певні стилі до деяких частин веб-сторінок, аби покращити їх вигляд та розташування на сторінці. Елемент «Головна» представляє собою головну сторінку веб-додатку. Він містить основну інформацію про сервіс, відгуки користувачів сервісу та приклади виконаних робіт, які представляються у вигляді фотогалереї (Рис. 3.2, Рис. 3.3).

Для звичайного користувача та для адміністратори воно відрізняється.

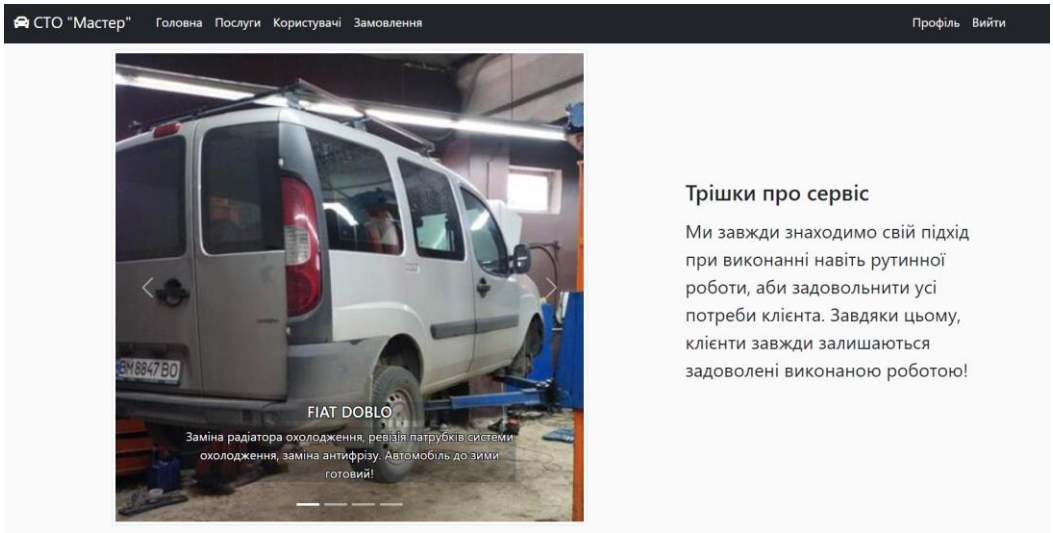

Рисунок 3.2 – Елемент навігаційного меню «Головна» адміністратора

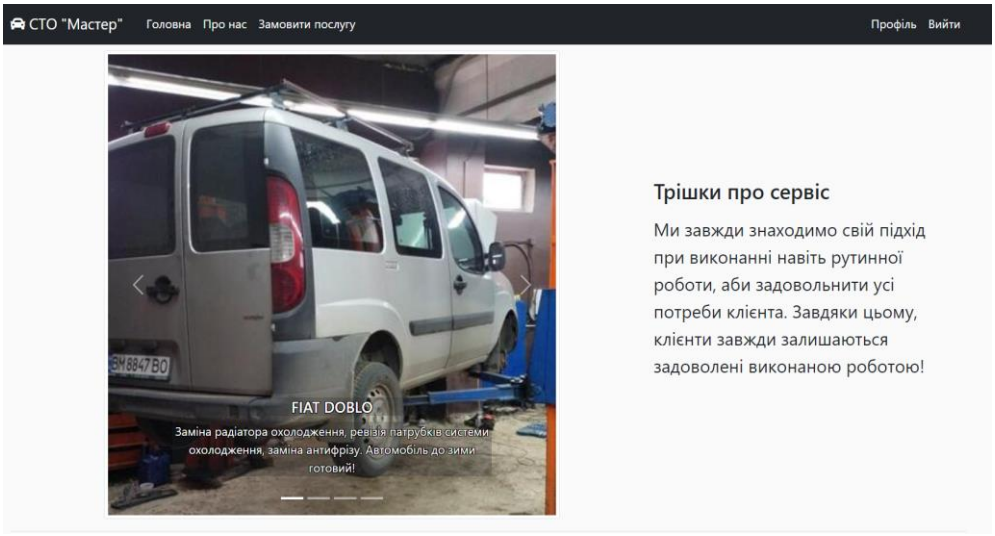

Рисунок 3.3 – Елемент навігаційного меню «Головна» користувача

Елемент «Про нас» містить контактну інформацію СТО (Рис. 3.4), а саме:

- телефонний номер;
- поштову скриньку;
- адресу;
- розташування на карті Гугл.

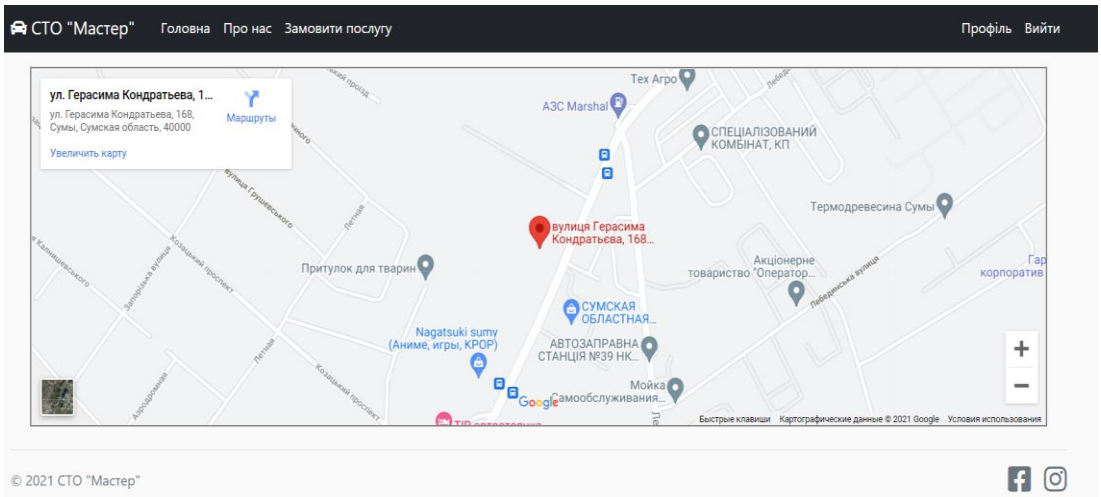

Рисунок 3.4 – Елемент навігаційного меню «Про нас»

Елемент «Замовити послугу» дозволяє виконати замовлення у сервісі та надіслати свої контактні дані до адміністратора, який надалі зв’яжеться з

замовником (Рис 3.5). Якщо користувач послугами вже зареєстрований, то деякі поля будуть заповнюватися автоматично (Рис. 3.6).

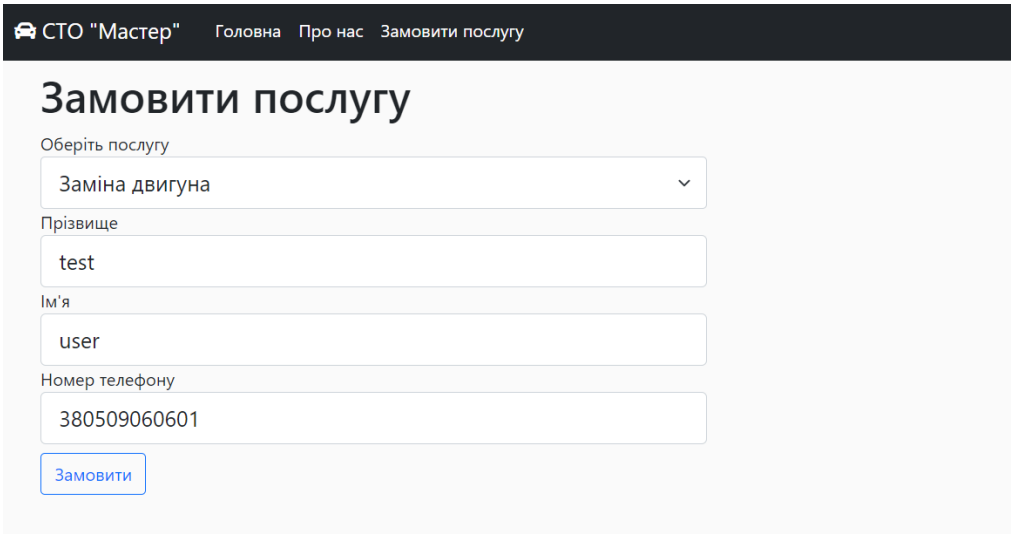

Рисунок 3.5 – Елемент навігаційного меню «Замовити послугу» якщо користувач зареєстрований

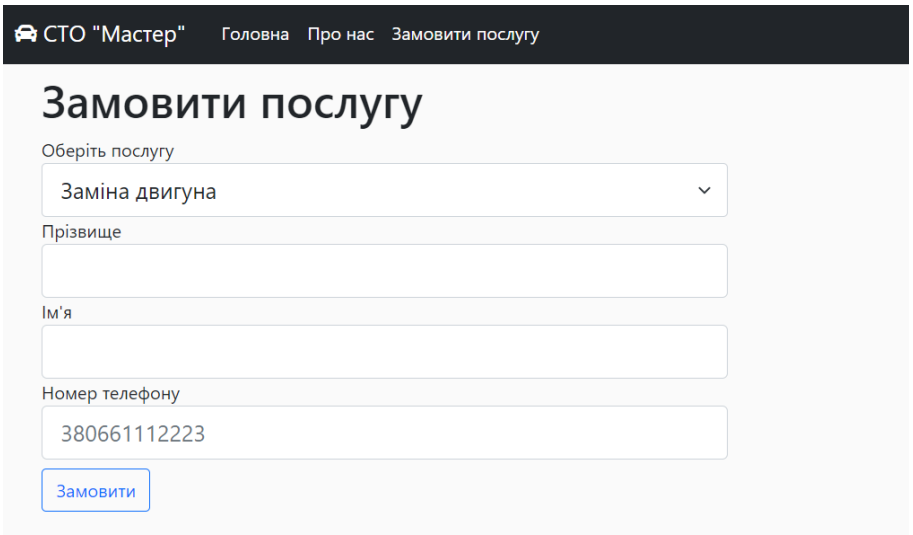

Рисунок 3.6 – Елемент навігаційного меню «Замовити послугу» якщо користувач незареєстрований

Також, у шапці присутні елементи аутентифікації такі як «Вхід» та «Реєстрація» (Рис. 3.7, Рис. 3.8). Якщо людина вже зареєстрована, то вона з легкістю зможе увійти на сайт, увівши електронну пошту, пароль та ОТП

ключ, який генерується мобільним додатком. Потрапити до свого профілю можливо завдяки елементу «Профіль» у шапці меню (Рис. 3.9). На даній вкладці представлена основна інформація про користувача, така як ім’я та прізвище, мобільний номер телефону та електронна пошта. Присутні дві колонки: «Мої замовлення» - на ній відображаються поточні замовлення користувача, та «Моє авто», де можливо знайти інформацію про авто користувача. Останні дві колонки відсутні у користувача з роллю «Адміністратор».

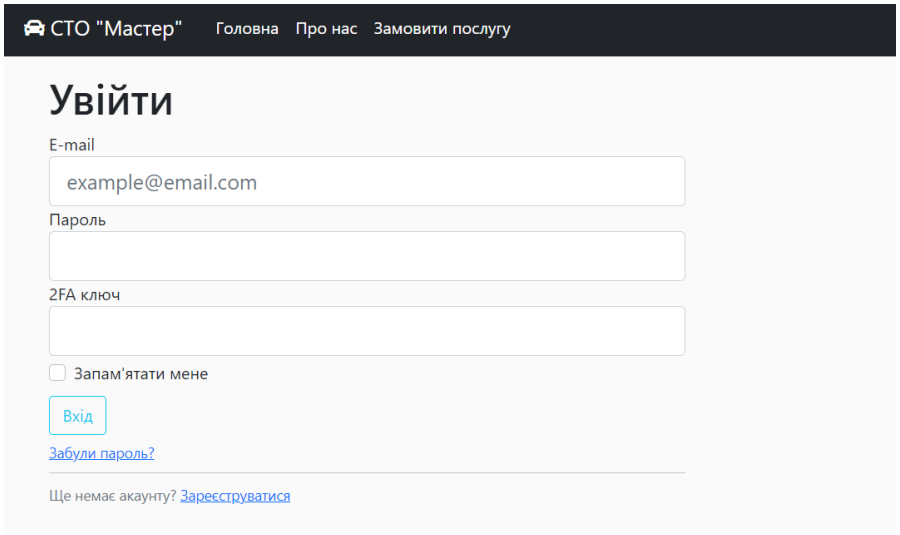

Рисунок 3.7 – Елемент навігаційного меню «Вхід»

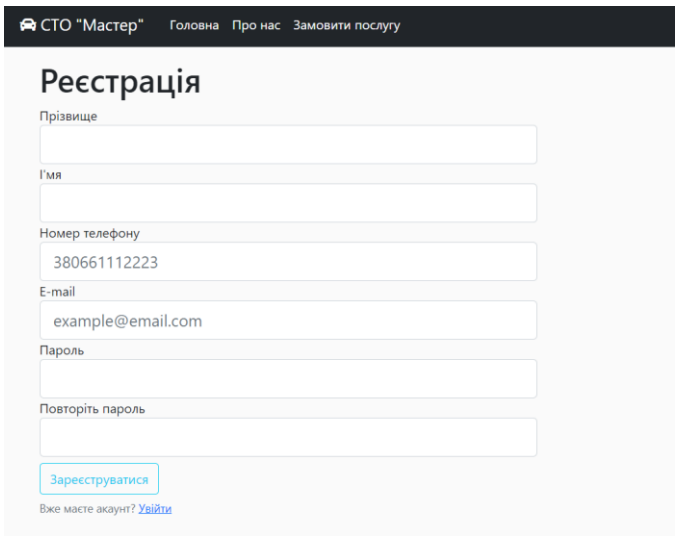

Рисунок 3.8 – Елемент навігаційного меню «Реєстрація»

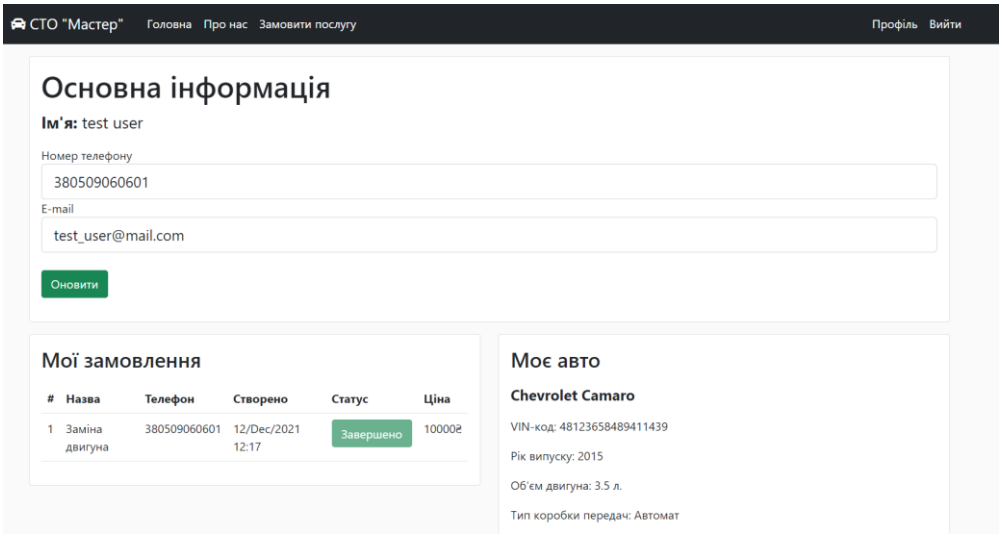

Рисунок 3.9 – Елемент навігаційного меню «Профіль»

Елемент «Послуги» містить у собі інформацію про послуги, які доступні у сервісі та має функцію додавання нових послуг до існуючого списку. Зовнішній вигляд кожної послуги однаковий (Рис. 3.10). Якщо на сторінці відображено більше 15 послуг, то реалізується перемикання сторінок.

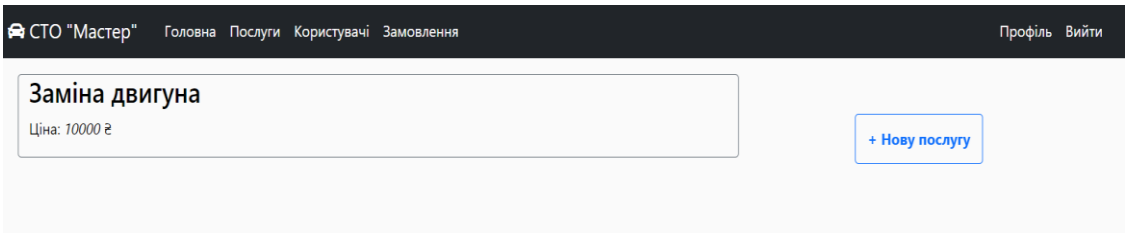

Рисунок 3.10 – Елемент навігаційного меню «Послуги»

Елемент «Користувачі» містить інформацію про зареєстрованих користувачів та ТЗ, які їм належать (Рис. 3.11). На даній сторінці реалізується перемикання сторінок, якщо відображено більше 15 користувачів.

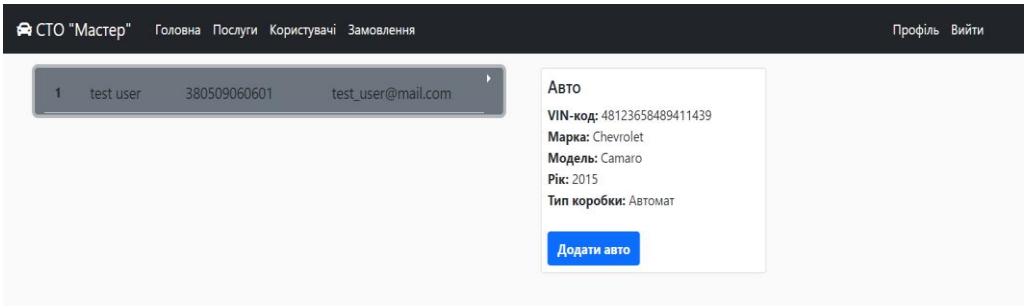

Рисунок 3.11 – Елемент навігаційного меню «Користувачі»

Елемент «Замовлення» містить інформацію про усі послуги, які надаються або надавалися та інформацію про них (Рис. 3.12). На даній сторінці також реалізовано перемикання сторінок, якщо відображено більше 20 послуг.

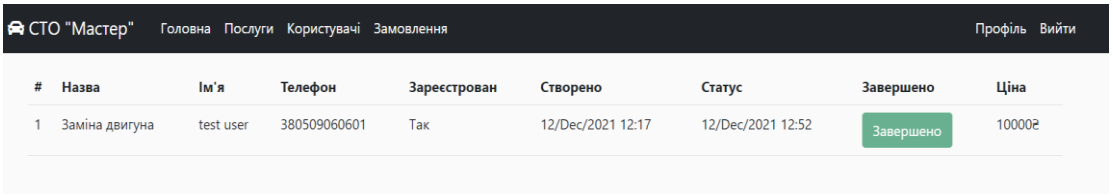

Рисунок 3.12 – Елемент навігаційного меню «Замовлення»

3.3 Реалізація серверної частини за допомогою Python Flask

Реалізація серверної частини вимагала підбору якісного фреймворку, який би мав можливість, завдяки невеликому набору даних, реалізувати водночас стабільну та гарну роботи веб-додатку. Таким фреймворком, у моєму випадку, виявився Flask. Під час реалізація на веб-серверної частини, використовувалася версія додатку 2.0.1.

Усю роботу над серверною частину можна розподілити на три етапи:

1) завантаження необхідних бібліотек;
2) налаштування веб-шляхів та створення користувацьких форм;
3) тюнення коду.

На першому етапі були встановленні такі бібліотеки, як: Flask – основна бібліотека, яка використовується веб-додатком; Flask_SQLAlchemy –

бібліотека бази даних; Flask_Bcrypt – бібліотека, необхідна для хешування приватних даних, які знаходяться на сервері; Flask_Login – бібліотека, необхідна для роботи з користувачами, які знаходяться у системі. Також. для роботи з користувацькими формами було застосовано Flask_WTF бібліотеку, яка дозволяє легко створювати форми на веб-сторінці та відправляти їх на сервер для обробки, та деякі основні бібліотеки Python: datetime для роботи з датами, os – для роботи з операційною системою, onetimepass – для роботи з ОТП паролями, pyqrcode – для створення QR-кодів.

На другому етапі розробки більша увага приділялася реалізації веб-серверу, його структурі та функціям. Основним файлом. в якому знаходився корінь усієї програми, називається __init__.py, де було створено образ нашого веб-додатку та додано необхідні конфігурації для корректної роботи програми (див. Додаток В), а саме:

- app.config['SECRET_KEY'] – секретний ключ програми (він знадобиться нам при надсиланні «POST» та «GET» запросів);
- app.config[‘SQLALCHEMY_DATABASE_URI’] – налаштування бази даних.

Далі, створювалися моделі таблиць бази даних, які були висвітлені у пункті 3.1, повну структуру яких можливо знайти у додатку. З точки зору Python, така таблиця – це наслідуємий клас від класу Model. Створення колонок у такій таблиці представляється наступним чином:

column_name = Column(тип даних колонки, додаткові параметри, такі як «нульове» або «ненульове» значення, унікальне значення і т.п.).

Для моделі «User» були зроблені додаткові функції, які в подальшому допоможуть реалізувати двофакторну аутентифікацію. Функція __init__() перевіряє, чи заповнена у користувача колонка «otp_secret», яка відповідає за генерацію секретного ключа, і якщо вона не заповнена, то відповідно генерує секретний ключ. Функція get_totp_uri() повертає URI аутентифікації. Це використовується для передачі секретного ключа та додаткової інформації

облікового запису на смартфон. Цей URI буде відображатися як QR-код, який потрібно відсканувати за допомогою телефону. Функція verify_totp() приймає токен аутентифікації як вхідні дані та перевіряє, використовуючи підтримку, надану пакетом onetimepass.

Основна частина, де відбувається «сілкування» між ділянками сервера, записано у файл routes.py (див. Додаток Г). Бібліотека Flask має декоратор @app.route(відносне посилання на сторінку, методи – GET, POST), завдяки якому можна оголошувати маршрути і вони працюватимуть, як очікувалося. Під декоратором завжді створюється функція, яка буде відповідати за рераутинг до необхідної сторінки. Така функціяя буде завжди повертати шаблон веб-сторінки за допомгою функції render_template(), аргументами якої є назва файлу та додаткові параметри: форма, яка надсилається, посилання на таблицю БД та інші параметри, які необхідно передати разом з шаблоном сторінки.

Після створення усіх необхідних маршрутів, було вирішено додати окремий файл, у якому записуються усі форми зв’язку між користувачами та сервером, - forms.py (див. Додаток Д). Усі класи, які були створені під даним файлом були створені як наслідуємі класи від FlaskForm, який імпортується з Flask_WTF бібілотекою. осна частина створення форм заключаєтсья у правильно підібраному типі данних, які будуть вводитися користувачем веб-додатку.

Структура такого коду дуже проста:

- створюється клас, який відповідає за певну форму на сайті
- далі створюються необхідні поля, наприклад

field_name = FieldType(аргумент_1 = назва поля, яка буде використовуватися HTML як лейбл, аргумент_2 = валідатори для даного поля)

- та створюється кнопка з типом SubmitField, яка відповідає за надсилання форми до серверу.

На третьому етапі відбувалося тюнення створеного коду, а саме:

- додавання валідацій до форм «Реєстрація» та «Оновлення Профілю»;
- реалізація двофакторної аутентифікаіцї за допомгою бібліотек onetimepass та pyqrcode.

На даному етапі було завершено реалізацію веб-серверу та створення основної частини програми. Попереду залишилося лише тестування створеного веб-додатку.

3.4 Тестування веб-додатку

Тестування є одним з головних процесів після написання коду, адже під час розробки, можливо було недогледіти деякі помилки. На цьому етапі тестування буде проходити наступним чином:

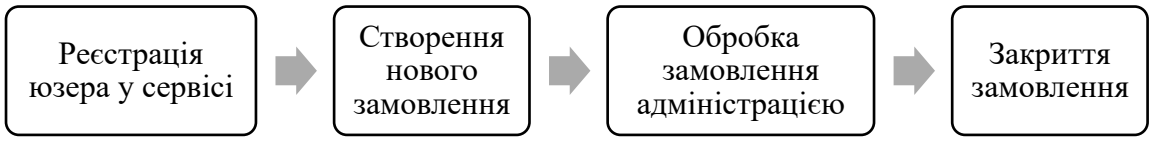

Рисунок 3.13 – Етапи тестування

Також, додатково протестуємо додавання нових видів послуг, оновлення даних користувача та додавання ТЗ користувачеві. Для початку, створимо профіль адміністратора та отримаємо секретний ключ аби підключити двофакторну аутентифікаіцю.

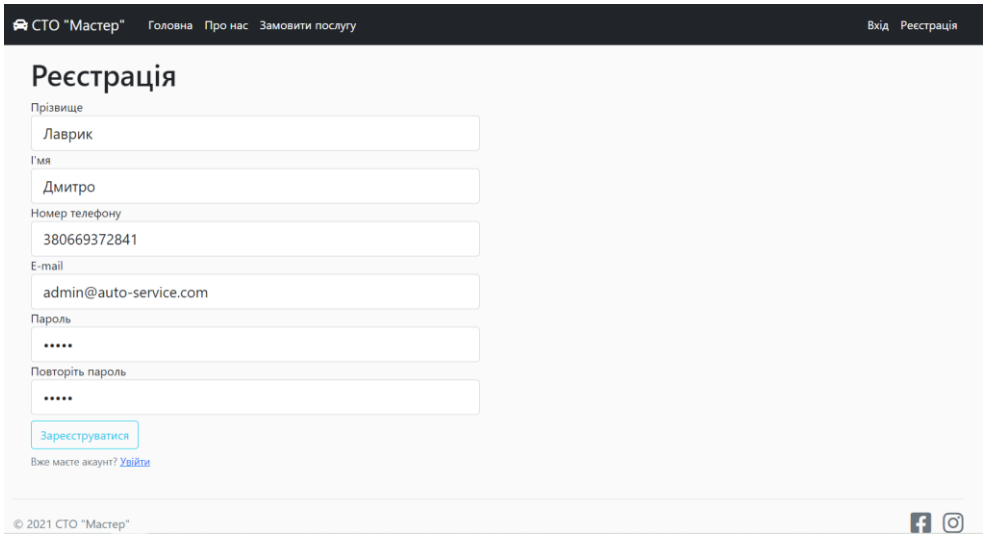

Рисунок 3.14 – Реєстрація користувача

Рисунок 3.15 – Налаштування двофакторної аутентифікації

Переходимо на сторінку входу та авторизуємося на сайті. При вірному заповненні полів «E-mail», «Пароль» та «2FA ключ», користувач зможе отримати доступ до ресурсу.

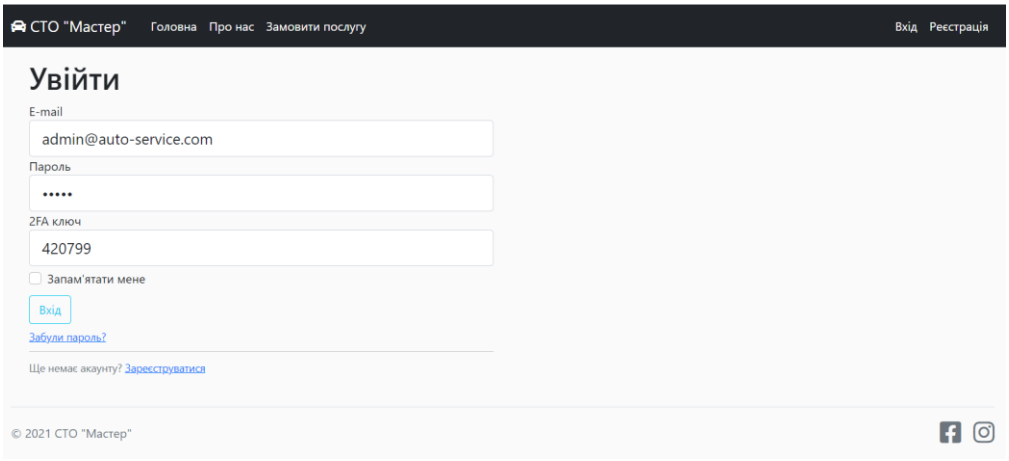

Рисунок 3.16 – Сторінка авторизації

Адміністратор успішно авторизувався. Перевіримо, що у нього доступні усі функції, а саме: створення нової послуги, доступ до підв’язки ТЗ до користувача та завершення замовлень. Попередньо, створимо ще одного користувача, який буде звичайним юзером, без прав адміністратора.

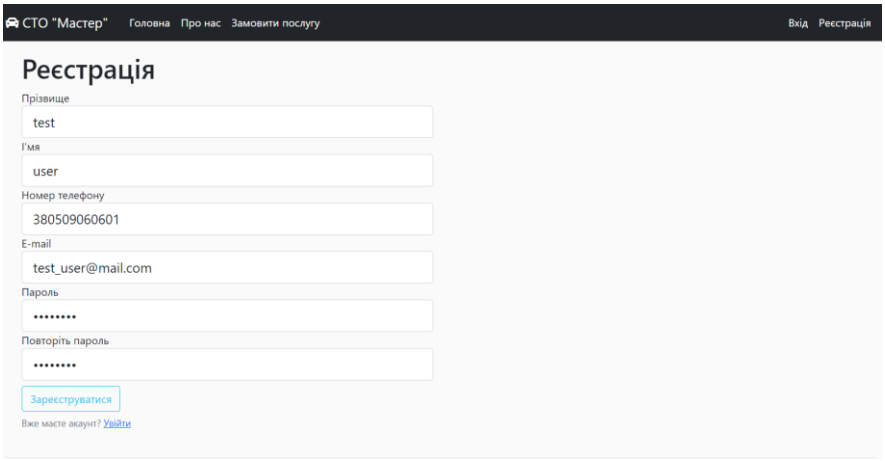

Рисунок 3.17 – Реєстрація звичайного користувача-клієнта

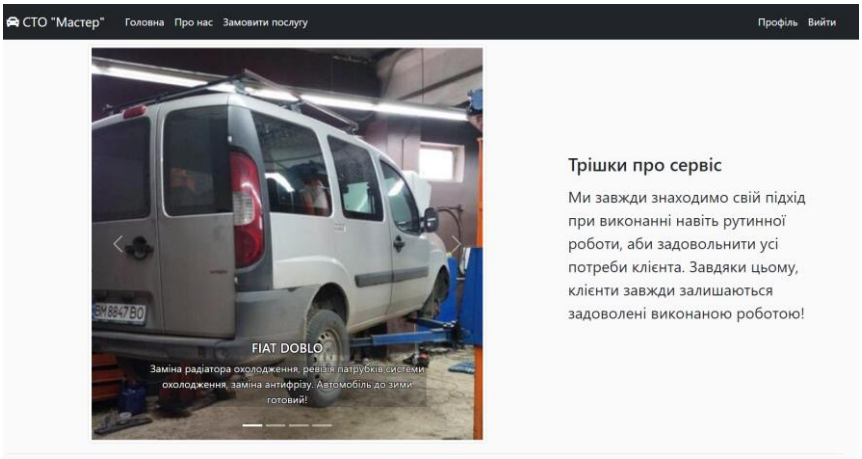

Рисунок 3.18 – Головна сторінка користувача

Тепер знову авторизуємося як адміністратор та приступимо до тестування основних функцій. Спершу створимо нову послугу, яка буде називатися «Заміна двигуна», ціна – 10000 грн., опис послуги – «Відбувається заміна старого двигуна на нову модель».

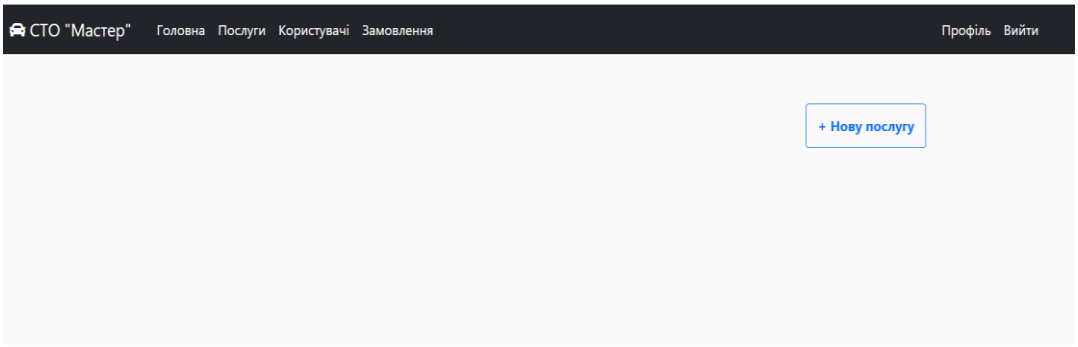

Рисунок 3.19 – Вкладка «Послуги» до створення нової послуги

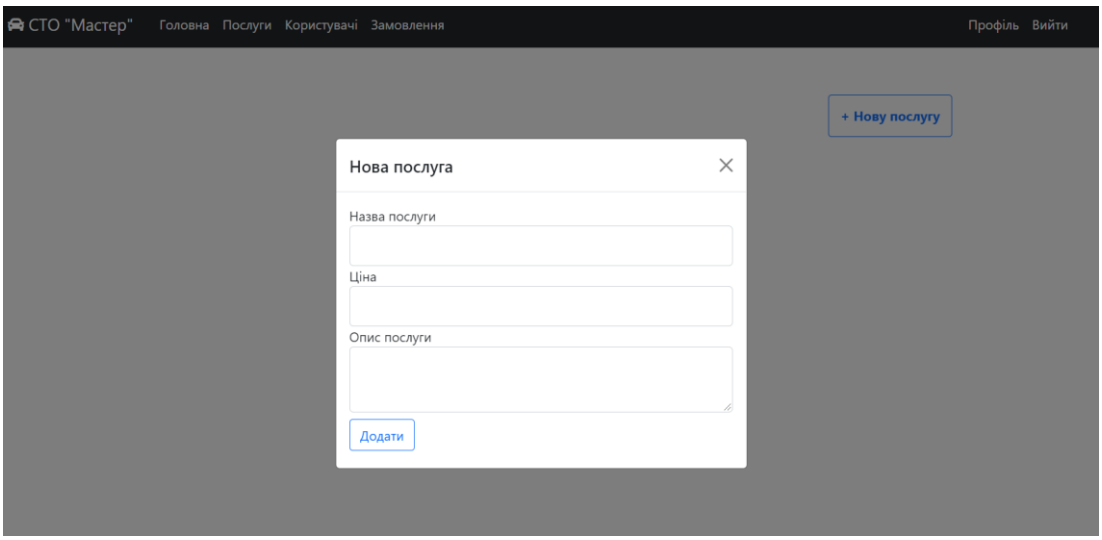

Рисунок 3.20 – Діалогове вікно для створення нової послуги

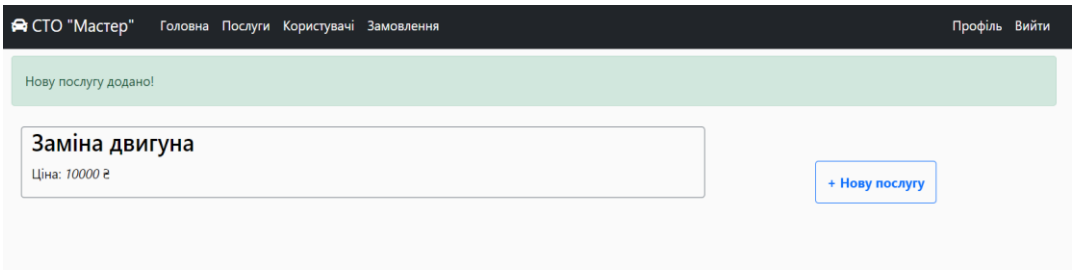

Рисунок 3.21 – Вкладка «Послуги» зі створеною послугою

Даний тест пройшов успішно. Тепер додамо користувачеві його ТЗ. Переходимо на вкладку «Користувачі». Обираємо необхідного юзера та натискаємо «Додати авто».

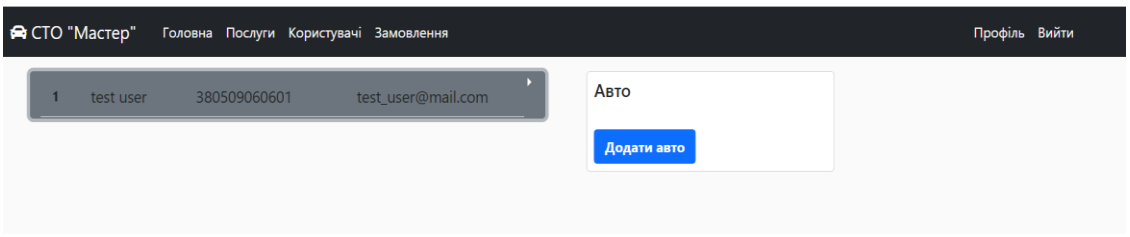

Рисунок 3.22 – Вкладка «Користувачі»

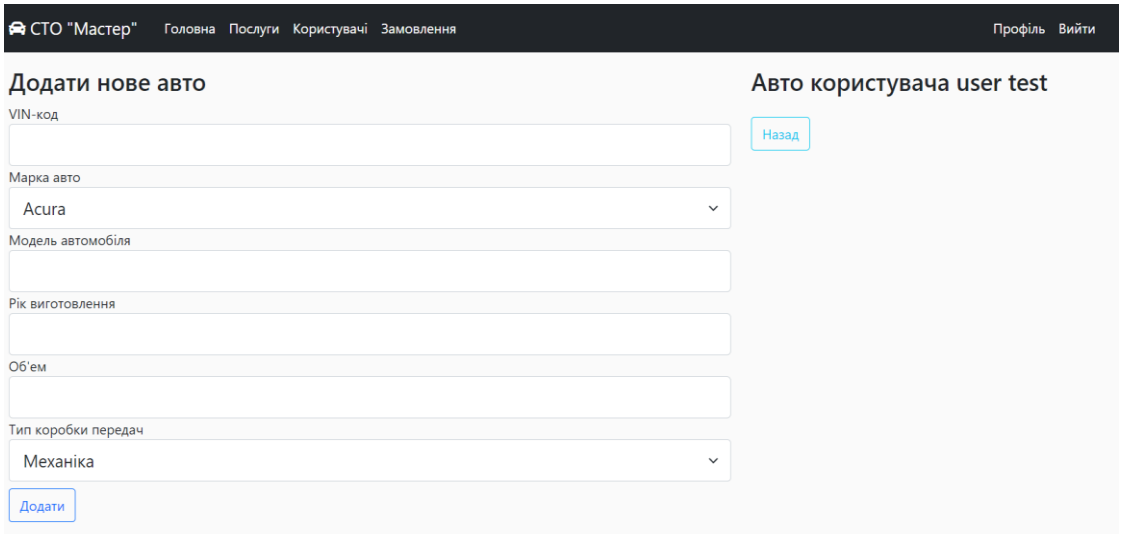

Рисунок 3.23 – Вигляд вкладки для прив’язки ТЗ користувачеві

Вводимо усі необхідні дані про автомобіль та натискає кнопку «Додати». На новій вкладці бачимо наступне повідомлення та інформацію про щойно доданий ТЗ.

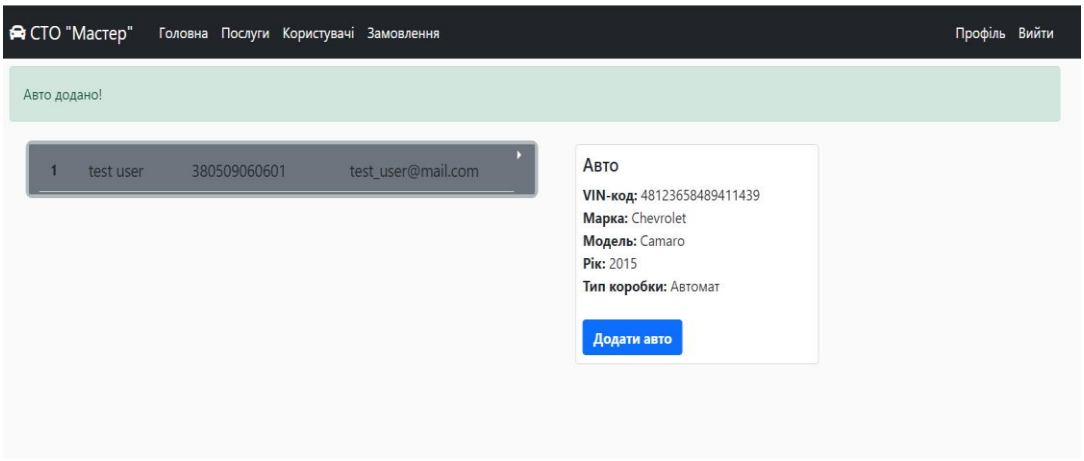

Рисунок 3.24 – Інформація про користувачі після прив’язки ТЗ

Перейдемо до зареєстрованого користувача та виконаємо замовлення послуги. Обираємо вкладку «Замовити послугу» та бачимо, що усі необхідні поля заповлені. Натискаємо «Замовити» та переходимо до свого профілю.

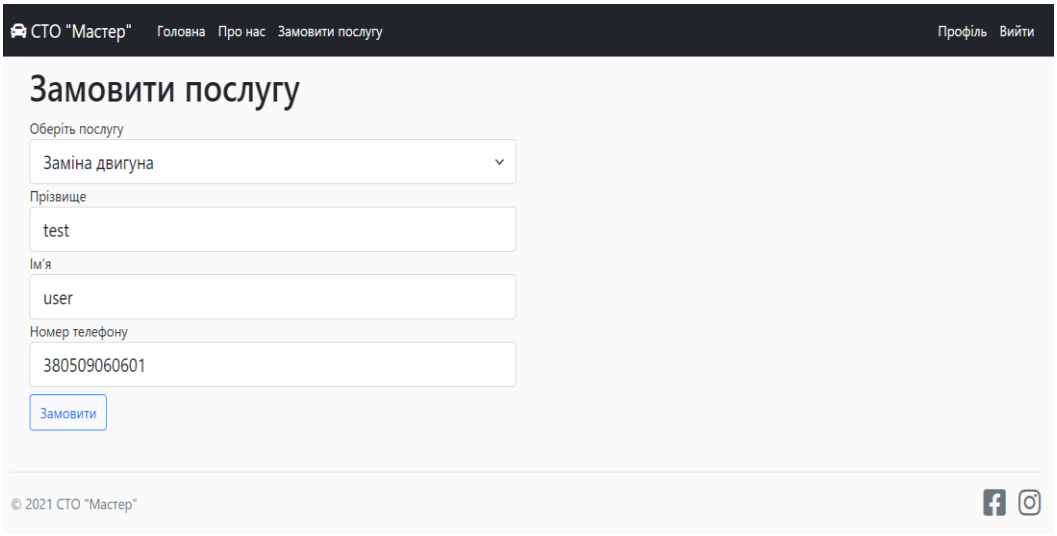

Рисунок 3.25 – Замовлення послуги на сайті

На сторінці профілю користувача присутнє щойно створене замовлення та, як можна бачити, інформація про авто користувача, яку вносив адміністратор сайту. Завершимо дане замовлення та перевіримо, чи оновилася інформація про замовлення у клієнта.

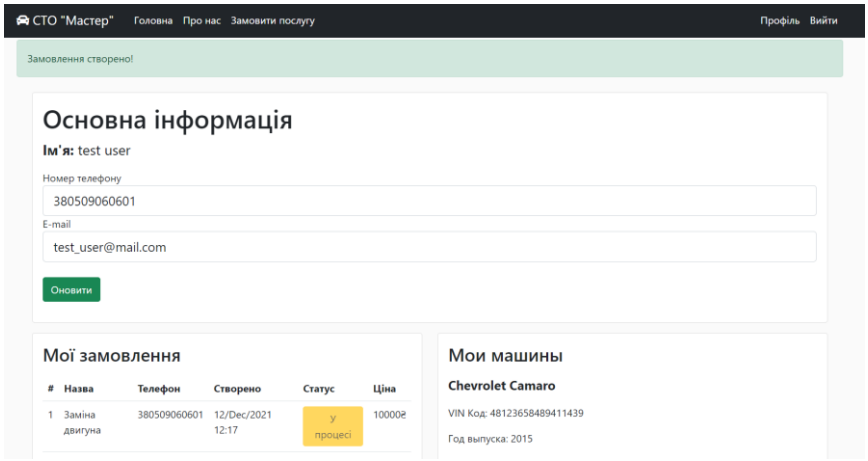

Рисунок 3.26 – Сторінка з інформацією користувача

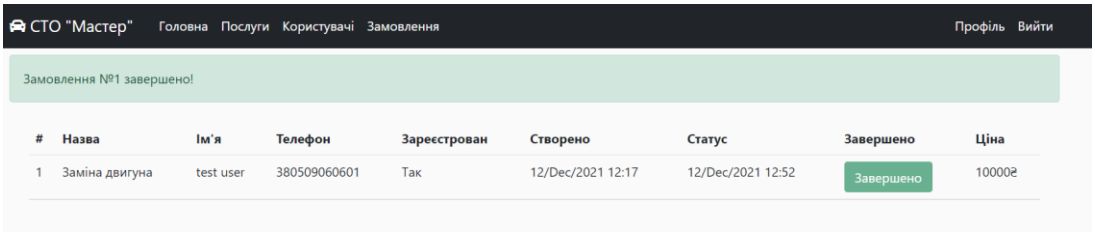

Рисунок 3.27 – Завершення замовлення

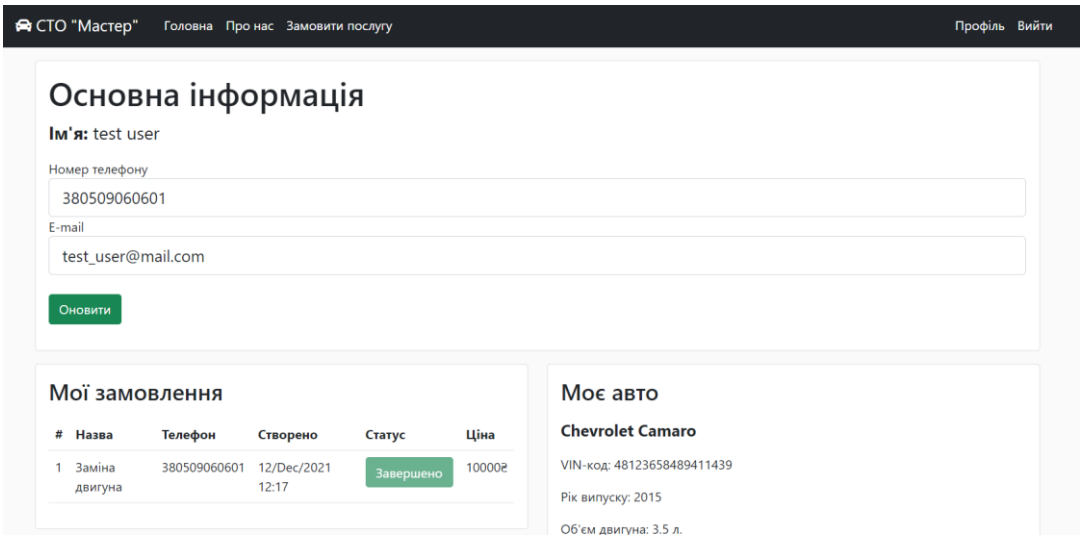

Рисунок 3.28 – Замовлення користувача завершено

Залишилося перевірити оновлення даних користувача. Для цього, спробуємо ввести значення електронної пошти та мобільного телефону адміністратора.

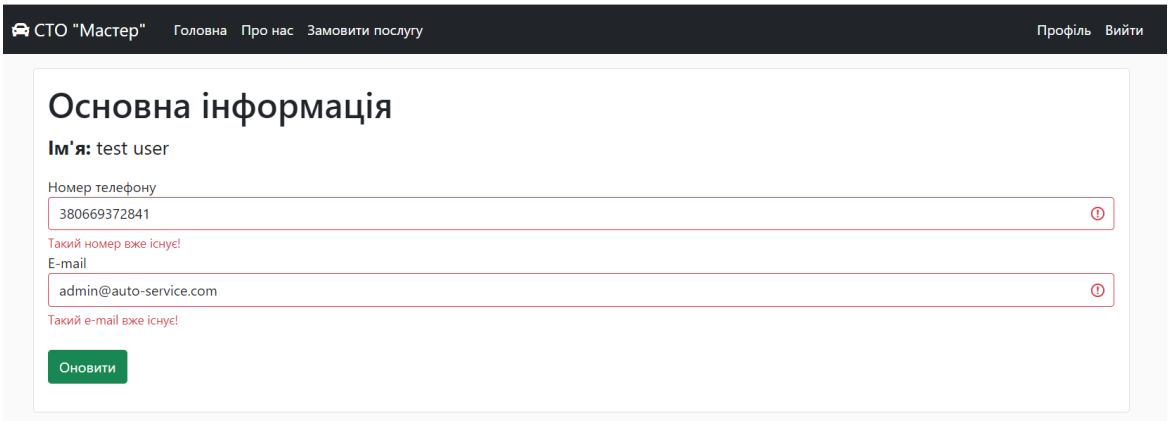

Рисунок 3.29 – Оновлення даних користувача

Змініти дані не вийшло, адже такі дані вже зареєстровані у системі. Так як, під час тестування не було знайдено жодніх порушень у роботі системи, можна вважати, що воно пройшло успішно.

ВИСНОВКИ

Узагальнюючи викладене у роботі, можна констатувати, що діяльність українського автобізнесу (управління бізнес-процесами, облсуговування клієнтів, ведення фінансів) тісно пов'язана з опрацюванням, відслідковуванням, зберіганням значних обсягів інформації, звітних та фінансових документів. У звязку з цим, ефективне управління автобізнесом неможливо здійснювати у сучасних умовах без спеціалізованих електронних систем та сервісів.

У магістерській роботі було здійснено аналітичний огляд наукових та навчальних ресурсів за тематикою особливостей автобізнесу та інформаційних систем, що застосовуються при управлінні у цьому бізнесі. Зокрема, було проаналізовано вже існуючі веб-додатки для автобізнесу та ті, що створені саме для CRM-систем.

У результаті виконання роботи розроблена інформаційна технологія управління відносинами з клієнтами автосервісу. Розроблена технології реалізована у веб-додатку, що функціонує наступним чином:

1) готова база даних, здатна зберігати та накопичувати клієнтську базу;
2) функції ефективної обробки замовлень користувачів;
3) ефективний зв'язок клієнта та адміністратора автосервісу.

Тестування веб-додатку здійснено успішно і розроблений веб-сервіс продемонстровано замовнику. Наразі проводяться налаштування для роботи розробленого веб-додатку на робочому місці замовника, для подальшого його застосування в реальному бізнес-процесі автосервісу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Handfield R. Shifts in buyer-seller relationships: A retrospective on. // Industrial Marketing Management, 2019, 83. – PP. 194-206.

2. Rayburn S. W., Badrinarayanan V., Anderson S. T. & Gupta A. Continuous techno-training and business-to-business salesperson success: How boosting techno-efficacy enhances sales effort and performance. // Journal of Business Research, 2021, 133. – PP. 66-78.

3. Gaspar D., Stouffer J. Mastering Flask Web Development: Build enterprise-grade, scalable Python web applications, 2nd Edition. – Packt Publishing Ltd, 2018. - 332 p.

4. Lee Y. & Cheon H. A Study on the Factors Affecting the User Intention of Omnichannel Shopping Based on Information Technology. // Proceedings of the 2019 5th International Conference on E-Business and Applications, 2019. – PP. 20-24.

5. Bootstrap. Introduction - URL: https://getbootstrap.com/docs/5.1/getting-started/introduction/

6. Chan J., Chung R., Huang J. Python API Development Fundamentals: Develop a full-stack web application with Python and Flask. – Packt Publishing Ltd, 2019. - 372 p.

7. Rodriguez M., Peterson R. M. & Krishnan V. Impact of CRM technology on sales process behaviors: empirical results from US, Europe, and Asia. // Journal of Business-to-Business Marketing, 2018, vol. 25(1). – PP. 1-10.

8. Chauhan N., Singh M., Verma A., Parasher A. & Budhiraja G. Implementation of database using python flask framework. International Journal of Engineering and Computer Science 8, 2019 - 24894-24899.

9. Fredstam M, Johansson G. Comparing database management systems with SQLAlchemy : A quantitative study on database management systems

[Electronic resource]. 2019. URL:http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-155648

10. Grinberg M. Flask Web Development: Developing Web Applications with Python. – O`Reilly Media, Inc., 2018. - 316 p.

11. Lacey N. SQLite. In Python by Example (pp. 134–146). Cambridge University Press. - 2019 https://doi.org/10.1017/9781108591942.021

12. Lubis A., Dalimunthe R., Absah Y. & Fawzeea B. K. The Influence of Customer Relationship Management (CRM) Indicators on Customer Loyalty of Sharia Based Banking System. // Lubis, 2020, A. – PP. 84-92.

13. Chatterjee S., Rana N. P., Tamilmani K. & Sharma A. The effect of AI-based CRM on organization performance and competitive advantage: An empirical analysis in the B2B context // Industrial Marketing Management, – 2021 vol. 97, PP. 205-219.

14. Relan K. Building REST APIs with Flask: Create Python Web Services with MySQL. – Apress, 2019. - 199 p.

15. Relan K. Database Modeling in Flask // Building REST APIs with Flask. Apress, Berkeley, CA. 2019. – PP. 27-58. https://doi.org/10.1007/978-1-4842-5022-8_2

16. Patel A., Tomar D., Mourya V. and Kumar N. S. An Enhanced Framework for Digital Repository using Bootstrap Technique // International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 711-713, doi: 10.1109/ICACITE51222.2021.9404569.

17. Zhang J., Tan X., Wang X., Yan A. & Qin Z. T2FA: Transparent Two-Factor Authentication. IEEE Access 6, 2018 - 32677–32686

18. Ahmad I., Kumar T., Liyanage M., Ylianttila M., Koskela T., Braysy T. & Huusko J. Towards gadget-free internet services: A roadmap of the naked world. // Telematics and Informatics, 2018, vol. 35(1). – PP. 82-92.

ДОДАТКИ

Додаток А

```
from main import db, login_manager
from flask_login import UserMixin
from datetime import datetime
import os
import base64
import onetimepass


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


class User(db.Model, UserMixin):
    '''
        Class contain information about site's users.
    '''
    id = db.Column(db.Integer, primary_key=True)
    first_name = db.Column(db.String(20), nullable=False)
    last_name = db.Column(db.String(20), nullable=False)
    phone = db.Column(db.Integer, nullable=False, unique=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(60), nullable=True)
    auto = db.relationship('Auto', backref='owner', lazy=True)
    otp_secret = db.Column(db.String(16))

    def __repr__(self):
        return f'[User: id={self.id}, name={self.first_name} {self.last_name},
phone_number={self.phone}]'

    def __init__(self, **kwargs):
        super(User, self).__init__(**kwargs)
        if self.otp_secret is None:
            # generate a random secret
            self.otp_secret = base64.b32encode(os.urandom(10)).decode('utf-8')

    def get_totp_uri(self):
        return 'otpauth://totp/auto-service:{0}?secret={1}&issuer=auto-
service&period=30' \
            .format(self.email, self.otp_secret)

    def verify_totp(self, token):
        return onetimepass.valid_totp(token, self.otp_secret)
```

```
class Service(db.Model):
    '''
        Class contain information about services presented in the web-add.
    '''
    id = db.Column(db.Integer, primary_key=True)
    service_title = db.Column(db.String(60), nullable=False, unique=True)
    service_price = db.Column(db.Integer, nullable=False)
    service_comment = db.Column(db.String(200), nullable=True)

    def __repr__(self):
        return f'[Service: id={self.id}, title={self.service_title},
service_price={self.service_price}]'


class Auto(db.Model):
    '''
        Class cotain information about users' auto.
    '''
    id = db.Column(db.Integer, primary_key=True)
    vin = db.Column(db.String(17), nullable=False, unique=True)
    brand = db.Column(db.String(25), nullable=False)
    model = db.Column(db.String(20), nullable=False)
    year = db.Column(db.Integer, nullable=False)
    engine_volume = db.Column(db.String(5), nullable=False)
    trans_type = db.Column(db.String(12), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    # inservice = db.relationship('CurrentWork', backref='inservice', lazy=True)

    def __repr__(self):
        return f'[Auto: {self.brand} {self.model}, vin_code={self.vin},
owner={self.user_id}]'



class CurrentWork(db.Model):
    '''
        Class contains information about current work.
    '''
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(60), nullable=False)
    first_name = db.Column(db.String(20), nullable=False)
    last_name = db.Column(db.String(20), nullable=False)
    phone = db.Column(db.Integer, nullable=False)
    price = db.Column(db.Integer, nullable=False)
    date_open = db.Column(db.DateTime, nullable=False,
                          default=datetime.now())
    date_close = db.Column(db.DateTime, nullable=True,
                           default=None)
```

```
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=True)
    # auto_id = db.Column(db.Integer, db.ForeignKey('auto.id'), nullable=False)

    def __repr__(self):
        return f'[Work #{self.id}: {self.title} for {self.last_name}
{self.first_name}]'
```

Додаток Б

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We"
crossorigin="anonymous">
    <script src="https://kit.fontawesome.com/99c004e205.js"
crossorigin="anonymous"></script>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}"
type="text/css">
    {% if title %}
      <title>СТО "Мастер" - {{ title }}</title>
    {% else %}
      <title>СТО "Мастер"</title>
    {% endif %}
    <link rel="shortcut icon" href="{{ url_for('static',
filename='pics/favicon.ico') }}">
  </head>
  <body>
    <nav class="navbar navbar-expand-md bg-dark navbar-dark fixed-top">
      <div class="container">
        <a class="navbar-brand me-4" href="{{ url_for('home') }}"><i class="fas
fa-car"></i> СТО "Мастер"</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <div class="navbar-nav me-auto">
            <a href="{{ url_for('home') }}" class="nav-item nav-link">Головна</a>
            {% if current_user.id == 1 %}
              <a href="{{ url_for('service') }}" class="nav-item nav-
link">Послуги</a>
              <a href="{{ url_for('users') }}" class="nav-item nav-
link">Користувачі</a>
              <a href="{{ url_for('orders') }}" class="nav-item nav-
link">Замовлення</a>
            {% else %}
              <a href="{{ url_for('about') }}" class="nav-item nav-link">Про
нас</a>
              {% if current_user.is_authenticated %}
```

```
                <a href="{{ url_for('user_new_order', user_id=current_user.id)
}}" class="nav-item nav-link">Замовити послугу</a>
              {% else %}
                <a href="{{ url_for('new_order') }}" class="nav-item nav-
link">Замовити послугу</a>
              {% endif %}
            {% endif %}
          </div>
          <div class="navbar-nav">
            {% if current_user.is_authenticated%}
              {% if current_user.id != 1 %}
              <!-- a href="#" class="nav-item nav-link">Корзина <i class="fas fa-
shopping-basket"></i></a -->
              {% endif %}
              <a href="{{ url_for('account') }}" class="nav-item nav-
link">Профіль</a>
              <a href="{{ url_for('logout') }}" class="nav-item nav-
link">Вийти</a>
            {% else %}
              <!--a href="#" class="nav-item nav-link">Корзина <i class="fas fa-
shopping-basket"></i></a-->
              <a href="{{ url_for('login') }}" class="nav-item nav-link">Вхід</a>
              <a href="{{ url_for('register') }}" class="nav-item nav-
link">Реєстрація</a>
            {% endif %}
          </div>
        </div>
      </div>
    </nav>
    <div class="container pt-2">
      {% with messages = get_flashed_messages(with_categories=True) %}
        {% if messages %}
          {% for category, message in messages %}
            <div class="alert alert-{{ category }}">
              {{ message }}
            </div>
          {% endfor %}
        {% endif %}
      {% endwith %}
      <div class="row">
        <div class="col-md me-md-auto">
          {% block content %}{% endblock content %}
        </div>
      </div>
    </div>
    {% if current_user.id != 1 %}
    <div class="container py-3">
      <footer class="d-flex flex-wrap justify-content-between align-items-center
py-3 border-top">
```

```
        <div class="col-md-4 d-flex align-items-center">
          <!--<a href="/" class="mb-3 me-2 mb-md-0 text-muted text-decoration-
none lh-1">
            image
          </a>-->
          <span class="text-muted">&copy; 2021 СТО "Мастер"</span>
        </div>

        <ul class="nav col-md-4 justify-content-end list-unstyled d-flex">
          <li class="ms-3"><a class="text-muted"
href="https://www.facebook.com/stomastersumy"><i class="fab fa-facebook-square
fa-2x"></i></a></li>
          <li class="ms-3"><a class="text-muted"
href="https://www.instagram.com/"><i class="fab fa-instagram fa-2x"></i></a></li>
          <!-- <li class="ms-3"><a class="text-muted" href="#">image</a></li> -->
        </ul>
      </footer>
    </div>
    {% endif %}
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js
" integrity="sha384-
U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFIhvj"
crossorigin="anonymous"></script>
  </body>
</html>
```

```
*, *::after, *::before {
  padding: 0;
  margin: 0;
}

body {
  background: #fafafa;
  margin-top: 3.5em;
}

.nav-link {
  color: white !important;
}

.nav-link:hover,
.nav-link:active {
  color: cyan !important;
}

.navbar .navbar-toggler:focus {
```

```
  box-shadow: none;
}

.text-height {
  padding-top: 25px;
}

@media (max-width: 768px) {
  .text-height {
    height: 150px;
  }
}

@media (min-width: 992px) {
  .text-height {
    height: 250px;
  }
}

.custom-button {
  width: 150px;
}

.new-service {
  position: fixed;
  width: 150px;
  height: 50px;
}

.service-box,
.user-style {
  text-decoration: none;
  color: black;
}

.service-box:hover,
.user-style:hover {
  text-decoration: none;
  color: white;
  background: #999;
}

.new-service-auto {
  width: 200px;
  height: 50px;
}

.caption {
```

```
  text-shadow: -1px -1px 0 #000, 1px -1px 0 #000, -1px 1px 0 #000, 1px 1px 0
#000;
}
```

```
{% extends 'layout.html' %}
{% block content %}
  <div class="container">
    <div class="row">
      <div class="col-lg-8">
        <div id="carouselExampleIndicators" class="carousel slide w-75 mx-auto"
data-bs-ride="carousel">
          <div class="carousel-indicators">
            <button type="button" data-bs-target="#carouselExampleIndicators"
data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide
1"></button>
            <button type="button" data-bs-target="#carouselExampleIndicators"
data-bs-slide-to="1" aria-label="Slide 2"></button>
            <button type="button" data-bs-target="#carouselExampleIndicators"
data-bs-slide-to="2" aria-label="Slide 3"></button>
            <button type="button" data-bs-target="#carouselExampleIndicators"
data-bs-slide-to="3" aria-label="Slide 4"></button>
          </div>
          <div class="carousel-inner">
            <div class="carousel-item active">
              <img src="{{ url_for('static', filename='pics/pics_1_new.jpg') }}"
class="d-block img-thumbnail w-100" alt="Picture 1" height="450px">
              <div class="carousel-caption d-none d-lg-block">
                <h5 class='caption bg-dark bg-opacity-25'>FIAT DOBLO</h5>
                <p class='caption bg-dark bg-opacity-25'>Заміна радіатора
охолодження, ревізія патрубків системи охолодження, заміна антифрізу. Автомобіль
до зими готовий!</p>
              </div>
            </div>
            <div class="carousel-item">
              <img src="{{ url_for('static', filename='pics/pics_2_new.jpg') }}"
class="d-block img-thumbnail w-100" alt="Picture 2" height="450px">
              <div class="carousel-caption d-none d-lg-block">
                <h5 class='caption bg-dark bg-opacity-25'>SSANGYONG KYRON</h5>
                <p class='caption bg-dark bg-opacity-25'>Замінено наружний ШРУС,
сальники коробки. Замінено масло ДВС та мостів, фільтри. Зроблено проточку
гальмівних дисків та встановлено нові колодки.</p>
              </div>
            </div>
            <div class="carousel-item">
              <img src="{{ url_for('static', filename='pics/pics_3_new.jpg') }}"
class="d-block img-thumbnail w-100" alt="Picture 3" height="450px">
              <div class="carousel-caption d-none d-lg-block">
                <h5 class='caption bg-dark bg-opacity-25'>KIA MOHAVE</h5>
```

```
              <p class='caption bg-dark bg-opacity-25'>Регламентна заміна
мастил ДВС, коробки, роздатки. Діагностика показала необхідність заміни втулок
заднього стабілізатора.</p>
            </div>
          </div>
          <div class="carousel-item">
            <img src="{{ url_for('static', filename='pics/pics_4_new.jpg') }}"
class="d-block img-thumbnail w-100" alt="Picture 3" height="450px">
            <div class="carousel-caption d-none d-lg-block">
              <h5 class='caption bg-dark bg-opacity-25'>MERSEDES BENZ,
190D</h5>
              <p class='caption bg-dark bg-opacity-25'>Заміна подушок двигуна
та КПП, рульових тяг, передніх тормозних дисків і колодок, сайлентблоків задніх
ричагів</p>
            </div>
          </div>
        </div>
        <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleIndicators" data-bs-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Previous</span>
        </button>
        <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleIndicators" data-bs-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Next</span>
        </button>
      </div>
    </div>
    <div class="col-lg-4 my-lg-auto mt-md-3">
      <h3 class="pb-2 border-bottom-3">Трішки про сервіс</h3>
      <p class="fs-4">Ми завжди знаходимо свій підхід при виконанні навіть
рутинної роботи, аби задовольнити усі потреби клієнта.
        Завдяки цьому, клієнти завжди залишаються задоволені виконаною
роботою!</p>
    </div>
  </div>
</div>
{% endblock content %}
```

```
{% extends "layout.html" %}
{% block content %}
<div class="container">
  <div class="row">
    <div class="col-md-12 mt-2 mb-1">
      <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2517.5960291092065!2d3
```

```
4.7711548156323!3d50.87567807953617!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1
m2!1s0x412901cc75eba787%3A0xc1d59602e7c25fb0!2z0YPQuy4g0JPQtdGA0LDRgdC40LzQsCDQmt
C-
0L3QtNGA0LDRgtGM0LXQstCwLCAxNjgsINCh0YPQvNGLLCDQodGD0LzRgdC60LDRjyDQvtCx0LvQsNGB0
YLRjCwgNDAwMDA!5e0!3m2!1sru!2sua!4v1639072102843!5m2!1sru!2sua"
      height="400"
      class="w-100"
      style="border: 2px solid gray;"
      allowfullscreen=""
      loading="lazy"></iframe>
    </div>
  </div>
</div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="container">
    <div class="row">
      <div class="col-md-7 col-lg-6 mb-4">
        <h1>Замовити послугу</h1>
        <form method="POST" action="">
          {{ form.hidden_tag() }}

          <!-- validation_form for title -->
          {{ form.title.label(class="form-select-label") }}
          {% if form.title.errors %}
            {{ form.title(class="form-select is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.title.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.title(class="form-select form-select-lg")}}
          {% endif %}

          <!-- validation_form for last_name -->
          {{ form.last_name.label(class="form-control-label") }}
          {% if form.last_name.errors %}
            {{ form.last_name(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.last_name.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
```

```
            {{ form.last_name(class="form-control form-control-lg")}}
          {% endif %}

          <!-- validation_form for first_name -->
          {{ form.first_name.label(class="form-control-label") }}
          {% if form.first_name.errors %}
            {{ form.first_name(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.first_name.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.first_name(class="form-control form-control-lg")}}
          {% endif %}

          <!-- validation_form for phone -->
          {{ form.phone.label(class="form-control-label") }}
          {% if form.phone.errors %}
            {{ form.phone(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.phone.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.phone(class="form-control form-control-lg",
placeholder="380661112223")}}
          {% endif %}
          {{ form.submit(class="btn btn-outline-primary mt-2 mb-1") }}
        </form>
      </div>
    </div>
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content%}
  <div class="container mt-2">
    <div class="row">
      <div class="col-md-8 col-lg-6 mb-4">
        <h1>Увійти</h1>
        <form method="POST" action="">
          {{ form.hidden_tag() }}

          <!-- validation_form for email -->
          {{ form.email.label(class="form-control-label") }}
```

```
          {% if form.email.errors %}
            {{ form.email(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.email.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.email(class="form-control form-control-lg",
placeholder="example@email.com")}}
          {% endif %}


          <!-- validation_form for password -->
          {{ form.password.label(class="form-control-label") }}
          {% if form.password.errors %}
            {{ form.password(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.password.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.password(class="form-control form-control-lg")}}
          {% endif %}


          <!-- validation_form for token -->
          {{ form.token.label(class="form-control-label") }}
          {% if form.token.errors %}
            {{ form.token(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.token.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.token(class="form-control form-control-lg")}}
          {% endif %}


          <!-- form for var remember in class LoginForm -->
          <div class="form-check mt-1">
            {{ form.remember(class="form-check-input") }}
            {{ form.remember.label(class="form-check-label")}}
          </div>
          {{ form.submit(class="btn btn-outline-info mt-2 mb-1") }}
        </form>
        <small class="col-md-6">
          <a href="#" class="position-relative">Забули пароль?</a>
```

```
        </small>
        <hr class="my-2">
        <small class="text-muted">
          Ще немає акаунту? <a href="{{ url_for('register') }}" class="me-2 link-
primary">Зареєструватися</a>
        </small>
      </div>
    </div>
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="container mt-2">
    <div class="row">
      <div class="col-md-8 col-lg-6 mb-4">
        <h1>Реєстрація</h1>
        <form method="POST" action="">
          {{ form.hidden_tag() }}

          <!-- validation_form for last_name -->
          {{ form.last_name.label(class="form-control-label") }}
          {% if form.last_name.errors %}
            {{ form.last_name(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.last_name.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.last_name(class="form-control form-control-lg")}}
          {% endif %}

          <!-- validation_form for first_name -->
          {{ form.first_name.label(class="form-control-label") }}
          {% if form.first_name.errors %}
            {{ form.first_name(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.first_name.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.first_name(class="form-control form-control-lg")}}
          {% endif %}


          <!-- validation_form for phone -->
```

```
          {{ form.phone.label(class="form-control-label") }}
          {% if form.phone.errors %}
            {{ form.phone(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.phone.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.phone(class="form-control form-control-lg",
placeholder="380661112223")}}
          {% endif %}

          <!-- validation_form for email -->
          {{ form.email.label(class="form-control-label") }}
          {% if form.email.errors %}
            {{ form.email(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.email.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.email(class="form-control form-control-lg",
placeholder="example@email.com")}}
          {% endif %}

          <!-- validation_form for password -->
          {{ form.password.label(class="form-control-label") }}
          {% if form.password.errors %}
            {{ form.password(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.password.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
          {% else %}
            {{ form.password(class="form-control form-control-lg")}}
          {% endif %}

          <!-- validation_form for confirm_password -->
          {{ form.confirm_password.label(class="form-control-label") }}
          {% if form.confirm_password.errors %}
            {{ form.confirm_password(class="form-control is-invalid")}}
            <div class="invalid-feedback">
              {% for error in form.confirm_password.errors %}
                <span>{{ error }}</span>
              {% endfor %}
            </div>
```

```
          {% else %}
            {{ form.confirm_password(class="form-control form-control-lg")}}
          {% endif %}

          {{ form.submit(class="btn btn-outline-info mt-2 mb-1") }}
        </form>
        <small class="text-muted">
          Вже маєте акаунт? <a href="{{ url_for('login') }}" class="me-2 link-
primary">Увійти</a>
        </small>
      </div>
    </div>
  </div>
{% endblock content %}
```

```
{% extends "layout.html" %}

{% block content %}
    <h1>Two Factor Authentication Setup</h1>
    <p>Ви майже завершили реєстрацію! Виконайте інструкцію нижче:</p>
    <p>
        <ol>
        <li>Скачайте Google Authenticator для <a
href="https://play.google.com/store/apps/details?id=com.google.android.apps.authe
nticator2&hl=en&gl=US" target="_blank">Android</a>
            або для <a href="https://apps.apple.com/ru/app/google-
authenticator/id388497605" target="_blank">IOS</a></li>
        <li>Відскануйте QR-Code нижче</li>
        </ol>
    </p>
    <p><img id="qrcode" src="{{ url_for('qrcode') }}"></p>
    <p>Ви закінчили сканування? Перейдіть на сторінку <a href="{{
url_for('login') }}">входу</a>.</p>
{% endblock content %}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="container mt-2">
    <div class="row">
      <div class="col-md">
        <div class="card">
          <div class="card-body">
            <h1 class="card-title">Основна інформація</h5>
            <p class="card-text" style="font-size: 22px;"><b>Ім'я:</b> {{
current_user.last_name }} {{ current_user.first_name }}</p>
            <p class="card-text">
              <form method="POST" action="">
                {{ form.hidden_tag() }}
```

```
            <!-- validation_form for phone -->
            {{ form.phone.label(class="form-control-label") }}
            {% if form.phone.errors %}
              {{ form.phone(class="form-control is-invalid")}}
              <div class="invalid-feedback">
                {% for error in form.phone.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.phone(class="form-control form-control-lg")}}
            {% endif %}

            <!-- validation_form for email -->
            {{ form.email.label(class="form-control-label") }}
            {% if form.email.errors %}
              {{ form.email(class="form-control is-invalid")}}
              <div class="invalid-feedback">
                {% for error in form.email.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.email(class="form-control form-control-lg") }}
            {% endif %}

            {{ form.submit(class="btn btn-success mt-4") }}
          </form>
        </p>
      </div>
    </div>
  </div>
</div>
{% if current_user.id != 1 %}
<div class="row">
  <div class="col-md-6 col-lg-6 mb-4 mt-3">
    <div class="card">
      <div class="card-body">
        <h3 class="card-title">Мої замовлення</h3>
        <p class="card-text">
          <table class="table">
            <tr>
              <th scope="col">#</th>
              <th scope="col">Назва</th>
              <th scope="col">Телефон</th>
              <th scope="col">Створено</th>
              <th scope="col">Статус</th>
              <th scope="col">Ціна</th>
            </tr>
```

```
                {% if orders %}
                  {% for order in orders %}
                    <tr scope="row">
                      <td>{{ order.id }}</td>
                      <td>{{ order.title }}</td>
                      <td>{{ order.phone }}</td>
                      <td>{{ order.date_open.strftime('%d/%b/%Y %H:%M') }}</td>
                      {% if order.date_close == None %}
                      <td><button type="button" class="btn btn-warning
disabled">У процесі</button></td>
                      {% else %}
                      <td><button type="button" class="btn btn-success
disabled">Завершено</button></td>
                      {% endif %}
                      <td>{{ order.price }}&#8372;</td>
                    </tr>
                  {% endfor %}
                {% else %}
                  <tr>
                    <td colspan="8" class="position-relative pt-4 pb-4">
                      <span class="position-absolute top-50 start-50 translate-
middle" style="font-size: 22px; font-weight: 400">Ще немає замовлень</span>
                    </td>
                  </tr>
                {% endif %}
              </table>
            </p>
          </div>
        </div>
      </div>
      <div class="col-md-6 col-lg-6 mb-4 mt-3">
        <div class="card">
          <div class="card-body">
            <h3 class="card-title">Моє авто</h3>
              {% for auto in current_user.auto %}
                      <p class="card-title mt-3 mb-3" style="font-size:
20px;"><b>{{ auto.brand }} {{ auto.model }}</b></p>
                      <p class="card-text">VIN-код: {{ auto.vin }}</p>
                      <p class="card-text">Рік випуску: {{ auto.year }}</p>
                      <p class="card-text">Об'єм двигуна: {{ auto.engine_volume
}} л.</p>
                      <p class="card-text">Тип коробки передач: {{
auto.trans_type }}</p>
                    {% endfor %}
          </div>
        </div>
      </div>
    </div>
    {% endif %}
```

```
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content%}
  <div class="container">
    <div class="row">
      <div class="col-md-8">
        {% for service in services %}
        <a class="row border border-secondary rounded mt-2 service-box"
        href="{{ url_for('new_service', service_id=service.id) }}"
        data-bs-toggle="tooltip"
        data-bs-placement="bottom"
        title="Опис послуги: {{ service.service_comment }}">
          <h3 class="service-box-title">{{ service.service_title }}</h3>
          <p class="service-box-price">Ціна: <i>{{ service.service_price }}</i>
&#8372;</p>
        </a>
        {% endfor %}
      </div>
      <div class="col-md-4 mt-5">
        <div class="mx-auto custom-button">
          <button type="button" class="btn btn-outline-primary new-service"
          data-bs-toggle="modal"
          data-bs-target="#exampleModal">
            <b>+ Нову послугу</b>
          </button>
        </div>
      </div>
    </div>
  </div>



  <!-- Modal -->
  <div class="modal fade" id="exampleModal" tabindex="-1"
  aria-labelledby="exampleModalLabel"
  aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Нова послуга</h5>
          <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
        </div>
        <div class="modal-body">
          <form method="POST" action="">
            {{ form.hidden_tag() }}
```

```
            <!-- validation_form for service_title -->
            {{ form.title.label(class="form-control-label") }}
            {% if form.title.errors %}
              {{ form.title(class="form-control is-invalid")}}
              <div class="invalid-feedback">
                {% for error in form.title.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.title(class="form-control form-control-lg")}}
            {% endif %}


            <!-- validation_form for service_price -->
            {{ form.price.label(class="form-control-label") }}
            {% if form.price.errors %}
              {{ form.price(class="form-control is-invalid")}}
              <div class="invalid-feedback">
                {% for error in form.price.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.price(class="form-control form-control-lg")}}
            {% endif %}

            <!-- validation_form for service_comment -->
            {{ form.comment.label(class="form-control-label") }}
            {% if form.comment.errors %}
              {{ form.comment(class="form-control is-invalid")}}
              <div class="invalid-feedback">
                {% for error in form.comment.errors %}
                  <span>{{ error }}</span>
                {% endfor %}
              </div>
            {% else %}
              {{ form.comment(class="form-control form-control-lg")}}
            {% endif %}

            {{ form.submit(class="btn btn-outline-primary mt-2 mb-1") }}
          </form>
        </div>
      </div>
    </div>
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="container">
    <div class="col-md-8 mx-auto">
      <h2>{{ service.service_title }}</h2>
      <p>Ціна: <i>{{ service.service_price }}</i> &#8372;</p>
      <p>Опис послуги: {{ service.service_comment }}</p>
      <div class="row">
        <div class="col-md-6">
          <a href="{{ url_for('service') }}">
            <button type="button" class="btn btn-outline-info">
              <i class="fas fa-arrow-left"></i> Назад
            </button>
          </a>
        </div>
        <div class="col-md-6">
          <a href="{{ url_for('update_service', service_id=service.id) }}">
            <button type="button" class="btn btn-outline-success">
              Оновити <i class="far fa-edit"></i>
            </button>
          </a>
        </div>
      </div>
    </div>
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content%}
  <div class="modal-body">
    <a href="{{ url_for('new_service', service_id=service.id) }}">
      <button type="button" class="btn btn-outline-info">
        <i class="fas fa-arrow-left"></i> Назад
      </button>
    </a>
    <form method="POST" action="">
      {{ form.hidden_tag() }}


      <!-- validation_form for service_title -->
      {{ form.title.label(class="form-control-label") }}
      {% if form.title.errors %}
        {{ form.title(class="form-control is-invalid")}}
        <div class="invalid-feedback">
          {% for error in form.title.errors %}
            <span>{{ error }}</span>
          {% endfor %}
        </div>
```

```
      {% else %}
        {{ form.title(class="form-control form-control-lg")}}
      {% endif %}


      <!-- validation_form for service_price -->
      {{ form.price.label(class="form-control-label") }}
      {% if form.price.errors %}
        {{ form.price(class="form-control is-invalid")}}
        <div class="invalid-feedback">
          {% for error in form.price.errors %}
            <span>{{ error }}</span>
          {% endfor %}
        </div>
      {% else %}
        {{ form.price(class="form-control form-control-lg")}}
      {% endif %}

      <!-- validation_form for service_comment -->
      {{ form.comment.label(class="form-control-label") }}
      {% if form.comment.errors %}
        {{ form.comment(class="form-control is-invalid")}}
        <div class="invalid-feedback">
          {% for error in form.comment.errors %}
            <span>{{ error }}</span>
          {% endfor %}
        </div>
      {% else %}
        {{ form.comment(class="form-control form-control-lg")}}
      {% endif %}

      {{ form.submit(class="btn btn-outline-success mt-2 mb-1") }}
    </form>
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="container mt-2">
    <div class="row">
      <div class="col-md-12">
        {% for user in users %}
        {% if user.id != 1 %}
        <div class="btn-group dropend user-style mb-2 w-50" href="#">
          <button type="button" class="btn btn-outline-secondary dropdown-toggle
pe-3 d-inline-flex" style="height: 53px;" data-bs-toggle="dropdown" aria-
expanded="false" style="width: 13rem;">
            <table class="table">
              <tr>
```

```
                <td><b>{{ user.id - 1 }}</b></td>
                <td><span style="font-size: 18px;">{{ user.last_name }} {{
user.first_name }}</span></td>
                <td><span style="font-size: 18px;">{{ user.phone }}</span></td>
                <td><span style="font-size: 18px;">{{ user.email }}</span></td>
              </tr>
            </table>
          </button>
          <ul class="dropdown-menu ms-5 p-2" style="width: 300px;">
            <li><h5>Авто</h5></li>
            {% for auto in user.auto %}
              <li><b>VIN-код:</b> {{ auto.vin }}</li>
              <li><b>Марка:</b> {{ auto.brand }}</li>
              <li><b>Модель:</b> {{ auto.model }}</li>
              <li><b>Рік:</b> {{ auto.year }}</li>
              <li><b>Тип коробки:</b> {{ auto.trans_type }}</li>
            {% endfor %}
            <br>
            <li><a href="{{ url_for('auto', user_id=user.id ) }}">
              <button type="button" class="btn btn-primary"><b>Додати
авто</b></button>
            </a></li>
          </ul>
        </div><br>
        {% endif %}
        {% endfor %}
      </div>
    </div>
  </div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
<div class="row">
  <div class="col-md-8 mt-2">
    <h3>Додати нове авто</h3>
    <form method="POST" action="">
      {{ form.hidden_tag() }}

      <!-- validation_form for vin -->
      {{ form.vin.label(class="form-control-label") }}
      {% if form.vin.errors %}
        {{ form.vin(class="form-control is-invalid")}}
        <div class="invalid-feedback">
          {% for error in form.vin.errors %}
            <span>{{ error }}</span>
          {% endfor %}
```

```
      </div>
    {% else %}
      {{ form.vin(class="form-control form-control-lg")}}
    {% endif %}

    <!-- validation_form for brand -->
    {{ form.brand.label(class="form-select-label") }}
    {% if form.brand.errors %}
      {{ form.brand(class="form-select is-invalid")}}
      <div class="invalid-feedback">
        {% for error in form.brand.errors %}
          <span>{{ error }}</span>
        {% endfor %}
      </div>
    {% else %}
      {{ form.brand(class="form-select form-select-lg")}}
    {% endif %}

    <!-- validation_form for model -->
    {{ form.model.label(class="form-control-label") }}
    {% if form.model.errors %}
      {{ form.model(class="form-control is-invalid")}}
      <div class="invalid-feedback">
        {% for error in form.model.errors %}
          <span>{{ error }}</span>
        {% endfor %}
      </div>
    {% else %}
      {{ form.model(class="form-control form-control-lg")}}
    {% endif %}

    <!-- validation_form for year -->
    {{ form.year.label(class="form-control-label") }}
    {% if form.year.errors %}
      {{ form.year(class="form-control is-invalid")}}
      <div class="invalid-feedback">
        {% for error in form.year.errors %}
          <span>{{ error }}</span>
        {% endfor %}
      </div>
    {% else %}
      {{ form.year(class="form-control form-control-lg")}}
    {% endif %}

    <!-- validation_form for engine_volume -->
    {{ form.engine_volume.label(class="form-control-label") }}
    {% if form.engine_volume.errors %}
      {{ form.engine_volume(class="form-control is-invalid")}}
      <div class="invalid-feedback">
```

```
          {% for error in form.engine_volume.errors %}
            <span>{{ error }}</span>
          {% endfor %}
        </div>
      {% else %}
        {{ form.engine_volume(class="form-control form-control-lg")}}
      {% endif %}

      <!-- validation_form for trans_type -->
      {{ form.trans_type.label(class="form-select-label") }}
      {% if form.trans_type.errors %}
        {{ form.trans_type(class="form-select is-invalid")}}
        <div class="invalid-feedback">
          {% for error in form.trans_type.errors %}
            <span>{{ error }}</span>
          {% endfor %}
        </div>
      {% else %}
        {{ form.trans_type(class="form-select form-select-lg")}}
      {% endif %}


      {{ form.submit(class="btn btn-outline-primary mt-2 mb-1") }}
    </form>
  </div>
  <div class="col-md-4 mt-2">
    <h3>Авто користувача {{ user.first_name }} {{ user.last_name }}</h3>
    {% for auto in user.auto %}
    <ul class="list-group">
      <li class="list-group-item">{{ auto.brand }} {{ auto.model }} ({{ auto.year
}})</li>
    </ul>
    {% endfor %}
    <a href="{{ url_for('users') }}">
      <button class='btn btn-outline-info mt-3'>Назад</button>
    </a>
  </div>
</div>
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="container mt-2">
    <div class="row">
      <div class="col-md-12">
        <table class="table">
          <tr>
            <th scope="col">#</th>
```

```
            <th scope="col">Назва</th>
            <th scope="col">Ім'я</th>
            <th scope="col">Телефон</th>
            <th scope="col">Зареєстрован</th>
            <th scope="col">Створено</th>
            <th scope="col">Статус</th>
            <th scope="col">Завершено</th>
            <th scope="col">Ціна</th>
          </tr>
          {% if orders %}
            {% for order in orders %}
              <tr scope="row">
                <td>{{ order.id }}</td>
                <td>{{ order.title }}</td>
                <td>{{ order.last_name }} {{ order.first_name }}</td>
                <td>{{ order.phone }}</td>
                <td>{{ 'Так' if order.user_id else 'Ні' }}</td>
                <td>{{ order.date_open.strftime('%d/%b/%Y %H:%M') }}</td>
                {% if order.date_close == None %}
                <td><button type="button" class="btn btn-warning disabled">У
процесі</button></td>
                <td>
                  <a href="{{ url_for('order', order_id=order.id) }}">
                    <button type="button" class="btn btn-
danger">Завершити</button>
                  </a>
                </td>
                {% else %}
                <td>{{ order.date_close.strftime('%d/%b/%Y %H:%M') }}</td>
                <td>
                  <button type="button" class="btn btn-success
disabled">Завершено</button>
                </td>
                {% endif %}
                <td>{{ order.price }}&#8372;</td>
              </tr>
            {% endfor %}
          {% else %}
            <tr>
              <td colspan="8" class="position-relative pt-4 pb-4">
                <span class="position-absolute top-50 start-50 translate-middle"
style="font-size: 22px; font-weight: 400">Немає замовлень</span>
              </td>
            </tr>
          {% endif %}
        </table>
      </div>
    </div>
  </div>
```

```
{% endblock content%}
```

```
{% extends "layout.html" %}
{% block content %}
  <div class="card col-md-6 mt-2">
    <div class="card-body">
      <h2 class="card-title">Order#{{ order.id }}</h2>
      <div class="row">
        <div class="col-md-3">
          <a href="{{ url_for('orders') }}">
            <button type="button" class="btn btn-info mt-2">
              <i class="fas fa-arrow-left"></i> Назад
            </button>
          </a>
        </div>
        <div class="col-md-3">
          <form method="POST" action="">
            {{ form.hidden_tag() }}

            <!-- validation_form for submit -->
            {{ form.submit(class="btn btn-success mt-2 mb-1") }}
          </form>
        </div>
      </div>
    </div>
  </div>
{% endblock content%}
```

Додаток В

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager

app = Flask(__name__)

app.config['SECRET_KEY'] = '59af7072baad112f53425d7380b654a1'  # secret key of
the app
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'  # config the
database

db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'
login_manager.login_message_category = 'info'

from main import routes
```

Додаток Г

```
from flask import render_template, url_for, flash, redirect, request, abort,
session
from main import app, db, bcrypt
from main.forms import RegistrationForm, LoginForm, UpdateAccountForm, OrderForm,
AutoForm, WorkForm, UpdateOrders
from main.models import User, Service, Auto, CurrentWork
from flask_login import login_user, current_user, logout_user, login_required
from datetime import datetime
from io import BytesIO
import pyqrcode


# general pages
@app.route('/')
@app.route('/home')
def home():
    return render_template('home.html')


@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        # if user.id == 1: # проверяет ,если логинится чел с ид=1 т опускает, в
противном случае - выдает ошибку
        if user and bcrypt.check_password_hash(user.password, form.password.data)
and user.verify_totp(form.token.data):
            login_user(user, remember=form.remember.data)
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else
redirect(url_for('home'))
        else:
            flash('E-Mail або пароль неправильний. Повторіть спробу ще раз',
'danger')
    return render_template('login.html', title='Login', form=form)


@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('home'))
    form = RegistrationForm()
    if form.validate_on_submit():
```

```
        hashed_password = bcrypt.generate_password_hash(
            form.password.data).decode('utf-8')
        user = User(first_name=str(form.first_name.data).rstrip(),
                    last_name=str(form.last_name.data).rstrip(),
                    phone=str(form.phone.data).rstrip(),
                    email=str(form.email.data).rstrip(),
                    password=hashed_password)
        db.session.add(user)
        db.session.commit()

        session['email'] = user.email
        return redirect(url_for('two_factor_setup'))
    return render_template('register.html', title='Register', form=form)


@app.route('/twofactor')
def two_factor_setup():
    if 'email' not in session:
        return redirect(url_for('home'))
    user = User.query.filter_by(email=session['email']).first()
    if user is None:
        return redirect(url_for('home'))

    return render_template('two-factor-setup.html'), 200, {
        'Cache-Contol': 'no-cache, no-store, must-revalidate',
        'Pragma': 'no-cache',
        'Expires': '0'}


@app.route('/qrcode')
def qrcode():
    if 'email' not in session:
        abort(404)
    user = User.query.filter_by(email=session['email']).first()
    if user is None:
        abort(404)

    del session['email']

    url = pyqrcode.create(user.get_totp_uri())
    stream = BytesIO()
    url.svg(stream, scale=5)
    return stream.getvalue(), 200, {
        'Content-Type': 'image/svg+xml',
        'Cache-Control': 'no-cache, no-store, must-revalidate',
        'Pragma': 'no-cache',
        'Expires': '0'}


@app.route("/logout")
```

```
def logout():
    logout_user()
    return redirect(url_for('home'))


@app.route("/account", methods=['GET', 'POST'])
@login_required
def account():
    form = UpdateAccountForm()
    orders = CurrentWork.query.filter_by(user_id=current_user.id).all()
    if form.validate_on_submit():
        current_user.phone = str(form.phone.data).rstrip()
        current_user.email = str(form.email.data).rstrip()
        db.session.commit()
        flash('Ваш акаунт оновлено!', 'success')
        return redirect(url_for('account'))
    elif request.method == 'GET':
        form.phone.data = current_user.phone
        form.email.data = current_user.email
    return render_template('account.html', title='Account', form=form,
orders=orders)


# typical user's pages
@app.route('/about')
def about():
    return render_template('about.html', title='About')


@app.route('/new_order', methods=['GET', 'POST'])
def new_order():
    if current_user.is_authenticated:
        abort(403)
    form = WorkForm()
    srvc_visible = Service.query.all()
    srvc = Service.query.filter_by(service_title=form.title.data).first()
    if form.validate_on_submit():
        work = CurrentWork(title=form.title.data,
                           first_name=form.first_name.data,
                           last_name=form.last_name.data,
                           phone=form.phone.data,
                           price=srvc.service_price)
        db.session.add(work)
        db.session.commit()
        flash('Замовлення створено!', 'success')
    return render_template('new_order.html', title='Current Work', form=form,
services=srvc_visible)


@app.route('/<int:user_id>/new_order', methods=['GET', 'POST'])
```

```
@login_required
def user_new_order(user_id):
    if current_user.id != user_id:
        abort(403)
    form = WorkForm()
    srvc_visible = Service.query.all()
    srvc = Service.query.filter_by(service_title=form.title.data).first()
    if form.validate_on_submit():
        work = CurrentWork(title=form.title.data,
first_name=form.first_name.data,
                           last_name=form.last_name.data, phone=form.phone.data,
user_id=current_user.id,
                           price=srvc.service_price)
        db.session.add(work)
        db.session.commit()
        flash('Замовлення створено!', 'success')
        return redirect(url_for('account'))
    elif request.method == 'GET':
        form.first_name.data = current_user.first_name
        form.last_name.data = current_user.last_name
        form.phone.data = current_user.phone
    return render_template('new_order.html', title='Current Work', form=form,
services=srvc_visible)


# admin's pages

@app.route('/service', methods=['GET', 'POST'])
@login_required
def service():
    if current_user.id != 1:
        abort(403)
    services = Service.query.all()
    form = OrderForm()
    if form.validate_on_submit():
        service = Service(service_title=str(form.title.data).rstrip(),
service_price=str(
            form.price.data).rstrip(), service_comment=str(form.comment.data))
        db.session.add(service)
        db.session.commit()
        flash('Нову послугу додано!', 'success')
        return redirect(url_for('service'))
    return render_template('service.html', title='Service', form=form,
services=services)


@app.route('/service/<int:service_id>', methods=['GET', 'POST'])
def new_service(service_id):
    if current_user.id != 1:
        abort(403)
```

```
    service = Service.query.get_or_404(service_id)
    return render_template('new_service.html', title=service.service_title,
service=service)


@app.route('/service/<int:service_id>/update', methods=['GET', 'POST'])
@login_required
def update_service(service_id):
    service = Service.query.get_or_404(service_id)
    if current_user.id != 1:
        abort(403)
    form = OrderForm()
    if form.validate_on_submit():
        service.service_title = str(form.title.data).rstrip()
        service.service_price = str(form.price.data).rstrip()
        service.service_comment = form.comment.data
        db.session.commit()
        flash('Послугу оновлено!', 'success')
        return redirect(url_for('new_service', service_id=service.id))
    elif request.method == 'GET':
        form.title.data = service.service_title
        form.price.data = service.service_price
        form.comment.data = service.service_comment
    return render_template('update_service.html', title='Update Service',
form=form, service=service)


@app.route('/users', methods=['GET', 'POST'])
@login_required
def users():
    if current_user.id != 1:
        abort(403)
    users = User.query.filter(id != 1).all()
    return render_template('users.html', title='Пользователи', users=users)


@app.route('/orders', methods=['GET', 'POST'])
@login_required
def orders():
    if current_user.id != 1:
        abort(403)
    orders = CurrentWork.query.all()
    return render_template('orders.html', title="Users' Orders", orders=orders)


@app.route('/orders/<int:order_id>', methods=['GET', 'POST'])
@login_required
def order(order_id):
    if current_user.id != 1:
        abort(403)
```

```
    order = CurrentWork.query.get_or_404(order_id)
    form = UpdateOrders()
    if form.validate_on_submit():
        order.date_close = datetime.now()
        db.session.commit()
        flash(f'Замовлення №{order.id} завершено!', 'success')
        return redirect(url_for('orders'))
    return render_template('order.html', title=f'Order#{order_id}', order=order,
form=form)


@app.route("/auto/user<int:user_id>", methods=['GET', 'POST'])
@login_required
def auto(user_id):
    if current_user.id != 1:
        abort(403)
    user = User.query.get_or_404(user_id)
    form = AutoForm()
    if form.validate_on_submit():
        auto = Auto(vin=str(form.vin.data).rstrip(),
                    brand=str(form.brand.data).rstrip(),
                    model=str(form.model.data).rstrip(),
                    year=str(form.year.data).rstrip(),
                    engine_volume=str(form.engine_volume.data).rstrip(),
                    trans_type=str(form.trans_type.data).rstrip(),
                    user_id=user.id)
        db.session.add(auto)
        db.session.commit()
        flash('Авто додано!', 'success')
        return redirect(url_for('users'))
    return render_template('auto_info.html', title='Добавить машину', user=user,
form=form)
```

Додаток Д

```
from flask_wtf import FlaskForm
from flask_login import current_user
from wtforms import StringField, PasswordField, SubmitField, BooleanField,
TextAreaField, IntegerField, SelectField
from wtforms.validators import DataRequired, Length, Email, EqualTo,
ValidationError
from main.models import User, Auto, Service

services = Service.query.all()


class RegistrationForm(FlaskForm):
    first_name = StringField(
        "І'мя", validators=[DataRequired(), Length(min=2, max=20)])
    last_name = StringField('Прізвище', validators=[
                            DataRequired(), Length(min=2, max=20)])
    phone = IntegerField('Номер телефону', validators=[DataRequired(
        'Будь ласка, уведіть телефон у форматі 380661112223')])
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    password = PasswordField('Пароль', validators=[DataRequired()])
    confirm_password = PasswordField('Повторіть пароль', validators=[
                                     DataRequired(), EqualTo('password')])
    submit = SubmitField('Зареєструватися')

    def validate_phone(self, phone):
        user = User.query.filter_by(phone=phone.data).first()
        if user:
            raise ValidationError('Такий номер вже існує!')

    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('Такий e-mail вже існує!')


class LoginForm(FlaskForm):
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    password = PasswordField('Пароль', validators=[DataRequired()])
    token = StringField('2FA ключ', validators=[DataRequired(), Length(6, 6)])
    remember = BooleanField("Запам'ятати мене")
    submit = SubmitField('Вхід')


class UpdateAccountForm(FlaskForm):
    phone = IntegerField('Номер телефону', validators=[DataRequired(
        'Будь ласка, уведіть номер телефону у форматі 0661112223')])
```

```
    email = StringField('E-mail', validators=[DataRequired(), Email()])
    submit = SubmitField('Оновити')

    def validate_phone(self, phone):
        if phone.data != current_user.phone:
            user = User.query.filter_by(phone=phone.data).first()
            if user:
                raise ValidationError('Такий номер вже існує!')

    def validate_email(self, email):
        if email.data != current_user.email:
            user = User.query.filter_by(email=email.data).first()
            if user:
                raise ValidationError('Такий e-mail вже існує!')


class OrderForm(FlaskForm):
    title = StringField('Назва послуги', validators=[DataRequired()])
    price = IntegerField('Ціна', validators=[DataRequired()])
    comment = TextAreaField('Опис послуги')
    submit = SubmitField('Додати')


class AutoForm(FlaskForm):
    vin = StringField('VIN-код', validators=[
                      DataRequired(), Length(min=17, max=17)])
    brand = SelectField('Марка авто', choices=[('Acura', 'Acura'),
                                               ('Alfa Romeo', 'Alfa Romeo'),
                                               ('Audi', 'Audi'),
                                               ('BMW', 'BMW'),
                                               ('Bentley', 'Bentley'),
                                               ('Cadillac', 'Cadillac'),
                                               ('Chevrolet', 'Chevrolet'),
                                               ('Chrysler', 'Chrysler'),
                                               ('Dodge', 'Dodge'),
                                               ('Fiat', 'Fiat'),
                                               ('Ford', 'Ford'),
                                               ('GMC', 'GMC'),
                                               ('Honda', 'Honda'),
                                               ('Hyundai', 'Hyundai'),
                                               ('Infiniti', 'Infiniti'),
                                               ('Jaguar', 'Jaguar'),
                                               ('Jeep', 'Jeep'),
                                               ('Kia', 'Kia'),
                                               ('Land Rover', 'Land Rover'),
                                               ('Lexus', 'Lexus'),
                                               ('Mazda', 'Mazda'),
                                               ('Mercedes-Benz',
                                                'Mercedes-Benz'),
                                               ('Mini', 'Mini'),
```

```
                                                ('Mitsubishi', 'Mitsubishi'),
                                                ('Nissan', 'Nissan'),
                                                ('Porsche', 'Porsche'),
                                                ('Saab', 'Saab'),
                                                ('Saturn', 'Saturn'),
                                                ('Smart', 'Smart'),
                                                ('Subaru', 'Subaru'),
                                                ('Suzuki', 'Suzuki'),
                                                ('Tesla', 'Tesla'),
                                                ('Toyota', 'Toyota'),
                                                ('Volkswagen', 'Volkswagen'),
                                                ('Volvo', 'Volvo'),
                                                ('VAZ', 'ВАЗ'),
                                                ('Niva', 'Niva')])
    model = StringField('Модель автомобіля', validators=[
                        DataRequired(), Length(min=2, max=20)])
    year = StringField('Рік виготовлення', validators=[DataRequired()])
    engine_volume = StringField("Об'єм", validators=[DataRequired()])
    trans_type = SelectField('Тип коробки передач', choices=[
                             ('Механіка', 'Механіка'), ('Автомат', 'Автомат')])
    submit = SubmitField('Додати')

    def validate_vin(self, vin):
        auto = Auto.query.filter_by(vin=vin.data).first()
        if auto:
            raise ValidationError(
                'VIN-код є у базі! Будь ласка, спробуйте ще раз')


class UpdateOrders(FlaskForm):
    submit = SubmitField('Завершити замовлення')


class WorkForm(FlaskForm):
    title = SelectField('Оберіть послугу', choices=[(
        service.service_title, service.service_title) for service in services])
    first_name = StringField(
        "Ім'я", validators=[DataRequired(), Length(min=2, max=20)])
    last_name = StringField('Прізвище', validators=[
                            DataRequired(), Length(min=2, max=20)])
    phone = IntegerField('Номер телефону', validators=[DataRequired(
        'Будь ласка, уведіть номер телефону у форматі 380661112223')])

    submit = SubmitField('Замовити')

from main import app

if __name__ == '__main__':
    app.run(debug=True)
```