МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

“До захисту допущений”

Зав. кафедри комп’ютерних наук

та прикладної математики

д.т.н., професор Турбал Ю.В

«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Створення інформаційної системи для обліку податків Шпанівської ОТГ»**

Виконав: Дем’янюк Михайло Андрійович

Студент навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки

Група КН-41

_____

Керівник: старший викладач кафедри комп’ютерних наук та прикладної математики Герус Володиммир Андрійович

_____

Рівне-2023

Національний університет водного господарства та природокористування

НН інститут автоматики, кібернетики та обчислювальної техніки

Кафедра **прикладної математики**

Освітньо-кваліфікаційний рівень **бакалавр**

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «ІнфорКомп'ютерні науки»**

«ЗАТВЕРДЖУЮ»

Завідувач кафедри

комп'ютерних наук та

прикладної математики

д.т.н., професор Турбал Ю.В

"____"_____2023року

ЗАВДАННЯ

КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дем'янюку Михайлу Андрійовичу

1. Тема роботи Створення інформаційної системи для обліку податків Шпанівської ОТГ

керівник роботи старший викладач кафедри комп'ютерних наук та прикладної математики Герус В.А.

затверджені наказом вищого навчального закладу від «19»квітня 2023 року С №-449

2. Термін подання роботи студентом 31.05.2023
3. Вихідні дані до роботи технології, що потрібні для розробки системи для обліку податків Шпанівської ОТГ.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) РОЗДІЛ 1. ОГЛЯД СИСТЕМ ОБЛІКУ

ПОДАТКІВ. НЕОБХІДНІСТЬ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ МОНІТОРИНГУ ОПЛАТИ ПОДАТКІВ.ОГЛЯД АНАЛОГІВ. РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СИСТЕМИ ОБЛІКУ ПОДАТКІВ. Розділ 3. РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ СИСТЕМИ ОБЛІКУ ПОДАТКІВ. РОЗДІЛ 4. ТЕСТУВАННЯ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ СИСТЕМИ ОБЛІКУ ПОДАТКІВ

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) мультимедійна презентація.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ1	старший викладач Герус В.А.	05.02.2023	05.02.2023
Розділ 2	старший викладач Герус В.А.	05.02.2023	05.02.2023
Розділ 3	старший викладач Герус В.А.	05.02.2023	05.02.2023
Розділ 4	старший викладач Герус В.А.	05.02.2023	05.02.2023

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вивчення літератури за обраною тематикою	05.03.2023-10.04.2023	Виконав
2	Формулювання постановки задачі	11.04.2023-14.04.2023	Виконав
3	Вибір стеку технологій	15.04.2023-21.04.2023	Виконав
4	Збір та аналіз даних	22.04.2023-10.05.2023	Виконав
5	Розробка алгоритму	11.05.2023-20.05.2023	Виконав
6	Розробка користувацького інтерфейсу	21.05.2023-01.06.2023	Виконав
7	Відлагодження програмного продукту	02.06.2023-05.06.2023	Виконав
8	Загальні висновки до роботи	06.06.2023-08.06.2023	Виконав
9	Підготовка звіту кваліфікаційної роботи	11.05.2023-24.06.2023	Виконав
10	Підготовка мультимедійної презентації	26.06.2023-27.06.2023	Виконав
11	Підготовка до виступу	26.06.2023-27.06.2023	Виконав

Студент _____Дем’янюк М. А.

(підпис) (прізвище та ініціали)

Керівник роботи _____Герус В.А.

(підпис) (прізвище та ініціали)

ЗМІСТ

РЕФЕРАТ

Кваліфікаційна робота: 53 с., 15 малюнків, 17 джерел.

Мета кваліфікаційної роботи полягає в проектуванні та розробці веб-додатку, який спрощує процес отримання інформації про заборгованість по уплаті за землю. Додаток повинен мати зручний і простий інтерфейс для користувачів, де вони можуть, ввівши свої персональні дані(РНОКПП), дізнатися заборгованість по уплаті за землю.

Об'єкт дослідження – Проектування та розробка веб-додатку для системи обліку податків.

Предмет дослідження – Функціональні вимоги та можливості веб додатку, що забезпечує отримання інформації про заборгованість по уплаті за землю.

Методи дослідження – Проектування архітектури та функціональності даного веб-додатку на основі аналізу вимог користувачів та особливостей систем обліку податків.

Результатом кваліфікаційної роботи є розроблений функціональний веб-додаток для системи обліку податків. Додаток було реалізовано використовуючи мову програмування Python та фреймворка Django. Розроблений додаток забезпечує достатньо зручну платформу, яка дозволяє користувачам дізнатися свою заборгованість по землі.

Ключові слова: PYTHON, DJANGO, ВЕБ-ДОДАТОК, ОБЛІК, ПОДАТКИ.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CSS - Cascading StyleSheets

HTML – HyperText Markup Language

JS – JavaScript

MVC – Model-View-Controller

АНОТАЦІЯ

Ця кваліфікаційна робота присвячена розробці та реалізації системи обліку податків, яка базується на веб-додатку з використанням мови програмування Python та фреймворку Django. Оптимальний збір та ефективна оцінка податків є важливими компонентами економічного розвитку та фінансової стабільності країни. З урахуванням складності податкової системи та зростання обсягів податкових даних, ефективний механізм обліку податків стає важливим завданням для державних органів та платників податків.

Метою цієї роботи є створення зручного та ефективного інструменту, який допоможе платникам податків відстежувати свою заборгованість, а державним органам - здійснювати контроль та моніторинг оплати податків. Дипломна робота складається з декількох розділів, які включають аналіз основних проблем, пов'язаних зі збором та обліком податків, опис архітектури системи, розробку функціональності, тестування системи та підсумки роботи.

У першому розділі роботи проводиться аналіз основних проблем, пов'язаних зі збором та обліком податків, виявляються недоліки традиційних методів обліку та висвітлюються потреби платників податків та державних органів у відповідних інструментах.

Другий розділ присвячений детальному опису архітектури системи обліку податків. Розглядається структура бази даних, визначаються моделі даних та їх взаємозв'язки. Окрема увага приділяється розробці функцій для взаємодії з базою даних, обробки запитів та відображення результатів на веб-сторінках. Розглядаються також питання забезпечення безпеки та конфіденційності даних у системі обліку податків.

Третій розділ присвячений реалізації системи обліку податків з використанням фреймворку Django. Детально описуються кроки розробки веб-додатку, включаючи створення моделей даних, налаштування маршрутів, розробку веб-сторінок та інтеграцію з базою даних. Надається інформація про використані бібліотеки та інструменти для розробки системи.

У четвертому розділі проводиться тестування розробленої системи обліку податків, оцінюється її функціональність, надійність та

продуктивність. Аналізуються результати тестування, виявлені проблеми та недоліки виправляються.

У заключному розділі підводяться підсумки роботи, формулюються висновки щодо досягнених результатів та надаються рекомендації щодо подальшого розвитку та вдосконалення системи обліку податків.

Ця дипломна робота спрямована на створення ефективного та надійного інструменту для обліку податків, який сприятиме фінансовій стабільності країни та полегшить процес оплати податків як для платників, так і для державних органів.

ВСТУП

Сучасне суспільство регулярно стикається з великими викликами у сфері оподаткування, адже оптимальне збирання та адекватна оцінка податків є необхідними компонентами економічного розвитку та фінансової стабільності країни. З урахуванням складності податкової системи та зростання обсягів податкових даних, ефективний механізм обліку податків стає актуальним завданням для державнимх органів та підприємств

Ця кваліфікаційна робота присвячена розробці та реалізації системи обліку податків на основі веб-додатку з використанням Python та фреймворку Django. Головною метою роботи є створення зручного та ефективного інструменту, який допоможе платникам податків відстежувати свою заборгованість, а державним органам – здійснювати контроль та моніторинг оплати податків.

У першому розділі роботи будуть проаналізовані основні проблеми, пов’язані зі збором та обліком податків. Будуть виявлені недоліки традиційних методів обліку та висвітлені потреби платників податків та державних органів у подібних інструментах. Також буде проведено аналіз аналогічних систем. Дослідження будуть зосереджені на виявленні прогалин у поточних підходах до обліку податків та обґрунтуванні необхідності впровадження нової системи обліку.

Другий розділ присвячений детальному опису архітектури системи обліку. Буде розглянуто податків структуру бази даних, визначення моделей даних та взаємозв’язки між ними. Окрему увагу буде приділено розробці функцій та взаємодії з базою даних, обробки запитів та відображення результатів на веб-сторінках. Також будуть розглянуті питання забезпечення безпеки та конфіденційності даних в системі обліку податків.

Третій розділ присвячений реалізації системи обліку податків з використанням фреймворку Django. Будуть детально описані кроки розробки веб-додатку, включаючи створення моделей даних, налаштування маршрутів та інтеграцію з базою даних. Також буде надана інформація про використані бібліотеки та інструменти для розробки системи.

У четвертому розділі буде проведено тестування розробленої системи обліку податків. Будуть оцінені функціональність надійність та продуктивність системи. Результати тестування будуть проаналізовані, а виявлені проблеми та недоліки будуть виправлені

У заключному розділі будуть проведені підсумки роботи та сформульовані висновки щодо досягнених результатів. Також будуть надані рекомендації щодо подальшого розвитку та вдосконалення системи обліку податків.

В цілому, ця система спрямована на створення системи обліку податків, яка буде забезпечувати ефективний та надійний інструмент для платників податків та державних органів. Розроблена система має на меті полегшити процес оплати податків, забезпечити точний облік та контроль оплати, а також сприяти фінансовій стабільності країни.

РОЗДІЛ І

ОГЛЯД СИСТЕМ ОБЛІКУ ПОДАТКІВ. НЕОБХІДНІСТЬ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ МОНІТОРИНГУ ОПЛАТИ ПОДАТКІВ.

ОГЛЯД АНАЛОГІВ

1.1 Система обліку податків та її особливості:

В сучасному суспільстві податки відіграють ключову роль фінансовому функціонуванні держави. Оптимальний збір та облік податків є необхідними елементами фінансової стабільності країни. Система обліку податків включає в себе процеси збору, обробки, зберігання та аналізу податкових даних. Вона спрямована на забезпечення точного обліку податків та заборгованості платників податків.

1.1.1 Означення системи обліку податків:

Система обліку податків — це комплексний механізм, що включає в себе процеси збору, обробки та аналізу податкових даних з метою забезпечення ефективного контролю та управління оплатою податків.

1.1.2 Особливості системи обліку податків:

- Збір та обробка різних видів податкових даних, включаючи податки на прибуток, податок на додану вартість, податок на нерухомість тощо.
- Встановлення і виконання правил та нормативів, пов’язаних з оплатою податків.
- Контроль за виконанням податкових зобов’язань та виявлення порушень.
- Взаємодія з платниками податків та державними органами з метою обміну інформацією та врегулювання питань, пов’язаних з оплатою податків.
- Забезпечення конфіденційності та безпеки податкової інформації.

1.2 Класифікація систем обліку податків:

За типом типом платників податків:

- Системи обліку податків для фізичних осіб.
- Системи обліку податків для юридичних осіб.
- Системи обліку податків для само зайнятих осіб.

За видами податків:

- Системи обліку податку на прибуток.
- Системи обліку податку на додану вартість.
- Системи обліку на нерухомість.
- Системи обліку акцизного податку та інших спеціальних податків.

За масштабом використання:

- Локальні системи обліку податків для конкретного регіону чи муніципалітету.
- Національні системи обліку податків, що охоплюють всю країну.

1.2.1 Системи обліку податків для фізичних осіб:

Системи обліку податків для фізичних осіб є важливою складовою сучасної податкової системи багатьох країн. Основним завданням таких систем є забезпечення збору та обробки податкових даних фізичних осіб, а також контроль за своєчасною сплатою податків.

Деякі ключові аспекти систем обліку податків для фізичних осіб включають:

- Реєстрація платників податків: Системи обліку податків для фізичних осіб зазвичай передбачають процедуру реєстрації платників податків, під час якої фізичні особи подають свої персональні дані та отримують унікальний ідентифікаційний код.
- Збір та обробка податкових декларацій: Платники податків зобов’язані подавати податкові декларації, в яких вони вказують свої доходи, витрати та інші релевантні податкові дані. Системи обліку податків включають механізми для збору та обробки декларацій, а також перевірку їх на достовірність та виявлення потенційних порушень.
- Розрахунок податків: На основі даних, вказаних у податковій декларації, система обчислює податкове зобов’язання фізичної особи
- Подання звітності: Фізичні особи зобов’язані подавати встановлену законодавством звітність, яка може включати річну звітність, звіти про доходи, витрати, майно та інші фінансові показники.
- Сплата податків: Фізичні особи повинні своєчасно сплачувати податки згідно з розрахунками, визначеними системою обліку податків.
- Взаємодія з податковими органами: Фізичні особи можуть взаємодіяти з іншими податковими органами щодо запитів, перевірок та інших процедур.

Система обліку податків для фізичних осіб дозволяє забезпечити правильність обліку та сплату податків, уникнути порушень податкового законодавства та забезпечити виконання податкових обов'язків фізичних осіб.

1.2.2 Системи обліку податків для юридичних осіб:

Системи обліку податків для юридичних осіб грають ключову роль у сфері оподаткування бізнесу. Вони забезпечують точний облік та контроль за сплатою податків юридичними особами.

Основні аспекти систем обліку податків для юридичних осіб включає:

- Реєстрація юридичних осіб: При створенні нового підприємства або реєстрації нової юридичної особи в системі обліку податків проводиться процедура реєстрації. Під час реєстрації фіксуються реквізити підприємства, його вид діяльності та інші важливі дані.
- Збір та обробка фінансової звітності: Юридичні особи зобов'язані подавати фінансову звітність, яка містить інформацію про їх фінансові результати та стан. Системи обліку податків забезпечують збір та обробку цієї звітності, а також перевірку її на відповідність податковому законодавству.
- Розрахунок та сплата податків: Основним завданням систем обліку податків для юридичних осіб є розрахунок сум податків, які повинні бути сплачені. Це включає рорахунок податку на прибуток, ПДВ, акцизів та інших видів податків відповідно до законодавства. Після розрахунку юридичні особи здійснюють сплату податків відповідно до встановленого графіку.
- Контроль та аудит: Система обліку податків для юридичних осіб також включають механізми контролю та аудиту. Вони дозволяють перевіряти правильність розрахунку та сплати податків, виявляти потенційні порушення та проводити перевірки згідно з податковим законодавством.

Ці системи забезпечують високу точність та ефективність обліку податків для юридичних осіб, допомагають зменшити ризики порушення податкових правил та сприяють виконанню податкових зобов'язань.

1.2.3 Системи обліку податків для самозайнятих осіб:

Система обліку податків для самозайнятих осіб є комплексом правил, процедур і інструментів, які використовуються для збору, обробки, аналізу та звітності щодо податкових даних самозайнятих осіб.

Самозайняті особи, такі як фрілансери, підприємці, власники малого бізнесу та інші особи, які працюють самостійно, зобов'язані виконувати свої податкові обов'язки перед державою. Система обліку податків для них допомагає відстежувати доходи, витрати, обчислювати податки та здійснювати звітність перед податковими органами.

Основна мета системи обліку податків для самозайнятих осіб полягає в забезпеченні точного та своєчасного визначення податкових зобов’язань, уникненні податкових порушень і спрощенні процесу оподаткування. Вона дозволяє самозайнятим особам вести незалежний облік своїх фінансових операцій, розрахувати відповідні податки та здійснювати своєчасну сплату.

Система обліку податків для самозайнятих осіб може включати такі елементи:

- Реєстрація самозайнятих осіб у податкових органах та отримання ідентифікаційних номерів.
- Ведення податкового обліку доходів та витрат.
- Розрахунок податків, їх сплата та подання податкової звітності.
- Дотримання податкових обов’язків, включаючи строкову звітність та сплату податків.
- Взаємодія з податковими органами, відповідь на запити та проведення перевірок.

Ця система має важливе значення для самозайнятих осіб, оскільки допомагає забезпечити відповідність податковому законодавству, уникнути штрафів та санкцій, а також зберегти час та ресурси, пов’язані з податковими обов’язками.

1.3 Переваги та недоліки систем обліку податків:

Переваги систем обліку податків:

1. Точність і надійність: Система обліку податків дозволяє вести детальний облік фінансових операцій і транзакцій, що допомагає забезпечити точний розрахунок податкового зобов'язання та звітність. Це сприяє надійності і достовірності фінансової інформації, яка подається податковим органам органам.
2. Ефективність та автоматизація: Система обліку податків дозволяє автоматизувати багато процесів, пов’язаних з розрахунком податків. Це допомагає зменшити час і зусилля, необхідні для виконання

податкових обов’язків, і забезпечує більш ефективне використання ресурсів.

3. Відповідність законодавству: Система обліку податків дозволяє відстежувати зміни в податковому законодавстві і забезпечує виконання всіх необхідних вимог. Це допомагає уникати порушень і санкцій з боку податкових органів.
4. Зручність і доступність: Система обліку податків може бути розроблена зручним і доступним інтерфейсом, що спрощує взаємодію з нею. Вона може бути доступна з будь-якого місця за допомогою Інтернету, що дозволяє фізичним особам ефективно керувати своїми податкові обов’язки.

Недоліки системи обліку податків:

1. Складність: Ведення і управління системою обліку податки може бути складним завданням, особливо для недосвідчених користувачів. Вимагається розуміння податкового законодавства і правил, а також навичок роботи з відповідними програмними інструментами.
2. Вартість: Розробка і впровадження системи обліку податків може бути затратною процедурою, особливо для невеликих підприємств або фізичних осіб з обмеженими фінансовими ресурсами. Потрібно брати до уваги витрати на програмне забезпечення, обладнання та навчання персоналу.
3. Технічні проблеми: Система обліку податків може стикатися з технічними проблемами, такими як відмови обладнання, помилки програмного забезпечення або проблеми зі збереженням даних. Це може призвести до перебоїв у роботі і втраті даних, що може вплинути на податкову звітність та виконання податкових обов’язків.
4. Залежність від змін податкового законодавства: Податкове законодавство може підлягати частим змінам і поправкам. Це може вимагати постійного оновлення системи обліку податків для врахування нових правил та вимог. Недотримання змін може призвести до порушень і санкцій з боку податкових органів.

Незважаючи на недоліки, система обліку податків є важливим інструментом для забезпечення відповідності податковому законодавству, точного розрахунку податкового зобов’язання та зменшення ризику порушень та санкцій.

1.4 Структура побудови систем обліку податків:

Система обліку податків має складну структуру, яка включає різні компоненти та елементи, що взаємодіють між собою. Основними компонентами структури системи обліку податків є:

Податкові закони і правила: Це набір правил і положень, які регулюють оподаткування і визначають обов'язки платників податків. Податкові закони встановлюють види податків, ставки оподаткування, порядок подання декларацій та сплати податків.

Податкові органи: Це державні органи, відповідальні за збір податків та контроль за дотриманням податкового законодавства. Вони здійснюють обробку податкової інформації, перевірки платників податків і виконання податкових обов'язків.

Реєстри платників податків: Це бази даних, в яких зберігається інформація про платників податків, їхні обов'язки та історія оплати податків. Реєстри платників податків використовуються для ідентифікації платників, перевірки правильності розрахунку податків і ведення податкової статистики.

Податкові декларації: Це форми, які заповнюють платники податків для повідомлення про свій дохід, витрати та іншу податкову інформацію. Податкові декларації подаються регулярно, зазвичай щороку або за певний період, і використовуються для розрахунку суми податку, яку платник повинен сплатити.

Системи обліку і звітності: Це комп'ютерні системи, які допомагають автоматизувати облік податків та підготовку податкової звітності. Вони включають бази даних, програмне забезпечення для обробки податкової інформації та інструменти для генерації звітів і документів.

Контрольні механізми: Це системи та процедури, призначені для контролю за дотриманням податкових правил і виявлення податкових порушень. Контрольні механізми включають перевірки, аудити, аналіз ризиків та інші заходи для забезпечення виконання податкового законодавства.

Податкова звітність і комунікація: Це процеси передачі податкової інформації між платниками податків, податковими органами та іншими зацікавленими сторонами. Включає в себе подання декларацій, сплату податків, отримання податкових повідомлень та листів, а також комунікацію з податковими органами щодо податкових питань.

Ця структура побудови систем обліку податків забезпечує ефективне функціонування системи, збір і обробку податкової інформації, а також контроль за дотриманням податкового законодавства. Вона дозволяє забезпечити точний розрахунок податків, зменшити ризики податкових порушень та забезпечити взаємодію між платниками податків і податковими органами.

1.5 Розвиток систем обліку податків:

Системи обліку податків мають багатовікову історію, що починається з появи податків як засобу збору фінансових ресурсів для держави. Ось кілька важливих етапів розвитку систем обліку податків:

1. Античні цивілізації: У давніх цивілізаціях, таких як Месопотамія, Давній Єгипет, Стародавня Греція та Римська імперія, були введені податки для фінансування державних потреб. Облік податків здебільшого відбувався вручну, де було збиралися дані про платників та їхні зобов'язання.
2. Середньовіччя: У середньовіччі, податки були широко використовувані для підтримки феодальної системи. Облік податків став більш складним, оскільки з'явилися нові види податків та складніші методи їх збору. Записи про платежі та зобов'язання велися у рукописних книгах та реєстрах.
3. Індустріальна революція: З появою індустріальної революції в 18-19 століттях, системи обліку податків стали більш складними і масштабними. Введення нових видів податків, таких як податок на прибуток, стало потребою держав для фінансування інфраструктури та соціальних програм. У цей період з'явилися перші організовані системи обліку податків, включаючи ведення реєстрів, відомостей про платників та методи розрахунку податкових зобов'язань.
4. Комп'ютеризація та інтернет-епоха: З появою комп'ютеризації та Інтернету в 20 столітті, системи обліку податків пройшли велику трансформацію. Впровадження комп'ютерних систем та баз даних дозволило автоматизувати багато процесів, спростити облік податків та забезпечити швидкий доступ до податкової інформації. Електронне

подання декларацій та сплати податків стало стандартом у багатьох країнах.

Крім історичного розвитку, системми обліку податків пройшли через такі процеси:

1. Законодавчі зміни: Податкове законодавство постійно змінюється відповідно до змін у економіці, фіскальних потребах та політичних рішень. Розвиток систем обліку податків пов'язаний з необхідністю врахувати та впровадити нові правила та вимоги, що можуть стосуватися розрахунку податків, звітності та способу взаємодії з податковими органами.
2. Технологічні інновації: Технологічний прогрес впливає на розвиток систем обліку податків. Впровадження сучасних інформаційних технологій, хмарних рішень, штучного інтелекту та аналітики даних дозволяє автоматизувати процеси обліку та звітності, поліпшувати точність та швидкість розрахунків та сприяти забезпеченню відповідності податковим правилам.
3. Глобалізація: В епоху глобалізації підприємства та фізичні особи все більше взаємодіють з податковими органами різних країн. Це ставить вимоги до розвитку систем обліку податків, які повинні враховувати міжнародні стандарти звітності, міжнародні договори та вимоги різних юрисдикцій.
4. Автоматизація та інтеграція: Розвиток систем обліку податків спрямований на автоматизацію та інтеграцію процесів. Це означає забезпечення безперервного потоку даних між різними системами, такими як облікова система, система звітності та система оподаткування. Інтеграція дозволяє зменшити помилки, забезпечити консистентність даних та підвищити ефективність процесів обліку податків.
5. Застосування аналітики: Розвиток систем обліку податків також пов'язаний зі зростанням застосування аналітики даних. Аналітичні інструменти дозволяють проводити детальний аналіз фінансової інформації, ідентифікувати ризики та можливості, робити прогнози та приймати обґрунтовані рішення в галузі оптимізації оподаткування та фінансового планування.
6. Забезпечення відповідності та контроль: Розвиток систем обліку податків також спрямований на забезпечення відповідності податковим правилам та політикам. Податковий контроль і перевірки стають більш суворими, тому системи обліку податків повинні бути готовими до

надання детальної інформації та документації при запиті податкових органів.

Сучасний розвиток систем обліку податків спрямований на постійне удосконалення технологій, забезпечення точності та ефективності розрахунку податків, а також зменшення адміністративних навантажень для платників податків. Застосування штучного інтелекту, блокчейн-технологій та аналітики даних в системах обліку податків дозволяє покращити контроль, зменшити ризики шахрайства та сприяти розвитку сучасних податкових політик.

Розвиток систем обліку податків є невід'ємною частиною еволюції фінансового управління та податкової системи. Цей процес постійно адаптується до змін у соціально-економічному середовищі та технологічному прогресі з метою забезпечення ефективного збору податків та створення стійких фінансових основ для розвитку країни.

1.6 Необхідність розробки веб-додатку для системи обліку податків:

Сучасний світ характеризується швидким розвитком технологій і зростаючою важливістю електронних сервісів. В контексті системи обліку податків, розробка веб-додатку стає необхідністю, оскільки надає ряд переваг і сприяє поліпшенню процесів збору та обробки податкової інформації. Нижче розглянемо детальніше необхідність розробки веб-додатку для системи обліку податків.

1. Автоматизація процесів: Розробка веб-додатку дозволяє автоматизувати багато процесів, пов'язаних з обліком податків. Це включає подання декларацій, розрахунок сум податку, контроль за виконанням податкових обов'язків та генерацію податкових звітів. Автоматизація допомагає зменшити ризик помилок, збільшити швидкість обробки інформації та полегшити взаємодію між платниками податків і податковими органами.
2. Зручність та доступність: Веб-додаток надає зручний спосіб доступу до системи обліку податків. Платники податків можуть легко отримати доступ до своєї податкової інформації, перевірити свою заборгованість, подати декларації та отримати податкові повідомлення в будь-який зручний для них час і місце. Це сприяє зменшенню адміністративних перешкод і поліпшує сервіс для платників податків.
3. Точність та надійність: Веб-додаток дозволяє забезпечити більшу точність та надійність обробки податкової інформації. Всі дані

вводяться безпосередньо в систему, що зменшує ризик помилок, пов'язаних з ручним введенням інформації. Крім того, веб-додаток може мати вбудовані механізми перевірки та контролю за правильністю введення даних, що забезпечує високу якість обробки податкової інформації.

4. Збереження часу та ресурсів: Розробка веб-додатку дозволяє зменшити зусилля та ресурси, необхідні для проведення податкових процедур. Платники податків можуть виконувати багато операцій самостійно, без необхідності фізично відвідувати податкові органи. Це економить час і зусилля як для платників податків, так і для податкових органів.
5. Звітність та аналітика: Розроблений веб-додаток може забезпечити широкі можливості звітності та аналітики. Він дозволяє генерувати різноманітні звіти і аналітичні дані, які допомагають зрозуміти податкову ситуацію, виявити тренди та здійснити аналіз для прийняття керівних рішень.
6. Забезпечення безпеки: Розробка веб-додатку дозволяє забезпечити високий рівень безпеки податкової інформації. Відповідні заходи безпеки, такі як шифрування даних, автентифікація, контроль доступу та інші, можуть бути впроваджені для захисту конфіденційності та цілісності даних платників податків.

Розробка веб-додатку для системи обліку податків є важливим етапом у вдосконаленні податкової системи. Вона сприяє автоматизації процесів, зручності та доступності для платників податків, поліпшенню точності та надійності обробки податкової інформації, збереженню часу та ресурсів, забезпеченню звітності та аналітики, а також забезпеченню високого рівня безпеки даних.

1.7 Аналогічні системи:

Існує багато аналогічних веб-додатків для обліку податків, які надають різноманітні функції та можливості. Ось кілька популярних веб-додатків для обліку податків:

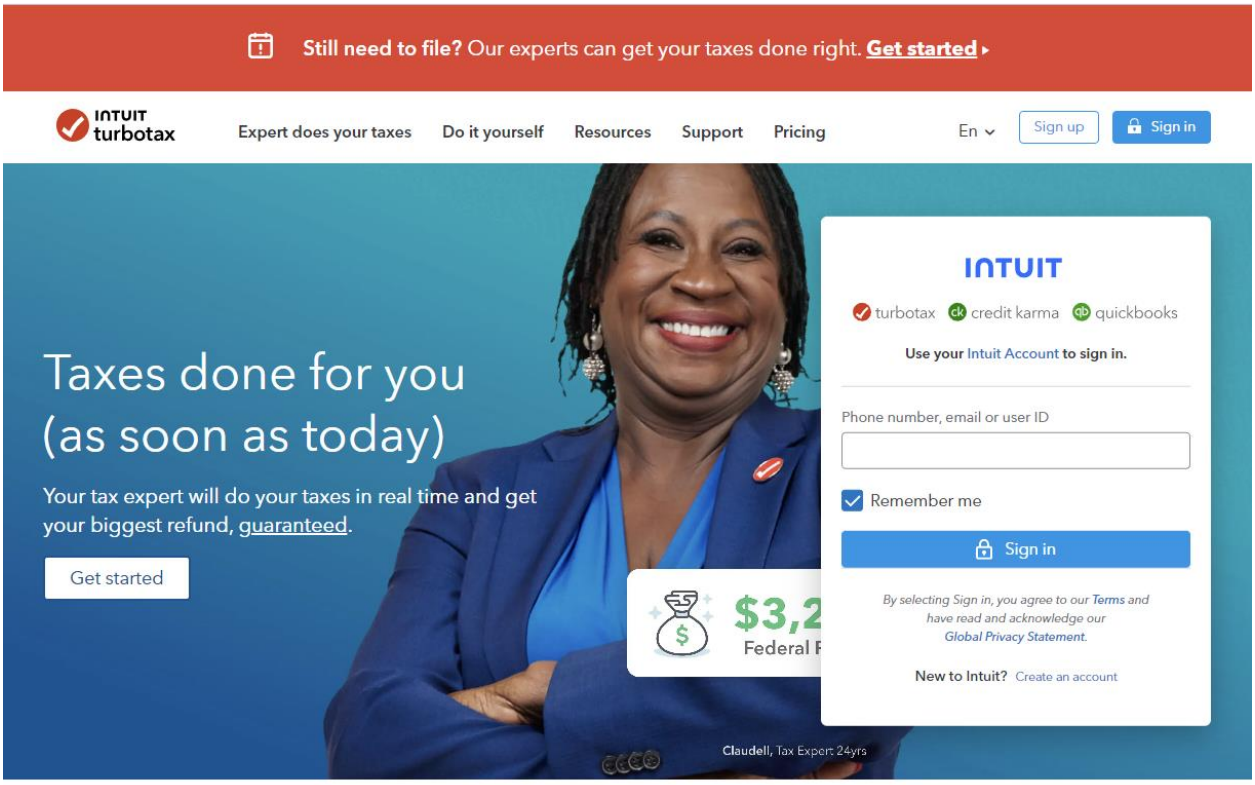
1. TurboTax: TurboTax є одним з найвідоміших веб-додатків для обліку податків. Він надає можливість самостійно заповнювати податкові декларації, розраховувати податкові зобов'язання та забезпечує детальні пояснення щодо податкових правил і законів.

2. H&R Block: H&R Block є іншим популярним веб-додатком для обліку податків. Він надає можливість заповнювати податкові декларації, вираховувати податкові зобов'язання та отримувати підказки та поради щодо максимізації повернень податків.
3. TaxJar: TaxJar є веб-додатком, який допомагає автоматизувати облік і розрахунок податків з великої кількості джерел доходу, включаючи електронну комерцію, продажі на майданчиках і розподілені операції.

Ці веб-додатки дозволяють платникам податків самостійно виконувати податкові процедури, заповнювати декларації, розраховувати податкові зобов'язання та отримувати необхідну підтримку та поради. Кожен з них має свої унікальні особливості та інтерфейс, що дозволяє користувачам знайти найбільш зручний варіант для своїх потреб у обліку податків.

1.7.1 Огляд веб-додатку “TurboTax”:

TurboTax є одним з найпопулярніших і надійних веб-додатків для обліку податків. Він розроблений компанією Intuit і використовується мільйонами користувачів по всьому світу. TurboTax пропонує широкий спектр функцій і інструментів, які допомагають платникам податків легко та точно заповнити свої податкові декларації.



Count on America’s #1 tax prep provider

Рис. 1.7.1.1 Головна сторінка сайту TurboTax

Одною з особливостей сайту TurboTax є простота використання. TurboTax має інтуїтивно зрозумілий і легкий у використанні інтерфейс. Він крок за кроком проводить користувачів через процес заповнення декларації та розрахунку податкових зобов'язань.

Також TurboTax підтримує різні типи податків, включаючи податок на дохід, підприємницький податок, податок на нерухомість, податок на продажі та інші. Він пропонує спеціалізовані підходи для кожного типу податку, забезпечуючи точність і відповідність до податкових правил.

Однією з головних особливостей даного веб-додатку є розрахунок оптимального повернення податків: TurboTax має розумний алгоритм, який допомагає знайти всі можливі витрати та знижки, що дозволяють максимізувати повернення податків. Він автоматично розраховує оптимальну стратегію для зниження податкових зобов'язань.

Наступною особливістю є експертна оцінка. TurboTax надає користувачам доступ до експертів у галузі оподаткування. Користувачі можуть отримувати підтримку та консультації від професіоналів у разі виникнення питань або складних ситуацій.

Останньою, але не менш важливою особливістю є можливість попереднього перегляду. TurboTax надає можливість попереднього перегляду та перевірки податкової декларації перед її поданням. Це дозволяє упевнитися в правильності та точності заповнення, а також уникнути можливих помилок або пропусків.

Хоча TurboTax є платним сервісом, проте він надає безкоштовну версію для простих податкових ситуацій. Для складніших випадків доступні платні пакети з додатковими функціями та підтримкою.

Незважаючи на свою популярність і багатий функціонал, TurboTax також має деякі недоліки, які варто врахувати:

1. Вартість: TurboTax є комерційним продуктом і вимагає певні витрати для його використання. Вартість може залежати від типу податку та обсягу необхідних функцій. Деякі користувачі можуть вважати, що витрати на використання TurboTax є високими порівняно з іншими доступними альтернативами.

2. Залежність від інтернет-підключення: TurboTax є веб-додатком, що означає, що для його використання потрібне постійне і стабільне підключення до Інтернету. Це може створювати проблеми для користувачів, які мають обмежений або непостійний доступ до Інтернету.
3. Складність для складних ситуацій: В хоча TurboTax надає розширений функціонал, в деяких складних податкових ситуаціях він може не забезпечити повнісню підтримку. У таких випадках може знадобитися додаткова професійна консультація або використання інших спеціалізованих програм.
4. Обмежена гнучкість: TurboTax має певні обмеження щодо налаштування та адаптації до конкретних потреб користувача. Він пропонує стандартні шаблони та підходи для заповнення податкових декларацій, але може бути менш гнучким у порівнянні з іншими програмами або професійними консультантами.
5. Обмежена локалізація: TurboTax переважно спрямований на американський ринок та американську податкову систему. Деякі функції та правила можуть бути обмеженими або неідентичними для інших країн. Тому, якщо ви не знаходитесь в США, варто переконатися, що TurboTax підтримує вашу податкову систему.

Враховуючи ці недоліки, важливо зробити об'єктивну оцінку і вибрати той інструмент, який найкраще відповідає вашим потребам і особистим обставинам.

1.7.2 Огляд веб-додатку “H&R Block”:

H&R Block є однією з провідних компаній, яка надає послуги з обліку податків і фінансового планування. Основна сфера діяльності H&R Block - це підтримка фізичних осіб і бізнесів у заповненні податкових декларацій та забезпечення відповідності податковим вимогам.

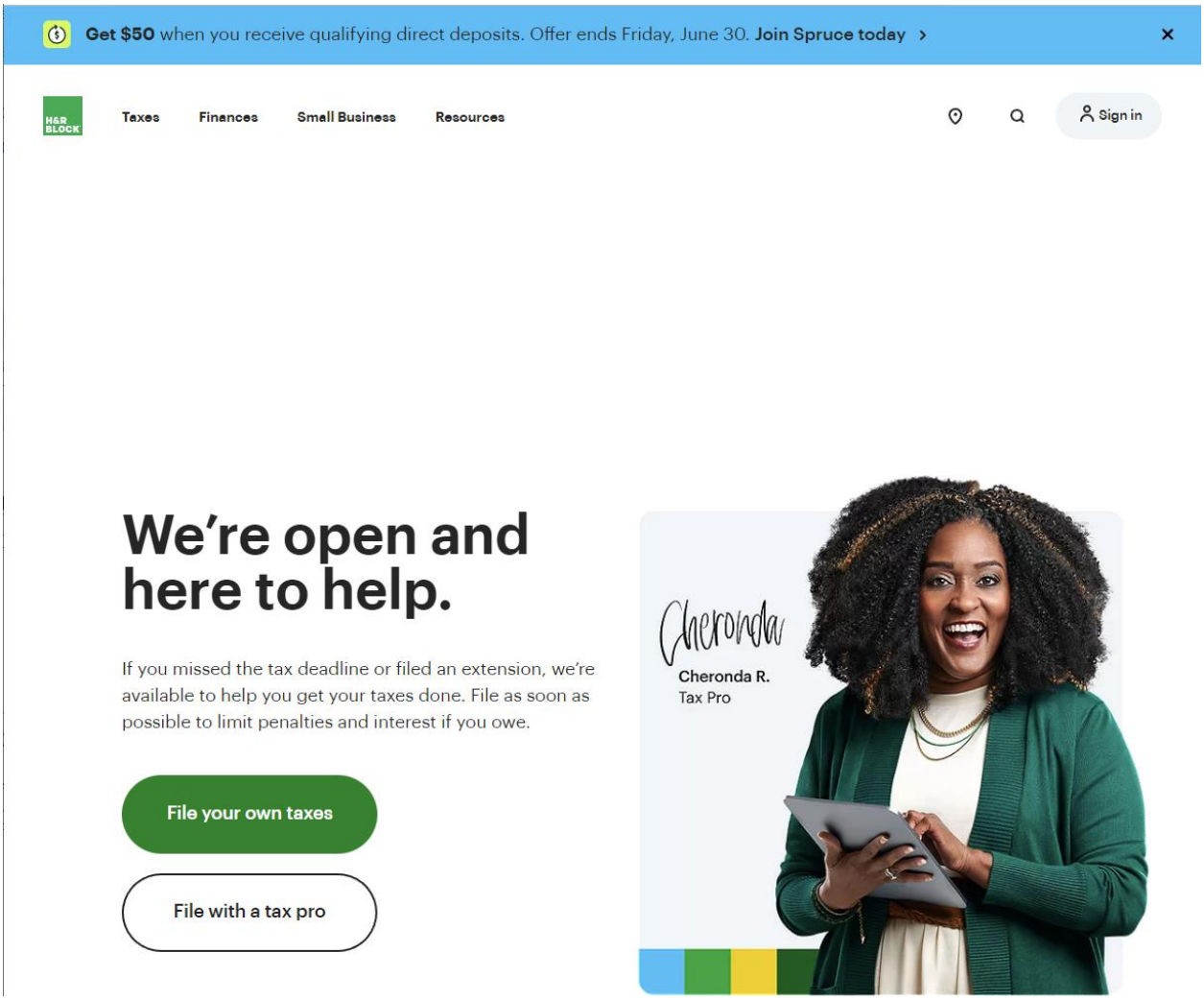


Рис. 1.7.2.1 Головна сторінка сайту H&R Block

Основні особливості та послуги H&R Block включають:

1. Онлайн-податкове планування: H&R Block надає можливість заповнювати податкові декларації онлайн. Користувачі можуть вибрати підходящий податковий план та використовувати вбудовані інструменти для заповнення та розрахунку податків.
2. Професійна підтримка: Компанія має широку мережу професіоналів, таких як бухгалтери та юристи, які надають консультації та підтримку з податкових питань. Користувачі можуть отримати персоналізовані поради щодо своєї конкретної ситуації.
3. Аудиторська підтримка: H&R Block надає підтримку під час аудиту податкових декларацій. Вони можуть допомогти у взаємодії з податковими органами та надати необхідні документи та пояснення.
4. Податкові калькулятори та інструменти: H&R Block надає різні онлайн-калькулятори та інструменти для розрахунку податків, перевірки вирахувань і оцінки впливу різних факторів на податкову ситуацію.

Безпека та конфіденційність: H&R Block забезпечує високий рівень безпеки та конфіденційності в обробці податкових даних. Вони використовують захисні технології та протоколи, щоб забезпечити безпеку особистої інформації своїх клієнтів.

Не дивлячись на те, що H&R Block є визнаною компанією з обліку податків і надає широкий спектр послуг, вона також має свої недоліки. Ось кілька недоліків, які можуть бути пов'язані з використанням H&R Block:

Вартість: Послуги H&R Block можуть бути відносно дорогими порівняно з іншими онлайн-платформами для заповнення податкових декларацій. Це може бути особливо проблематично для людей з обмеженим бюджетом.

Обмежена географічна наявність: В деяких країнах або регіонах H&R Block може бути обмеженою або недоступною. Це означає, що не всі користувачі зможуть скористатися їхніми послугами.

Відсутність персонального підходу: Використання онлайн-платформи, такої як H&R Block, може означати менш особистий підхід до обліку податків. Ви можете не мати можливості спілкуватися безпосередньо з бухгалтером або фахівцем з податків, що може бути важливо для складних ситуацій або індивідуальних запитів.

Обмежений функціонал: В деяких випадках H&R Block може бути менш гнучким у порівнянні зі спеціалізованими податковими програмами або системами. Він може не мати всіх необхідних функцій для конкретних податкових сценаріїв або можливостей налаштування.

Залежність від точності введених даних: Як і в будь-якій системі обліку податків, точність введених даних є важливим чинником. Хоча H&R Block надає різні перевірки та контролі, він може покладатися на користувачів щодо правильного введення податкової інформації.

Важливо пам'ятати, що недоліки H&R Block можуть бути відносними і залежати від індивідуальних потреб і вимог користувача. Рекомендується здійснити власну оцінку і порівняти різні рішення перед вибором найбільш підходящого для ваших потреб.

1.7.3 Огляд веб-додатку “TaxJar”:

TaxJar - це веб-додаток для автоматизації обліку та розрахунку податків. Він призначений для бізнесів, які мають різні джерела доходу,

включаючи електронну комерцію, продажі на майданчиках та розподілені операції.

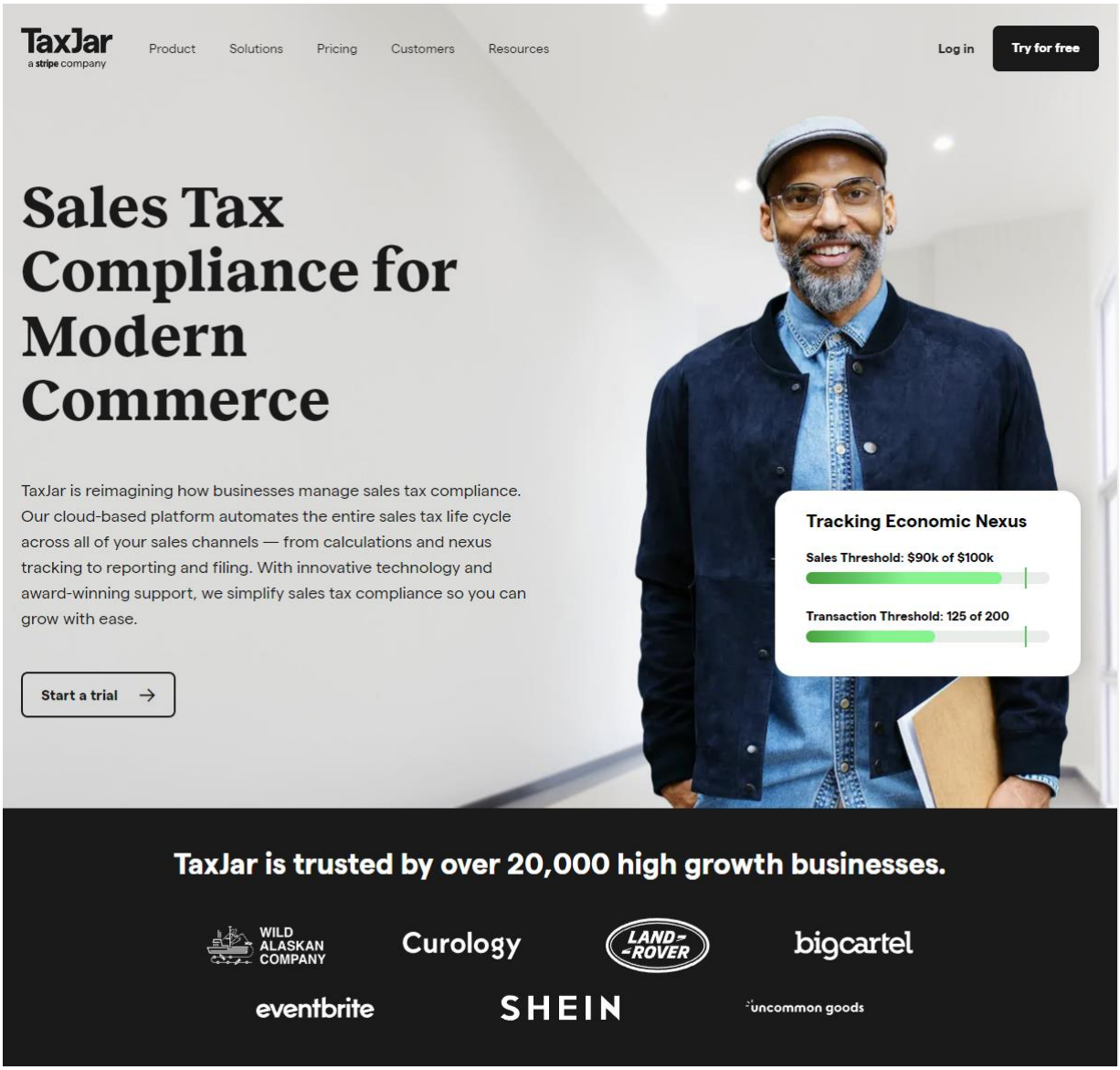


Рис. 1.7.3.1 Головна сторінка сайту TaxJar

Основні особливості та функціонал TaxJar включають:

Автоматичний розрахунок податків: TaxJar використовує розумні алгоритми, щоб автоматично розрахувати податки на основі вашої діяльності, місцезнаходження, видів товарів та інших факторів.

Отримання ставок податку: Система TaxJar забезпечує точні ставки податку для різних юрисдикцій, у тому числі штатів, округів, міст та інших адміністративних одиниць.

Заповнення податкових звітів: TaxJar автоматично генерує податкові звіти, такі як форми продажу та використання, що спрощує процес подання податкової звітності.

Інтеграція з платформами електронної комерції: TaxJar підтримує інтеграцію з популярними платформами електронної комерції, такими як Shopify, Magento, WooCommerce та іншими, щоб автоматично отримувати дані про продажі та податки.

Моніторинг меж перевищення ліміту: TaxJar допомагає відстежувати обсяги продажів та дотримуватись вимог про перевищення ліміту для реєстрації податку в різних юрисдикціях.

Хоча TaxJar є потужним інструментом для автоматизації обліку та розрахунку податків, він також має деякі недоліки, які варто враховувати:

Вартість: Використання TaxJar може бути витратним, особливо для невеликих бізнесів або початківців. Вартість планів TaxJar може бути вищою порівняно з іншими подібними сервісами, що може становити фінансове виклик.

Обмежена географічна підтримка: Незважаючи на те, що TaxJar підтримує багато юрисдикцій, існують певні обмеження в їхній географічній підтримці. Деякі менш розповсюджені регіони або міжнародні ринки можуть бути недоступні або мати обмежені можливості в TaxJar.

Складність налаштування: Враховуючи широкий функціонал TaxJar, налаштування та конфігурація системи можуть вимагати певного часу та ресурсів. Навчання персоналу та налаштування можуть бути складними для менш технічно освічених користувачів.

Залежність від інтеграцій: Для повноцінного використання TaxJar необхідно налагоджувати інтеграцію з платформами електронної комерції та іншими системами. Це може бути складним завданням, особливо якщо ви вже маєте налаштовані системи, які потребують синхронізації з TaxJar.

Обмеження функціоналу в основних планах: Деякі розширені функції та можливості TaxJar можуть бути доступні лише в преміум-планах, що може обмежити функціональність користувачів базових або менших планів.

Будьте впевнені, що переглянули всі деталі та вимоги, перш ніж прийняти рішення про використання певної програми обліку податків.

Розділ ІІ

ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СИСТЕМИ ОБЛІКУ ПОДАТКІВ

2.1 Стек використаних технологій

Архітектура системи обліку податків, яка була створена для даної кваліфікаційної роботи, базується на веб-додатку з використанням фреймворку Django. Основні компоненти архітектури включають клієнтську сторону, серверну сторону і базу даних. Давайте детальніше розглянемо кожен з цих компонентів:

1. Клієнтська сторона:
- Фронтенд: На фронтенді використовується HTML, CSS і JavaScript для створення користувацького інтерфейсу. Ми можемо використовувати фреймворк Django Templates або розширення, таке як Django REST Framework для створення динамічних сторінок.
- Клієнтський JavaScript: Ми можемо використовувати JavaScript для взаємодії з сервером за допомогою AJAX-запитів, а також для реалізації додаткової функціональності, такої як валідація форм, динамічні елементи і т.д.
2. Серверна сторона:
- Django: Django є основним фреймворком, який використовується для розробки серверної частини. Він надає нам потужні інструменти для обробки запитів, маршрутизації, управління користувачами, доступом до бази даних і багато іншого.
- Контролери (Views): У Django ми використовуємо контролери для обробки запитів і взаємодії з моделями та шаблонами. Контролери приймають дані з клієнта, обробляють їх, виконують необхідні операції та повертають результати на клієнт.
- Моделі: Моделі відображають структуру даних та бізнес-логіку системи. Ми можемо використовувати ORM (Object-Relational Mapping) Django для взаємодії з базою даних, визначення моделей та виконання операцій CRUD (створення, читання, оновлення, видалення) з даними.
- Шаблони: У Django ми можемо використовувати шаблони для розмітки сторінок і відображення динамічного контенту. Ми можемо використовувати шаблонну мову Django для вставки даних з контролерів та передачі їх на клієнт.

3. База даних:

- SQLite: У нашому випадку ми використовуємо базу даних SQLite, яка є вбудованою базою даних Django. SQLite є легковаго, одноразовим файлом бази даних, що підходить для розробки і випробування на локальному середовищі. Проте, в реальному виробничому середовищі можна використовувати різні бази даних, такі як PostgreSQL, MySQL або Oracle, залежно від потреб проекту.

Проаналізувавши декілька мов програмування було прийнято рішення використати Python.

Django - це високорівневий веб-фреймворк розробки на мові програмування Python. Він надає потужні інструменти для швидкої і ефективної розробки веб-додатків. Ось деякі ключові особливості і переваги Django:

Перша причина, скорочення часу розробки: Django пропонує набір готових модулів і бібліотек, які спрощують створення веб-додатків. Це включає систему аутентифікації, управління формами, маршрутизацію URL-адрес та багато іншого. Готові компоненти дозволяють розробникам ефективно використовувати час і зосередитись на бізнес-логіці своїх додатків.

Друга причина, масштабованість: Django розроблений з урахуванням масштабованості веб-додатків. Він підтримує горизонтальне та вертикальне масштабування, що дозволяє розширювати додаток залежно від зростаючих потреб.

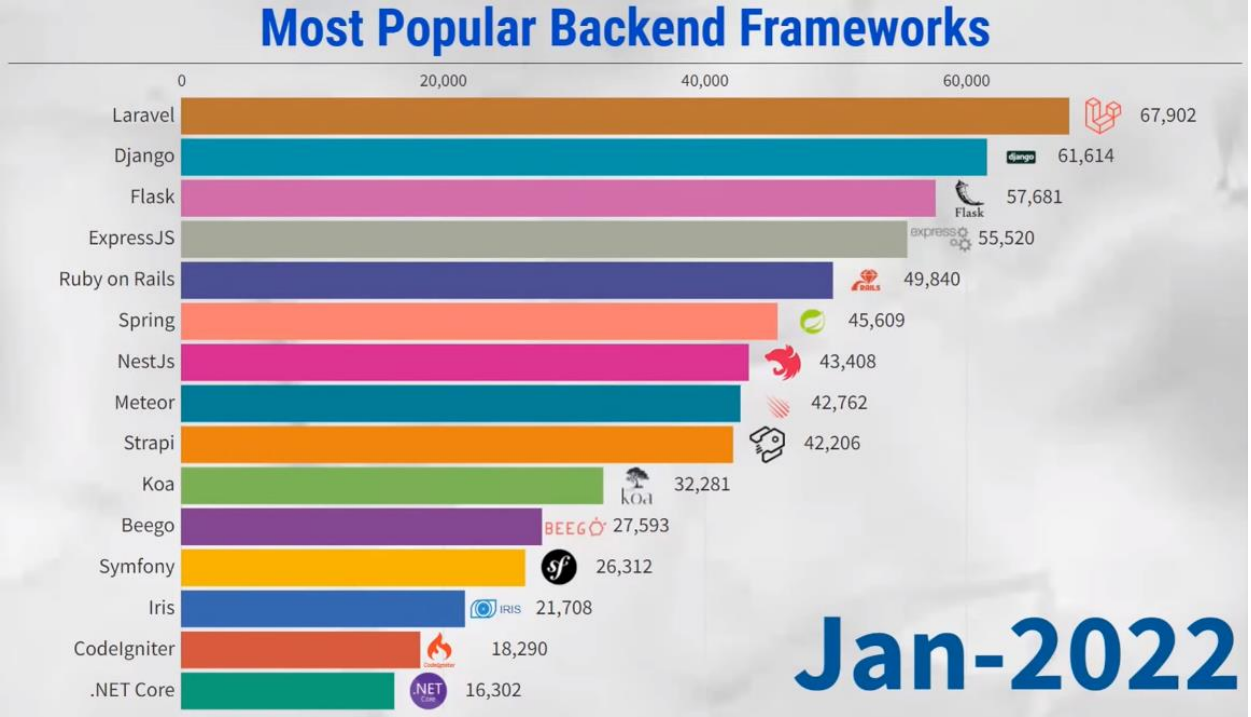
Третя причина, безпека: Django має вбудовані механізми безпеки, що дозволяють захищати веб-додатки від таких загроз, як внедрення SQL-запитів, перекривання CSRF, вразливості XSS та інші. Вбудована система аутентифікації та авторизації спрощує управління доступом користувачів.

Четверта причина, ORM (Object-Relational Mapping): Django використовує ORM для взаємодії з базами даних. Це дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід, замість написання складних SQL-запитів. ORM спрощує роботу з базою даних і забезпечує більшу переносимість додатків між різними типами баз даних.

Ну і п’ята причина, підтримка спільноти: Django має широку та активну спільноту розробників. Це означає, що ви зможете знайти багато

ресурсів, документацію, приклади коду та готові рішення для вирішення різних завдань. Спільнота також активно внесе покращення до фреймворку та забезпечує оновлення та підтримку.

Django - це потужний і популярний фреймворк, який пропонує зручні інструменти для розробки веб-додатків на мові програмування Python. Він є вибором багатьох розробників завдяки своїй простоті використання, гнучкості та широким можливостям.



2.1.1 Популярність фреймворків

Як зазначено на рисунку 2.1.1, фреймворк Django є одним з найбільш популярних фреймворків станом на січень 2022 року. Більшу популярність має тільки Laravel.

Фронтенд даного проекту, розробленого з використанням фреймворка Django, відповідає за відображення і взаємодію з користувачем через веб-інтерфейс. Він складається з HTML-шаблонів, CSS-стилів та JavaScript-коду. Основна мета фронтенду - забезпечити зручне, привабливе та інтуїтивно зрозуміле спілкування користувача з системою обліку податків.

Основні компоненти фронтенду нашого проекту:

1. HTML-шаблони: Використовуються для створення розмітки сторінок веб-додатку. Ми можемо створювати шаблони для кожного екрану або

компонента додатку, які визначають структуру сторінки та включають необхідні дані для відображення.
2. CSS-стилі: Використовуються для задання зовнішнього вигляду і стилізації елементів веб-додатку. За допомогою CSS можна встановлювати кольори, розміри, шрифти, вирівнювання тощо, щоб створити привабливий і консистентний дизайн.
3. JavaScript: Використовується для реалізації інтерактивності на стороні клієнта. Ми можемо використовувати JavaScript для реагування на події користувача, зміни стану сторінки без перезавантаження, виконання асинхронних запитів до сервера та багато іншого.
4. Взаємодія з Django: Фронтенд взаємодіє з бекендом (розробленим на Django) через API. Він може здійснювати HTTP-запити до сервера для отримання або надсилання даних, оновлення сторінок за допомогою AJAX і виконання інших операцій з бекендом.
5. Валідація даних: Фронтенд виконує перевірку та валідацію даних, введених користувачем, перед їх надсиланням на сервер. Це допомагає запобігти некоректним або неправильним даним, забезпечує точність та цілісність даних, які обробляються системою обліку податків.
6. Дизайн і взаємодія з користувачем: Фронтенд створює зручний та привабливий інтерфейс для користувача. Він включає розміщення елементів, навігацію, взаємодію з формами, кнопками, списками, таблицями та іншими елементами, що спрощують використання системи обліку податків.

Під час розробки фронтенду ми враховували принципи доброї практики проектування інтерфейсу, забезпечуючи зручність використання, ефективність та коректну відповідь на дії користувача. Окрім того, ми дотримувалися дизайну, щоб забезпечити єдність зовнішнього вигляду та стиль облікової системи.

В результаті отримали функціональний та привабливий фронтенд, який дозволяє користувачам легко взаємодіяти з системою обліку податків, вносити необхідні дані, переглядати і аналізувати інформацію та здійснювати інші дії, необхідні для правильного та зручного управління податками.

В цьому проекті, база даних використовується для зберігання і керування інформацією, необхідною для системи обліку податків. Ми використовуємо базу даних SQLite, яка є вбудованою в фреймворк Django, і забезпечує нам надійність та ефективність зберігання даних.

Робота з базами даних в проекті включає такі ключові аспекти:

Перший, моделі бази даних: Ми використовуємо мову програмування Python та ORM (Object-Relational Mapping) Django для визначення моделей бази даних. Моделі відображають структуру та взаємозв'язки даних, які зберігаються в базі даних. Кожен клас моделі відповідає таблиці в базі даних, а атрибути цього класу відображають поля таблиці.

Другий, міграції бази даних: Для створення та оновлення структури бази даних ми використовуємо механізм міграцій Django. Міграції дозволяють автоматично створювати таблиці, поля, індекси та інші об'єкти бази даних на основі визначених моделей. Це забезпечує зручну та безпечну роботу з базою даних, оновлення схеми даних та міграції даних.

Третій, запити до бази даних: Django надає потужний ORM, який дозволяє нам виконувати запити до бази даних за допомогою зручного API. Ми можемо виконувати запити на вибірку, оновлення, вставку та видалення даних з бази даних. ORM автоматично перетворює наші запити на мову SQL і взаємодіє з базою даних.

Четвертий, робота з даними: Ми можемо отримувати, модифікувати та видаляти дані з бази даних за допомогою ORM. Можемо використовувати фільтри, сортування та інші методи ORM для отримання необхідних даних. Ми також можемо виконувати складні запити, об'єднувати дані з різних таблиць та використовувати агрегаційні функції.

П’ятий, захист даних: Для захисту даних в базі даних ми використовуємо вбудовані механізми Django, такі як міграція паролів, захист від SQL-ін'єкцій та інших атак на безпеку даних. Ми також можемо використовувати різні рівні доступу та автентифікацію для забезпечення конфіденційності та безпеки даних.

Шостий, оптимізація та масштабованість: Django надає можливості для оптимізації роботи з базою даних, такі як індекси, кешування та інші методи для покращення продуктивності системи. Ми також можемо масштабувати базу даних, використовуючи розподілені системи баз даних або реплікацію даних для забезпечення високої доступності та швидкодії.

Загалом, робота з базами даних у нашому проекті відбувається через мощний ORM Django, який надає зручність, безпеку та ефективність у взаємодії з базою даних. Ми можемо зберігати, отримувати та модифікувати дані з бази даних, виконувати складні запити та забезпечувати безпеку даних в системі обліку податків.

Для керування базою даних у нашому проекті використовувалася система управління базами даних (СУБД) SQLite, яка є вбудованою в фреймворк Django. SQLite є легковаго, портативним та швидкодіючим СУБД, яка дозволяє нам зберігати та керувати даними в нашому проекті.

Одним з головних переваг використання SQLite є його простота встановлення та налаштування. Він не вимагає окремого сервера баз даних, оскільки база даних зберігається у вигляді файлу на файловій системі. Це полегшує розгортання проекту та його перенесення між середовищами.

Крім того, SQLite підтримує багато особливостей, які дозволяють нам ефективно працювати з даними. Ми можемо використовувати різні типи даних (такі як цілі числа, рядки, дата та час), індексувати дані для покращення швидкодії запитів, використовувати транзакції для забезпечення цілісності даних та багато іншого.

Однак, варто відзначити, що SQLite має свої обмеження. Він більше підходить для невеликих або середніх проектів з обмеженою кількістю одночасних з'єднань та обсягом даних. Для великих проектів з великою кількістю користувачів та вимог до масштабованості, можуть бути вибрані інші СУБД, такі як MySQL, PostgreSQL або Oracle.

У нашому випадку, SQLite відповідає потребам проекту зі зберіганням та керуванням даними системи обліку податків. Використання SQLite спрощує налаштування та розгортання проекту, забезпечує швидкий доступ до даних та надійність зберігання.

2.2 Архітектура проекту:

Архітектура проекту Django використовує модель-представлення-контролер (MVC) або модель-представлення-шаблон (MVT) шаблон для розділення логіки додатку. Ця архітектура дозволяє легко керувати та розширювати проект. Моделі відображають структуру та поведінку даних, представлення обробляють логіку додатку, а шаблони відображають дані користувачам. URL-шаблони визначають, які представлення мають обробляти запити. Завдяки цій архітектурі проект на Django стає зрозумілим та зручним для розробки та підтримки. Архітектура проекту Django використовує модель-представлення-контролер (MVC) або модель-представлення-шаблон (MVT) шаблон для розділення логіки додатку. Ця архітектура дозволяє легко керувати та розширювати проект. Моделі відображають структуру та поведінку даних, представлення обробляють

логіку додатку, а шаблони відображають дані користувачам. URL-шаблони визначають, які представлення мають обробляти запити. Завдяки цій архітектурі проект на Django стає зрозумілим та зручним для розробки та підтримки.

Які ж основні компоненти архітектури проекту в Django?

Перший компонент, моделі (Models): Моделі відображають структуру та поведінку даних в базі даних. Вони представляють таблиці бази даних та включають поля, методи та зв'язки між моделями. Моделі Django використовують ORM (Object-Relational Mapping) для спрощення взаємодії з базою даних.

Другий, представлення (Views): Представлення обробляють логіку додатку та взаємодіють з моделями та шаблонами. Вони приймають HTTP-запити від користувача, обробляють дані та відповідають на запити, використовуючи дані з моделей.

Третій, шаблони (Templates): Шаблони використовуються для відображення даних користувачам. Вони містять HTML-код разом з динамічними елементами, які заповнюються даними з представлень. Шаблони дозволяють структурувати вигляд сторінок та переиспользовати код.

Четвертий, URL-шаблони (URL Patterns): URL-шаблони визначають шаблони URL-адрес, за якими Django мапує HTTP-запити до відповідних представлень. Вони вказують, яке представлення має бути викликане для кожного конкретного URL.

П’ятий, утиліти та допоміжні класи: Django надає багато утиліт та допоміжних класів, що спрощують роботу зі зберіганням даних, формуванням запитів до бази даних, аутентифікацією користувачів, обробкою форм та багато іншого.

Архітектура Django забезпечує розділення логіки додатку на компоненти, що сприяє простоті управління та розширенню проекту. Вона дозволяє розміщувати бізнес-логіку в моделях, виконувати обробку запитів у представленнях та відображати дані за допомогою шаблонів. Це сприяє розумінню та підтримці коду, а також полегшує роботу у команді розробників.

Архітектура модель-представлення-контролер (MVC) має декілька переваг:

Розділення логіки: MVC дозволяє чітко розділити бізнес-логіку додатку від його представлення та управління. Це полегшує розробку, тестування та підтримку коду.

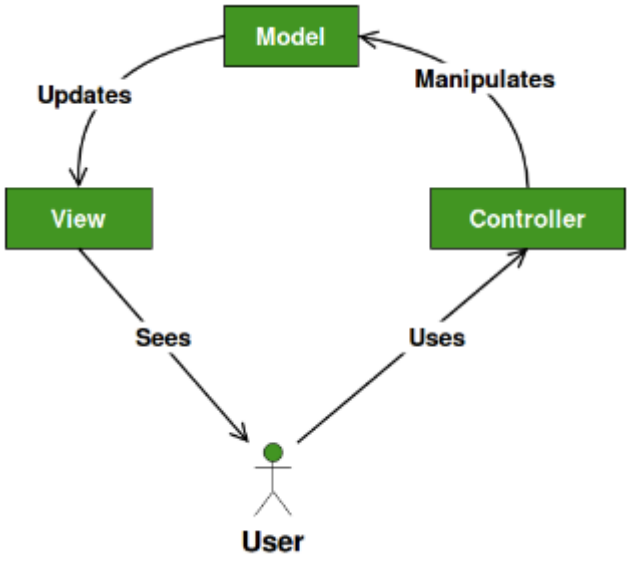
Модульність: Кожна складова архітектури (модель, представлення, контролер) є незалежною і може бути розроблена та модифікована окремо. Це дозволяє зосередитись на конкретних аспектах додатку та забезпечує більшу гнучкість і перевикористання коду.

Покращена управління: MVC спрощує управління додатком, оскільки логіка розподілена між різними компонентами. Це полегшує зрозуміння та відлагодження коду, а також сприяє прискоренню розробки.

Розширюваність: Архітектура MVC дозволяє легко додавати нові функції та змінювати існуючі без необхідності вносити значні зміни в інші частини додатку. Це сприяє гнучкості та швидкості розвитку проекту.

Відокремлення інтерфейсу: Представлення в MVC відповідає за відображення даних користувачам та взаємодію з ними. Це дозволяє легко змінювати та адаптувати інтерфейс без впливу на логіку додатку.

Узагальнюючи, архітектура MVC забезпечує модульність, гнучкість, легкість управління та розширюваність проекту. Вона є популярною в багатьох фреймворках розробки програмного забезпечення, включаючи Django, через свої переваги у розробці веб-додатків.



2.2.1 UML діаграма шаблону проектування MVC

2.3 Структура бази даних:

Реалізована база даних була з використанням SQLite. Використання інструментів Django спростило роботу з базою. Таблиці створюються за допомогою файлів міграції

Ось приклад файлу міграції

```
from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='TaxRecord',
            fields=[
                ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('rno_kpp', models.CharField(max_length=10)),
                ('arrears_amount', models.DecimalField(decimal_places=2,
max_digits=10)),
                ('surname', models.CharField(default=100, max_length=100)),
                ('name', models.CharField(default=100, max_length=100)),
                ('middle_name', models.CharField(default=100,
max_length=100)),
            ],
        ),
    ]
```

Назви полів, типи полів, та зв’язки між таблицями записуються саме в цей файл.

Для заповнення таблиць існує два методи

Перший метод:

Відкриваємо консоль і вписуємо наступні програми:

```
from tax_app.models import TaxRecord

tax_record_1 = TaxRecord(rno_kpp='1234567890',
arrears_amount=1000.00)

    tax_record_1.save()
```

Другий метод:

Відкриваємо таблицю і створюмо рюдок. Далі вводимо дані

Нижче наведено приблизний вигляд самої таблиці бази даних

	id	rno_kpp	passport	arrears_amount	surname	name	middle_name
1	1	1234567890	657342174	1000	Сидорчук	Андрій	Мстиславович
2	2	9876543210	423565879	500.5	Антонов	Владислав	Ігорович
3	3	3244544535	654872032	3240	Сергіюк	Сидір	Андрійович

Рис. 2.3.1. Таблиця бази даних

Розділ ІІІ

РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ СИСТЕМИ ОБЛІКУ ПОДАТКІВ

Починаємо з налаштування файлу конфігурації. У даному проекті файлом конфігурації є файл **“settings.py”**. Цей файл знаходиться в кореневій папці проекту і містить налаштування та конфігурацію проекту. Він є одним з основних файлів, який використовується Django для управління проектом.

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-
_%!_47lxg%d8*e5r$)msd=pt$2j2@qo012xl8&a0fzh)wxl$-i'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tax_app',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'tax_accounting.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates']
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
```

```
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'tax_accounting.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
```

```
# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

У файлі **“settings.py”** ми можемо налаштовувати різноманітні аспекти проекту, такі як:

Налаштування бази даних: вказуємо тип бази даних, параметри підключення, назву таблиць тощо.

Налаштування шаблонів (templates): визначаємо шляхи до шаблонів, включаємо необхідні розширення шаблонів, встановлюємо параметри шаблонізатора тощо.

Налаштування статичних файлів: вказуємо шляхи до статичних файлів (CSS, JavaScript, зображення тощо).

Налаштування аутентифікації і авторизації: встановлюємо параметри для системи аутентифікації, такі як рівні доступу, ролі користувачів, правила авторизації тощо.

Налаштування електронної пошти: вказуємо параметри для відправки і отримання електронних листів.

Налаштування мови та локалізації: встановлюємо мову, формат дати, формат валюти та інші параметри локалізації.

Налаштування сторонніх бібліотек і плагінів: включаємо та конфігуруємо додаткові бібліотеки, які використовуються в проекті.

Файл **“settings.py”** грає важливу роль у конфігурації проекту і дозволяє нам налаштовувати його поведінку та функціональність згідно з нашими потребами.

Потім ми визначаємо моделі Django, які представляють сутності в нашій системі обліку податків. Це можуть бути моделі для користувачів, податкових декларацій, платежів, клієнтів тощо. Ми використовуємо класи моделей Django для опису структури таблиць бази даних, включаючи поля, зв'язки та методи. Самі моделі визначаються в файлі **“models.py”**.

Ось приклад моделі:

```
class TaxRecord(models.Model):
    rno_kpp = models.CharField(max_length=100)
    arrears_amount = models.DecimalField(max_digits=10, decimal_places=2)
    surname = models.CharField(max_length=100)
```

```
    name = models.CharField(max_length=100)
    middle_name = models.CharField(max_length=100)
    passport = models.CharField(max_length=100)
```

Після визначення моделей ми створюємо міграції, які описують зміни в базі даних на основі визначених моделей.

Для створення файлів міграцій використовується команда:

python manage.py makemigrations tax_app

Для застосування цих міграцій до бази даних використовується команда:

python manage.py migrate

Тепер можемо заповнювати базу даних для подальшого користування:

	id	rno_kpp	passport	arrears_amount	surname	name	middle_name
1	1	1234567890	657342174	1000	Сидорчук	Андрій	Мстиславович
2	2	9876543210	423565879	500.5	Антонов	Владислав	Ігорович
3	3	3244544535	654872032	3240	Сергіюк	Сидір	Андрійович
4	4	1893642081	134729480	5400	Богданов	Андрій	Сергійович
5	5	9387649781	028754094	3300	Ігнатенко	Ігор	Васильович
6	6	9834754892	169843098	1800	Мельник	Сергій	Ярославович
7	7	1002384939	123423008	2300	Сидорчук	Артем	Андрійович
8	8	7234592798	564298974	1000	Пальчевський	Дмитро	Антонович
9	9	4568293846	213646879	1000	Романюк	Роман	Богданович
10	10	4878576687	653567656	500	Баранюк	Ігор	Андрійович

Рис. 3.1. База даних

Далі ми створюємо html шаблон системи обліку податків.

Вигляд html шаблону веб-додатку системи обліку податків:

```
<h1>Заборгованість по землі</h1>
<form method="post">
    {% csrf_token %}
    <label for="rno_kpp">РНОКПП:</label>
    <input type="text" name="rno_kpp">
    <button type="submit">Перевірити</button>
</form>
{% if rno_kpp %}
    {% if error_message %}
        <p>{{ error_message }}</p>
    {% else %}
        <p>Прізвище: {{ tax_record.surname }}</p>
        <p>Ім'я: {{ tax_record.name }}</p>
        <p>По батькові: {{ tax_record.middle_name }}</p>
        <p>Номер паспорта: {{ tax_record.passport }}</p>
        <p>РНОКПП: {{ rno_kpp }}</p>
        <p>Заборгованість: {{ tax_record.arrears_amount }}</p>
```

```
    {% endif %}
{% endif %}
```

<form>: Форма, яка використовується для введення даних користувачем і їх відправки на сервер. Метод **method="post"** вказує, що дані будуть відправлені методом POST.

{% csrf_token %}: Тег шаблону Django, який додає CSRF-токен для захисту від атаки типу "Cross-Site Request Forgery".

<label>: Мітка для введення користувача, вказується текст "РНОКПП".

<input>: Поле введення тексту, яке дозволяє користувачеві ввести значення. Вказано атрибут **name="rno_kpp"**, який використовується для ідентифікації поля на серверній стороні.

<button>: Кнопка, яка виконує відправку форми.

м**{% if rno_kpp %}**: Умовний оператор шаблону Django, який перевіряє, чи існує значення **rno_kpp** (введене користувачем РНОКПП).

{% if error_message %}: Умовний оператор шаблону Django, який перевіряє, чи існує значення **error_message** (повідомлення про помилку).

{{ tax_record.surname }}, **{{ tax_record.name }}**, тощо: Шаблонний тег Django, який відображає значення полів об'єкта **tax_record** (прізвище, ім'я, по батькові, номер паспорта, заборгованість).

Загальною метою цього HTML-коду є відображення сторінки, яка містить форму для введення РНОКПП, а також виведення інформації про заборгованість, якщо вона доступна.

Далі ми визначаємо функції перегляду Django, які обробляють HTTP-запити від клієнтів. У цих функціях ми взаємодіємо з моделями та іншими компонентами системи, щоб виконувати необхідні дії, такі як отримання, створення, редагування або видалення даних.

Відповідні функції перегляду визначаються в файлі **“views.py”**:

```
def tax_arrears(request):
    if request.method == 'POST':
        rno_kpp = request.POST.get('rno_kpp', '')  # Отримати РНОКПП з форми

        try:
            tax_record = TaxRecord.objects.get(rno_kpp=rno_kpp)  # Отримати
запис за РНОКПП
            context = {
                'rno_kpp': rno_kpp,
                'tax_record': tax_record
            }
```

```
            return render(request, 'tax_app/tax_arrears.html', context)
        except TaxRecord.DoesNotExist:
            error_message = "Для РНОКПП {0} дані по заборгованості не
знайдені.".format(rno_kpp)
            context = {
                'rno_kpp': rno_kpp,
                'error_message': error_message
            }
            return render(request, 'tax_app/tax_arrears.html', context)

    return render(request, 'tax_app/tax_arrears.html')
```

Даний код є функцією **tax_arrears**, яка відповідає за обробку запиту на сторінці "Заборгованість по землі". Давайте розглянемо деталі цього коду:

def tax_arrears(request): Визначає функцію **tax_arrears**, яка приймає об'єкт запиту **request**.

м**if request.method==’POST:’**: Перевіряємо, чи метод запиту є POST. Це означає, що форма була відправлена.

rno_kpp = request.POST.get(‘rno_kpp’, ’’): Отримуємо значення поля з ім'ям **rno_kpp** з POST-даних форми. Якщо поле не існує, встановлюємо значення за замовчуванням як пустий рядок.

try:: Розпочинаємо блок **try**, де виконується пошук запису в базі даних за значенням РНОКПП.

tax_record = TaxRecord.objects.get(rno_kpp=rno_kpp): Отримуємо запис з моделі **TaxRecord**, де значення поля **rno_kpp** співпадає з **rno_kpp**, отриманим з форми.

context = {…}: Створюємо словник context, який містить дані для передачі до шаблону. Включаємо значення **rno_kpp** та знайдений **tax_record**.

return render(request, ‘tax_app/tax_arrears.html’, context): Повертаємо відповідь на запит з відображенням шаблону **‘tax_app/tax_arrears.html’** і передачею **context**.

exept TaxRecord.DoesNotExist:: Якщо запис не знайдено, виконуємо блок

exept. Створюємо повідомлення про помилку.

context = {…}: Створюємо словник context, який містить дані для передачі до шаблону. Включаємо значення **rno_kpp** та **error_message**.

return render(request, ‘tax_app/tax_arrears.html’, context): Повертаємо відповідь на запит з відображенням шаблону **‘tax_app/tax_arrears.html’** і передачею **context**.

return render(request, ‘tax_app/tax_arrears.html’, context): Якщо метод запиту не є POST, просто повертаємо відповідь на запит з відображенням шаблону **‘tax_app/tax_arrears.html’** без передачі будь-яких додаткових даних.

Цей код виконує обробку форми з полем **rno_kpp** і виконує пошук в базі даних за цим значенням. Якщо запис знайдено, він передається в шаблон для відображення відповідних даних. У разі, якщо запис не знайдено, виводиться повідомлення про помилку.

Ось такий вигляд воно буде мати на веб-сторінці

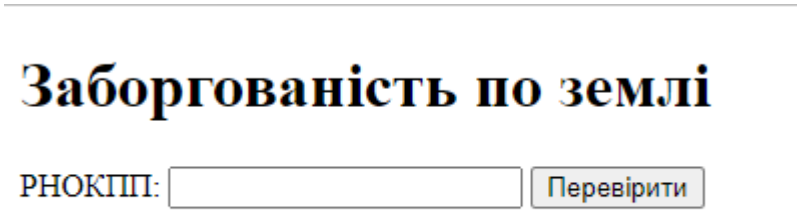


Рис. 3.2. Результат виконаного коду

Розділ IV

ТЕСТУВАННЯ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ СИСТЕМИ ОБЛІКУ ПОДАТКІВ

Для початку за допомогою команди “python manage.py runserver” запускаємо сервер. Далі відкриваємо веб-додаток. Реєструватись для того, щоб дізнатись свою заборгованість по землі не треба.

На веб-додатку нам видає форму, де можна дізнатись заборгованість по землі рис. 4.1

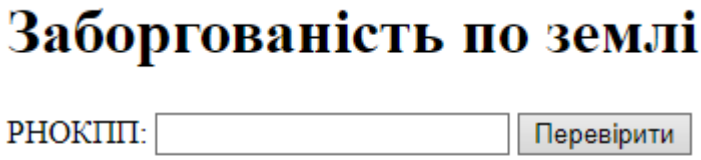


Рис. 4.1. Форма системи обліку податків

Далі відкриваємо джерела, щоб перевірити, чи не стався витік інформації з бази даних

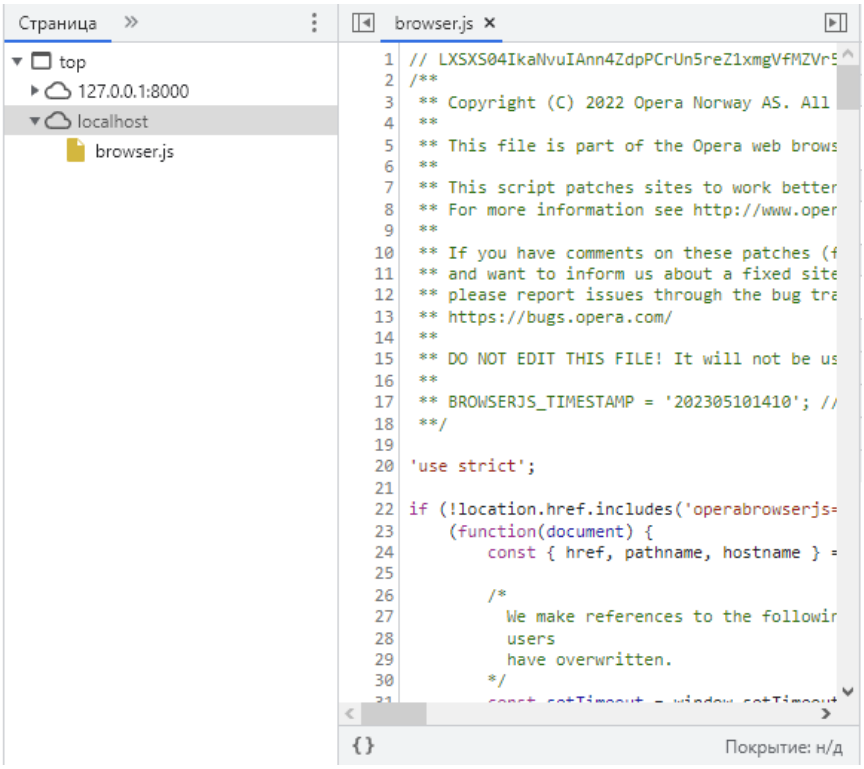


Рис. 4.2. Перевірка доступних до перегляду джерел веб-додатка

Як бачимо витоку з бази даних немає.

Далі перевіряємо коректність відображення коду веб-додатка відповідно до тогоБ що виведено на веб-сторінці

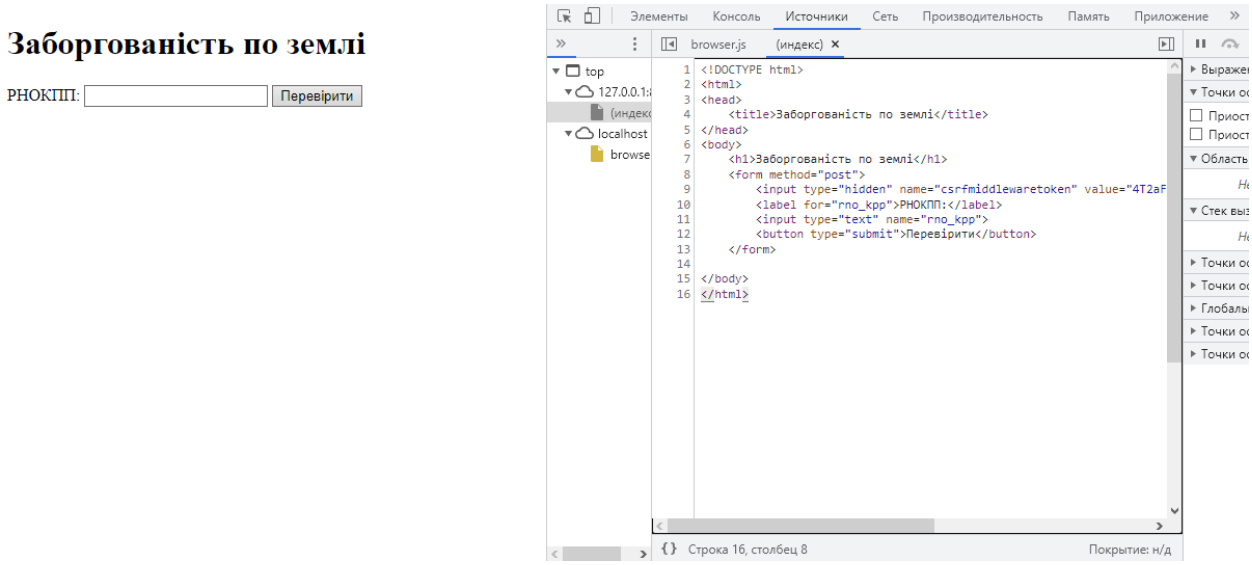


Рис. 4.3. Перевірка коду

Як бачимо код повністю відповідає виведеному на веб-сторінці.

Далі спробуємо дізнатись заборгованість по землі ввівши РНОКПП:

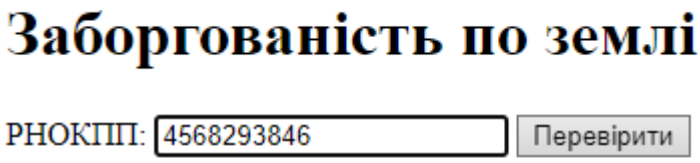


Рис. 4.4 Вводимо РНОКПП

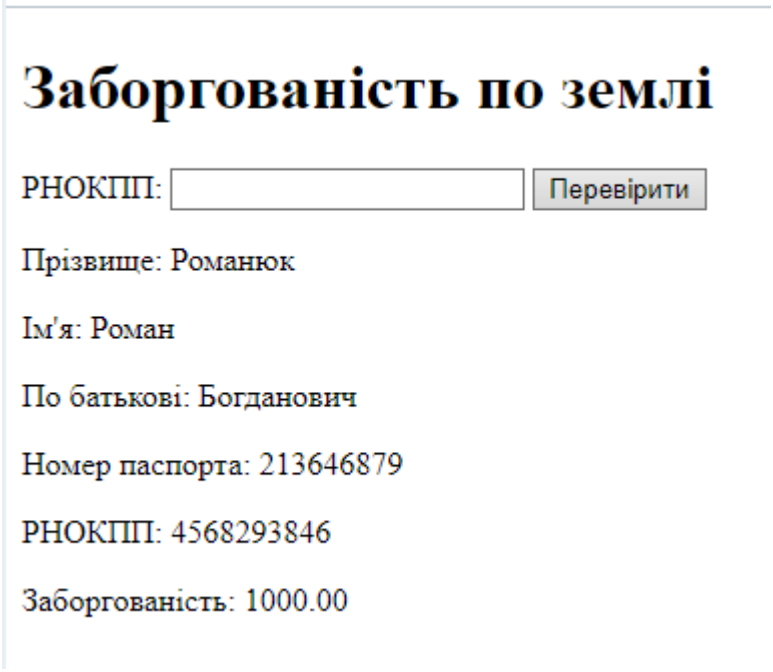


Рис. 4.5. Результат

На рисунку 4.5 зображено виведення персональних даних користувача при введенні РНОКПП, який співпав з тим, що є в базі даних.

Html код веб-додатку змінився відповідно

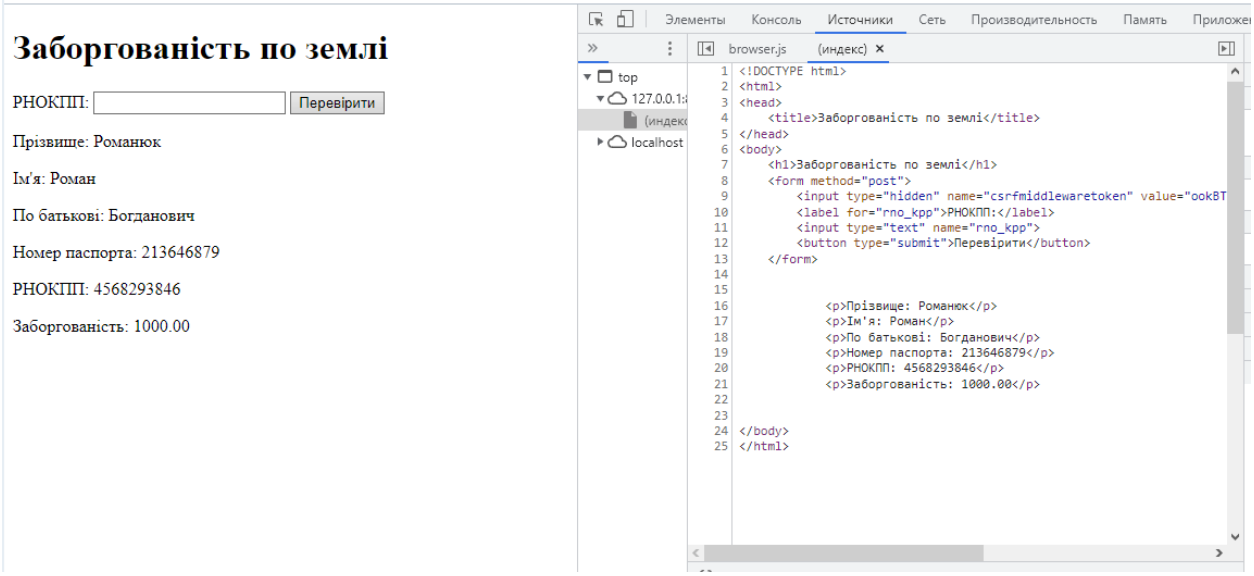


Рис. 4.6. Код сторінки після виконання дії

Давайте тепер введемо код РНОКПП, якого нема в нашій базі даних

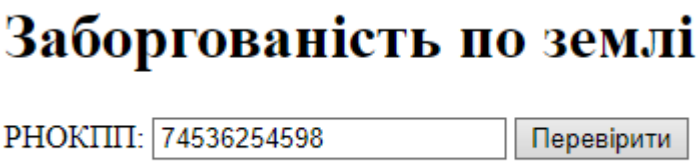


Рис. 4.7. Введення невірного коду РНОКПП

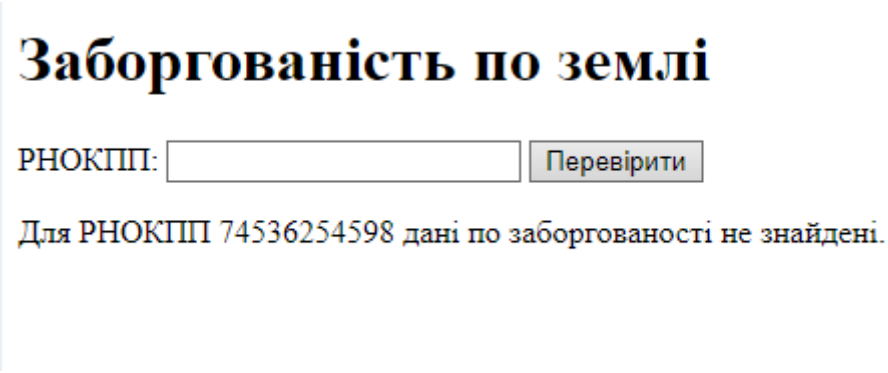


Рис. 4.8. Результат

В результаті було виведено повідомлення про те, що такого користувача немає в базі даних

Одразу відмітимо один з мінусів, що при натисканні клавіші F5(для оновлення веб-сторінки) веб сторінка не повертається в початковий стан.

ВИСНОВКИ

У результаті проведення дипломної роботи була розроблена система обліку податків на основі веб-додатку з використанням Python та фреймворку Django. Метою роботи було створення зручного та ефективного інструменту, який допоможе платникам податків відстежувати свою заборгованість, а державним органам - здійснювати контроль та моніторинг оплати податків.

В ході дослідження було проаналізовано основні проблеми, пов'язані зі збором та обліком податків. Було виявлено недоліки традиційних методів обліку та висвітлені потреби платників податків та державних органів у відповідних інструментах. Результати аналізу підкреслили необхідність впровадження нової системи обліку податків, яка б мала на меті полегшити процес оплати податків, забезпечити точний облік та контроль оплати, а також сприяти фінансовій стабільності країни.

Була визначена структура бази даних, встановлення моделей даних та їх взаємозв'язки. Також була розроблена функціональність для взаємодії з базою даних, обробки запитів та відображення результатів на веб-сторінках. Забезпечення безпеки та конфіденційності даних у системі обліку податків було також ретельно розглянуто.

У третьому розділі було проведено реалізацію системи обліку податків з використанням фреймворку Django. Були описані кроки розробки веб-додатку, включаючи створення моделей даних, налаштування маршрутів, розробку веб-сторінок та інтеграцію з базою даних. Також було надано інформацію про використані бібліотеки та інструменти для розробки системи.

У четвертому розділі було проведено тестування розробленої системи обліку податків, де оцінювалися функціональність, надійність та продуктивність системи. Результати тестування були проаналізовані, а виявлені проблеми та недоліки були виправлені.

Отже, в результаті дипломної роботи була розроблена ефективна та зручна система обліку податків, що відповідає потребам платників податків та державних органів. Система дозволяє платникам податків відстежувати свою заборгованість, а державним органам - здійснювати контроль та моніторинг оплати податків. Розроблена система сприятиме полегшенню процесу оплати податків, забезпечить точний облік та контроль оплати, а також сприятиме фінансовій стабільності країни.

Далі рекомендується продовжити розвиток та вдосконалення системи обліку податків, зокрема, можна розширити функціонал додатку, додати додаткові можливості аналітики та звітності, а також поліпшити інтерфейс для більш зручного користування. Також, необхідно забезпечити постійне оновлення системи, враховуючи зміни у законодавстві та податкових процедурах.

Загальною висновком є те, що розроблена система обліку податків відповідає вимогам та потребам сучасного суспільства і може бути успішно використана для збирання та обліку податків. Цей проект є важливим кроком у поліпшенні економічного розвитку та фінансової стабільності країни, сприяючи ефективному взаємодії між платниками податків та державними органами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Eric Matthes (2015). "Python Crash Course"
2. William S. Vincent (2020), "Django for Beginners"
3. Antonio Melé (2020), "Django 3 By Example"
4. Daniel Roy Greenfeld, Audrey Roy Greenfeld (2020), "Two Scoops of Django 3.x"
5. Arun Ravindran (2015), "Django Design Patterns and Best Practices"
6. Nigel George (2020), "Mastering Django: Core"
7. Harry Percival (2017), "Test-Driven Development with Python"
8. Gaston C. Hillar (2018), "Django RESTful Web Services"
9. Agiliq (2015), "Django ORM Cookbook"
10. Офіційна документація Django (https://docs.djangoproject.com/)
11. Stack Overflow (https://stackoverflow.com)
12. Robert G. Dromey (2017), "Tax Accounting: Principles and Practice"
13. John Weiss (2016) "Fundamentals of Tax Accounting"
14. Michael Grantham, Jennifer Grantham (2015), "Tax Accounting: Theory and Practice"
15. Michael Chris, David Dowling (2013), "Tax Accounting: Aspects of Theory and Practice"
16. James Hasselberg (2011), "Tax Accounting: Implementation and Strategies"
17. Donald J. Krese (2010), "Tax Accounting: Concepts and Practice"