

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»  
Кафедра мультимедійних систем факультету інформатики



**Проектування та розробка користувацького інтерфейсу веб-системи**

**Текстова частина до курсової роботи  
за спеціальністю «Інженерія програмного забезпечення»- 121**

**Керівник курсової роботи**

Асистент

Корнійчук Максим Анатолійович

\_\_\_\_\_

(підпис)

“ \_\_\_\_ ” \_\_\_\_\_ 2021 р.

**Виконав студент ІІЗ-4:**

Кривошеєнко Ю.Д.

“ \_\_\_\_ ” \_\_\_\_\_ 2021р.

Київ 2021

Міністерство освіти і науки України  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА  
АКАДЕМІЯ»  
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних систем,  
к. ф.-м. н. О. П. Жежерун

\_\_\_\_\_ (підпис)  
“ \_\_\_\_\_ ” \_\_\_\_\_ 2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

Студента Кривошеєнко Юлії Дмитрівни факультету інформатики 4 курсу

ТЕМА

Вихідні дані:

Зміст ТЧ до курсової роботи:

Календарний план

Вступ

Розділ 1. Проектування користувацького інтерфейсу

Розділ 2. Реалізація веб системи

Висновки

Список використаної літератури

Дата видачі “ \_\_\_\_\_ ” \_\_\_\_\_ 2021 р. Керівник \_\_\_\_\_ (підпис)

Завдання отримав \_\_\_\_\_ (підпис)

## Календарний план виконання роботи

**Тема:**

№	Назва етапу	Термін виконання	Примітка
1.	Отримання теми курсової роботи	09.10.2020	
2.	Пошук літератури за темою роботи	10.10.2020	
3.	Визначення продукту для розробки	20.10.2020	
4.	Дослідження цільової аудиторії	01.11.2020	
5.	Створення інтерактивного прототипу	27.11.2021	
6.	Реалізація веб системи	05.03.2021	
7.	Написання першої частини курсової роботи	20.03.2021	
8.	Написання другої частини курсової роботи	1.04.2021	
9.	Перегляд змісту роботи з керівником	10.04.2021	
10.	Внесення змін до роботи	10.04.2021	
11.	Створення презентації	11.04.2021	
12.	Завантаження курсової роботи	12.04.2021	

Студент Кривошеєнко Ю. Д.

Керівник Корнійчук М. А.

“        ”  
\_\_\_\_\_

## Зміст

Календарний план виконання роботи .....	3
Зміст .....	4
Анотація .....	5
Вступ .....	6
1 Проектування користувацького інтерфейсу .....	8
1.1 Поняття та важливість проектування користувацького інтерфейсу .....	8
1.2 Етапи проектування користувацького інтерфейсу .....	10
1.3 Визначення продукту .....	11
1.4 Дослідження користувачів .....	14
1.5 Створення user story map .....	16
1.6 Прототипування .....	19
1.6.1 Створення wireframes .....	21
1.6.2 Створення інтерактивного прототипу .....	24
1.7 Тестування .....	26
2 Реалізація веб системи .....	28
2.1 Структура веб системи .....	28
2.2 Обрані засоби розробки .....	29
2.3 Шаблонізатор Jinja2 .....	30
2.4 Опис розробки фронтенд частини .....	33
Висновок .....	40
Список літератури .....	42

## Анотація

Метою даної роботи є розгляд проектування користувацького інтерфейсу веб системи, а також реалізація даного проектування та розробка фронтенд частини веб системи на прикладі пошуковика по документам Єдиного державного реєстру судових рішень України.

В даній роботі описане поняття проектування користувацького інтерфейсу, задачі та проблеми, що вирішує проектування користувацького інтерфейсу, підходи до його проектування, ключові моменти та основні етапи проектування. Детально розглянуті такі етапи проектування як: визначення продукту, дослідження користувачі, створення карт користувацьких історій, прототипування та тестування.

На основі аналізу результатів дослідження цільової аудиторії було створено інтерактивний прототип користувацького інтерфейсу пошуковика документів та була розроблена фронтенд частина даної веб системи.

## Вступ

З розвитком технологій у 21 столітті всі хочуть отримати найкращий досвід від використання технологій, не витрачаючи багатого свого часу та зусиль. Тож при розробці будь-якого цифрового продукту: веб-сайту, мобільного додатку, тощо, приваблива графіка та захоплюючий зміст не є гарантими успішного результату кінцевого продукту.

Тож, першочерговою метою створення будь-якого продукту є створення рішення, яке буде приносити користь та яке буде зручним у користуванні. Оскільки користувачі мають безліч варіантів вибору та часто навіть альтернатив для продуктів та/або послуг, які їм пропонують, для успіху продукту виконується проектування користувацького інтерфейсу з використанням ці/их підходів. Адже вони допомагають зрозуміти свою аудиторію під час проведення різноманітних досліджень, збільшити задоволеність своїх користувачів, покращивши зручність взаємодії, знизити вартість залучення клієнтів та збільшити частку утримання клієнтів.

Метою дослідження є розгляд підходів до проектування користувацького інтерфейсу та їх етапів, та реалізація проектування користувацького інтерфейсу за етапами та власне розробка фронтенд частини веб системи.

Для демонстрації прикладного застосування даного дослідження командою “ThemidaDevs”, яка складається з студентів ІІІ-4: Кривошеєнко Юлії, Мероник Тетяни, Бойцова Симона, Кузьменка Дмитра, та студента Менеджмент-4 Кожевнікова Микити, була розроблена веб система – пошуковик по документам Єдиного державного реєстру судових рішень України. Для визначення продукту (даної веб системи) був проведений аналіз процесів взаємодії користувачів з наявними системами пошуку документів з Реєстру судових рішень та визначено основні задачі та проблеми, які виникають при автоматизації процесів в цій сфері. Розроблена веб система дає змогу користувачу виконувати пошуковий запит, що складається як із одного слова, так і з фраз, використовуючи пошукові фільтри.

Дана робота складається з двох розділів.

У першому розділі наводяться пояснення поняття проектування користувацького інтерфейсу та його важливості. Описується концепція «design thinking» та її ключові етапи. Проводиться опис основних етапів проектування користувацького інтерфейсу таких як: визначення продукту, дослідження користувачів, створення карт користувацьких історій, прототипування та тестування, та їх реалізація та отримані результати на прикладі проектування користувацького інтерфейсу веб системи пошуковика по документам Єдиного державного реєстру судових рішень України.

Другий розділ даної роботи включає у себе опис архітектури та структуру розробленої веб системи. Описуються обрані засоби для розробки пошуковика та аргументується їх вибір. Детально описується розробка фронтенд частини веб системи та демонструється зовнішній вигляд розробки за допомогою скрінів веб системи.

# 1 Проектування користувацького інтерфейсу

## 1.1 Поняття та важливість проектування користувацького інтерфейсу

Проектування користувацького інтерфейсу передбачає глибоке розуміння користувачів за допомогою досліджень, упорядкування інформації, візуального дизайну (і багато іншого), і все це з метою задоволення потреб користувачів. Під час проектування користувацького інтерфейсу користувачі стає центром процесу проектування та розробки та встановлення ітеративного циклу досліджень, проектування та оцінки.

Більшість користувачів шукають та обирають продукти програмного забезпечення, які мають привабливий зовнішній вигляд та максимально зручні та інтуїтивні в користуванні. Ці критерії, як правило, є результатом ефективного поєднання дизайну досвіду користувача (UX design) та дизайну інтерфейсу користувача (UI design). Чудовий User Interface дизайн створить миттєвий потяг до програми, тоді як гарний досвід роботи змусить користувача ще раз повернутися до користування продуктом.

Розробка користувацького інтерфейсу (UI) - це процес вдосконалення презентації та інтерактивності веб- або мобільних додатків, який фокусується на зовнішньому вигляді програми та її взаємодії з користувачами. Користувацький інтерфейс включає в себе текстові блоки, що читають користувачі, кнопки, текстові поля, форми, зображення та інші візуальні елементи, які бачить та з якими взаємодіє користувач під час використання програми. Також користувацький інтерфейс складається з макету екранів, переходів між сторінками, анімації та кожної мікро-взаємодії з користувачем.

Розробка взаємодії з користувачем (UX) - це процедура поліпшення загального досвіду користувачів при взаємодії з додатком або веб-сайтом для досягнення своєї основної мети - забезпечити максимальне задоволення споживачів, щоб користувачі знаходили в продукті додаткову цінність. User



Experience дизайн складається з багатьох різних підпунктів, такі як: інтерактивний дизайн, інформаційну архітектуру, візуальний дизайн, юзабіліті та взаємодію між користувачем та продуктом.[1].

UX підхід включає в себе деякий повторювальний цикл під час розробки продукту, який складається з:

- дослідження (початкове дослідження користувачів, для розуміння цільової аудиторії та їх можливості, обмеження, цілі та очікування),
- дизайну (при створенні якого використовується статистика досліджень користувачів, щоб допомогти генерувати ідеї та початковий прототип для реалізації концепцій),
- оцінки (фіксування відгуків користувачів протягом розробки проекту).[2].

Ключові моменти UX дизайну це:

- інформаційна архітектура,
- дизайн взаємодії,
- юзабіліті,
- вайфрейми,
- візуальний дизайн.

Інформаційна архітектура (Information Architecture) полягає у задоволенні бізнес-стратегій шляхом проектування інформаційної структури програми або сайту. Її основна роль: забезпечити своїм користувачам зручну навігацію. Мова йде про використання максимальних перестановок та комбінацій для забезпечення найкращого та навігаційного меню верхнього рівня.

Дизайн взаємодії - це створення концептуального дизайну, за допомогою якого користувачі взаємодіють з продуктом та / або додатком. Ця взаємодія включає різні елементи, такі як естетика, колір, шрифт, піктограми, зображення, рух, звук, простір, графіка тощо.

Юзабіліті це зручність продукту у використанні для користувача. Юзабіліті з'ясовує питання чи користувачі отримують інформацію, яку вони хочуть,

використовуючи програму або відвідуючи веб-сайт вперше, і чи зручною для навігації та інших функцій є програма / веб-сайт.

Вайфрейм (wireframe) - це зразок програми для перевірки функцій, вигляду та зручності використання програми до її фактичного запуску. Це дешевий спосіб перевірити функціональність та оцінити, чи додаток відповідає поставленим цілям. Візуальний дизайн програми або веб-сайту має визначати бренд компанії. Це не лише вибір текстових шрифтів, кольорової палітри, зображень, іконок, але також визначення впливу зовнішнього вигляду програми на взаємодію від користувачів. [3]

## 1.2 Етапи проектування користувацького інтерфейсу

У цій роботі буде розглянута концепція «design thinking». Цей процес складається з п'яти етапів: empathize (співпереживання), define (визначення проблеми), ideate (формування ідеї), prototype (створення прототипу), та test (перевірка). Більшість процесів проектування походять саме з цієї концепції. При застосуванні design thinking до дизайну продукту, можна виокремити такі п'ять основних етапів:

- Визначення продукту
- Дослідження
- Аналіз
- Дизайн
- Тестування

Визначення продукту – один з найважливіших етапів, який відбувається найпершим. Адже перед створенням продукту необхідно зрозуміти його концепцію та ціль його існування. Фаза визначення продукту задає вектор розвитку для фінального продукту. Під час цього етапу відбувається брейншторм концепції продукту.

Після визначення продукту проводиться етап дослідження, під час якого проводяться опитування та інтерв'ю, цілю яких є отримання даних для розуміння цільової аудиторії, їх потреб, бажань, цілей, мотивації та поведінки, а також розуміння ніші індустрії, до якої відноситься майбутній продукт.

Метою етапу аналізу є проаналізувати дані, що були зібрані на попередньому етапі, та ґрунтуючись на них, отримати розуміння того, чого хочуть та потребують потенційні користувачі та чому вони хочуть та потребують саме цього. На основі цього визначається чи були попередні припущення правильними. Цей етап може включати в себе створення персон та user story.

Після визначення усіх бажань, потреб та очікувань користувачів від продукту починається етап дизайну. На цьому етапі відбуваються різні види діяльності, які можуть варіюватися в залежності від проекту, починаючи від створення інформаційної архітектури та закінчуючи самим дизайном інтерфейсу користувача. Зазвичай етап дизайну включає в себе створення вайфреймів та прототипу.

Останній етап – тестування. Саме цей етап допомагає зрозуміти наскільки розроблений дизайн є зручним та зрозумілим для користувачів. Під час низки ітерацій користувацького тестування продукт перевіряється як і зацікавленими сторонами, так і кінцевими користувачами. [4].

### 1.3 Визначення продукту

Після визначення проблеми та брейншторму для генерації ідей її рішення необхідно обрати конкретну ідею та визначити майбутній продукт, а саме виділити певну кількість загальних очікувань (вимог) щодо його реалізації. Як правило, результат є досить абстрактним переліком, завдання якого - не сформулювати точний план розвитку, а навпаки, визначити вектори подальшої діяльності та роботи команди.

Для визначення майбутнього продукту використовується канва ціннісної пропозиції (Value Proposition Canvas) - інструмент, який може допомогти забезпечити позиціонування товару чи послуги навколо того, чого потребує та цінує користувач. Канва формується навколо двох основних елементів - клієнтського профілю та карти цінностей, щоб відобразити ключові аспекти продукту, а саме описати переваги, які отримує клієнт від споживання нашого продукту.

Клієнтський профіль складається з:

- клієнтських завдань,
- клієнтських болів,
- користі (вигоди) клієнта.

Блок клієнтських завдань (customer jobs) описує ті завдання, які потенційні користувачі намагаються виконати в своєму житті: завдання, які вони намагаються виконати та реалізувати, або ж їх проблеми, які вони хочуть розв'язати чи потреби, які вони бажають задовольнити.

Блок клієнтських болів (customer pains) містить в собі болі клієнтів, їх хвилювання, перешкоди, які заважають клієнтам виконувати чи почати виконання якоїсь їх роботи чи гальмує досягнення певних цілей. Також в цьому сегменті описуються ризики – потенційні погані результати, які були отримані через неякісно або не ефективно виконаною роботою чи взагалі відсутністю її виконання.

Останній сегмент користувацького профілю – вигоди клієнта (customer gains). Цей блок канви описує результати та користі, які очікують та прагнуть користувачі. Серед вигід клієнта можна виділити такі 4 типи:

- потрібна користь (найголовніша вигода, без якої рішення не буде працювати),
- очікувані вигоди (основні вигоди, які очікуються користувачами, хоча навіть без них рішення буде працювати),
- бажана користь (вигоди, які виходять за межі очікуваних переваг від рішення),

- неочікувані вигоди (користі, що виходять за межі очікуваних бажань і очікувань клієнтів від рішення).

Карта цінностей складається з:

- продуктів і послуг,
- знеболювальних,
- створення переваг.

Блок продуктів і послуг (product/service) демонструє всі рішення та послуги, на яких ґрунтується ціннісна пропозиція. Цей перелік рішень має давати змогу конкретному клієнтському сегменту виконати свої завдання та задовольняти їх потреби та бажання.

Сектор знеболювальних (product pain killers) вказують на те, яким саме чином продукт гамує клієнтський біль, що вказаний у клієнтському профілі. Хороша ціннісна пропозиція продукту ґрунтується на болях клієнта, проте втамувати водночас всі болі неможливо, тож необхідно зосередитися на декількох основних проблемах, які продукт здатен вирішити.

Блок створення переваг (product gain creators) складається з переваг, які отримає клієнт від використання продукту. Як і з знеболювальними необхідно зосередитися тільки на декількох перевагах, які є важливими для користувачів, бо задовольнити усі бажання одним рішенням неможливо.

При виборі ніші, до якої буде відноситися майбутньо розроблений продукт, було вибрано судову. Після цього було проаналізовано правову сферу та труднощі, з якими стикаються люди, що відносяться до неї, при автоматизації процесів. Тож шляхом аналізу та зроблених висновків після спілкування з людьми, що мають відношення до правничої сфери командою була виокремлена проблема, що потребувала рішення: відсутність єдиного зручного та швидкого пошуку по документам Єдиного державного реєстру судових рішень України. Наявні доступні продукти не задовольняють усі потреби користувачів та мають недоліки. Пошук по документах судового реєстру працює повільно та з затримками, має незручний інтерфейс, деякі дефекти, документи

відображаються незручно та неструктуровано. Тож рішенням стала розробка власного пошуковика судового реєстру.

Тож, для більш точного визначення продукту, а саме пошуковика судового реєстру була розроблена канва ціннісної пропозиції.

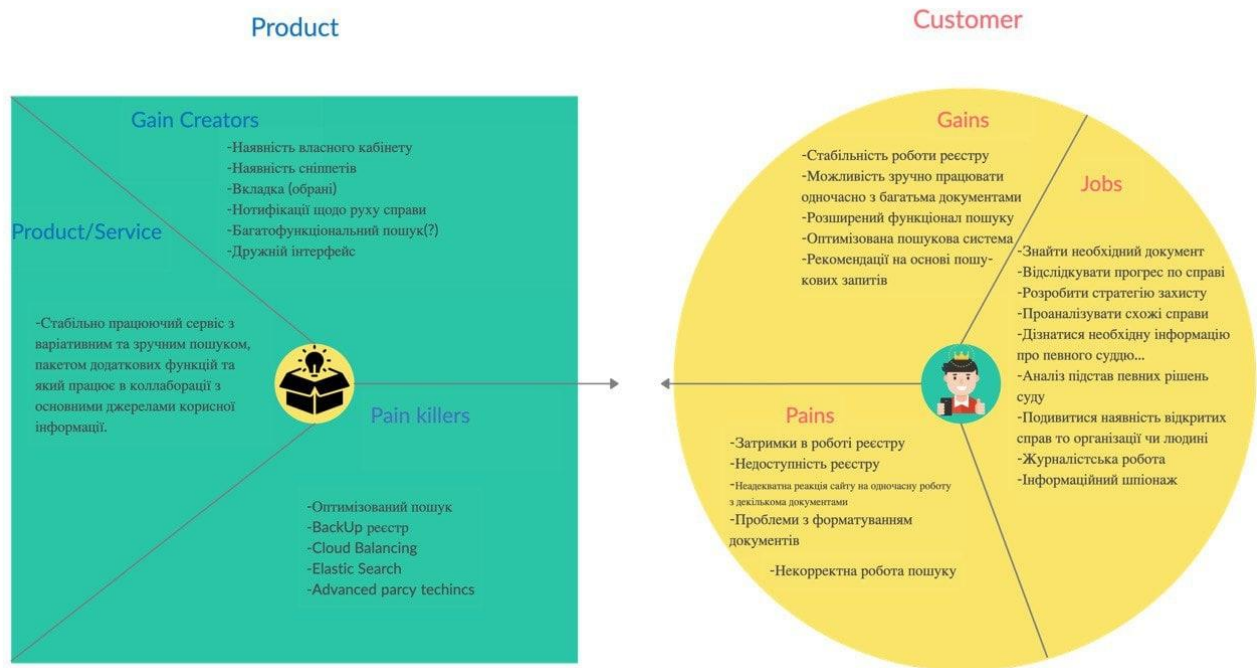


Рисунок 1 Розроблена канва ціннісної пропозиції

## 1.4 Дослідження користувачів

Дослідження користувачів є невід'ємною частиною процесу проектування користувацького інтерфейсу. Коли ми працюємо на основі припущень або просто власного досвіду, ми часто не помічаємо, яким може бути досвід користувачів для інших людей, зокрема наших користувачів. Це означає, що ми можемо легко втратити можливості вдосконалити наш сервіс або продукт, щоб задовольнити потреби клієнтів. Крім того, нам може здатися, що нашим продуктом легко користуватися, а навігація в ньому є зручною та очевидною, через те що ми маємо знання системи та вже мали досвід користування нею, а користувачі не мають переваг цього досвіду та знань. Тож дослідження

користувачів допомагає з'ясувати, як саме почуваються цільові клієнти під час взаємодії з продуктом, який розроблений для досягнення їх цілей та виконання їх задач. Дослідження користувачів повинно виконуватися на перших етапах роботи над продуктом, оскільки без нього проектування інтерфейсу може базуватися лише на власному досвіді та припущеннях, які не можна вважати об'єктивними.

Дослідження користувачів надає нам дані, необхідні для проектування продукту. Під час дослідження користувачів інформація збирається за допомогою різноманітних засобів та джерел для кращого формування остаточного інтерфейсу, наприклад, інтерв'ю з користувачем, опитування та інше.

Інтерв'ю з користувачем - це поглиблена дискусія між інтерв'юером та користувачем із цільової демографічної групи. Вона призначена для виявлення основних потреб та вимог користувача під час використання продукту.

Інтернет-опитування надсилаються вибірці із цільової аудиторії та складається з набору точних питань. Тривалість та формат онлайн-опитування можуть варіюватися, найкраще проводити опитування вже після проведення інтерв'ю, щоб сформулювати більш точно, орієнтуючись на аналіз інтерв'ю.

Створення персон є також частиною дослідження користувачів. Персона це вигадані клієнти, які представляють вибірку справжньої цільової аудиторії та її поведінки. Персона дають змогу зрозуміти користувача, його бажання, думки та завдання. Їх можна використовувати для підтвердження або спростовування гіпотез під час проектування та пріоритезувати функціонал, а також перевірити чи відповідає продукт потребам потенційних клієнтів. [5].

Під час роботи над пошуковиком судового реєстру учасником команди Микитою Кожевником було проведено декілька інтерв'ю з юристами та адвокатами. Також ним було створено інтернет опитування, яке було поширено поміж людей, пов'язаних з правничою сферою віком від 20 до 35 років. Це дало змогу зрозуміти краще бажання та потреби наших потенційних клієнтів. Також були створені персон юриста, адвоката, журналіста-розслідувача та пересічної

особи, яка має інтерес у документах судового реєстра. Задачі, які вирішує продукт для юриста: відслідковування справ та моніторинг нових по певній компанії. Адвокат зацікавлений в відслідковуванні власних справ, які він веде та знаходження подібних справ. Журналіста-розслідувач використанням продукту вирішує задачу пошуку не тільки по конкретним справам, а й по діючим особам або організаціям. Персона пересічної особи зацікавлена дізнатися номер своєї справи та слідкувати за змінами у своїй справі.

## 1.5 Створення user story map

User Story Mapping (карта користувальницьких історій) - інструмент цілісного проектування продукту на основі призначеного для користувача шляху. Ця техніка використовується для розповіді «історій користувачів» за допомогою слів та малюнків з метою не збору письмових вимог, а досягнення консенсусу, загального розуміння проблем користувачів, щоб врешті створити кращий продукт. В основі створення user story map лежать три основні складові: персона (User Persona), подорож користувача (User Journey) та користувацька історія (User Story).

Персона - це опис конкретного користувача, чиї цілі та характеристики відображають потреби більшої групи користувачів, з його завданнями, які він сподівається вирішити за допомогою вашого продукту. Створення персон відбувається на етапі дослідження користувачів.

Подорож користувача – це шлях користувача, а саме опис його задач, дій, досвіду, які отримує користувач для досягнення своєї цілі під час використання продукту. Кроки користувача фіксують у зручному форматі, а потім використовують для створення скелету карти користувацьких історій.

Користувацька історія – це коротке структуроване цілеспрямоване речення про те, чого хоче досягнути користувач від користування продуктом. Зазвичай для їх створення використовується такий шаблон "Як <роль>, я хочу



<ціль/бажання> щоб <вигода>" або більш короткий варіант: "Як <роль>, я хочу <ціль/бажання>".

Користь карти користувацьких історій складається з:

- збереження фокусу на користувачі,
- відображення шляху користувача,
- візуалізація зв'язку дій клієнта за допомогою користувацьких історій,
- можливість побачити додаткові можливості, що може дати поштовх до появи нових ідей,
- досягнення консенсусу у стейкхолдерів.

Сама карта користувацьких історій складається з трьох рівнів. Скелетом основою є подорож користувача, вона відображається на верхньому горизонтальному рівні. Наступний горизонтальний рівень – користувацькі історії. Вертикальні лінії (ребра карти) складаються з деталей-дій для кожного кроку, які пріоритезовані за важливістю зверху вниз. [6].

Для кращого розуміння деталей, які потенційний користувач пройде на кожному кроці користування пошуковиком судового реєстру було розроблено карту користувацьких історій. Для створення карт використовувалась онлайн платформа Miro.

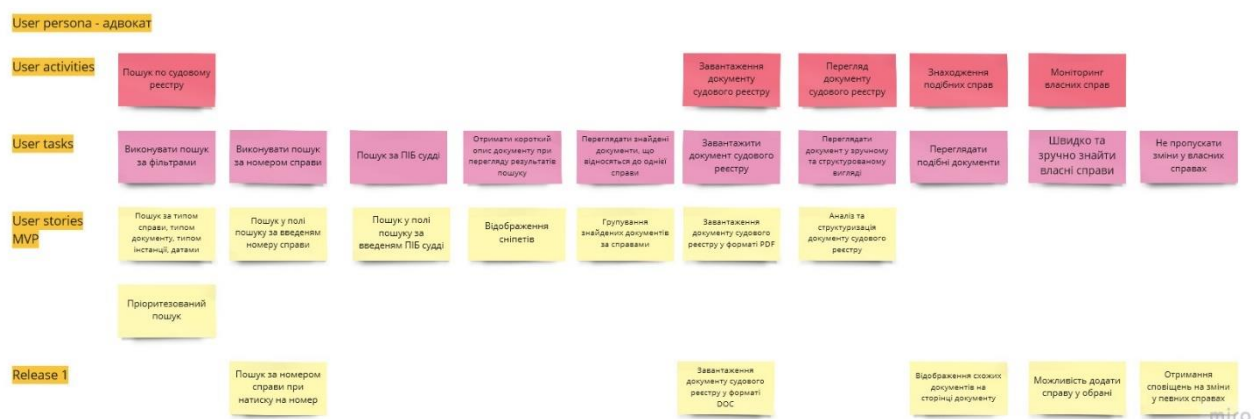


Рисунок 2 Карта користувацької історії персони адвокат

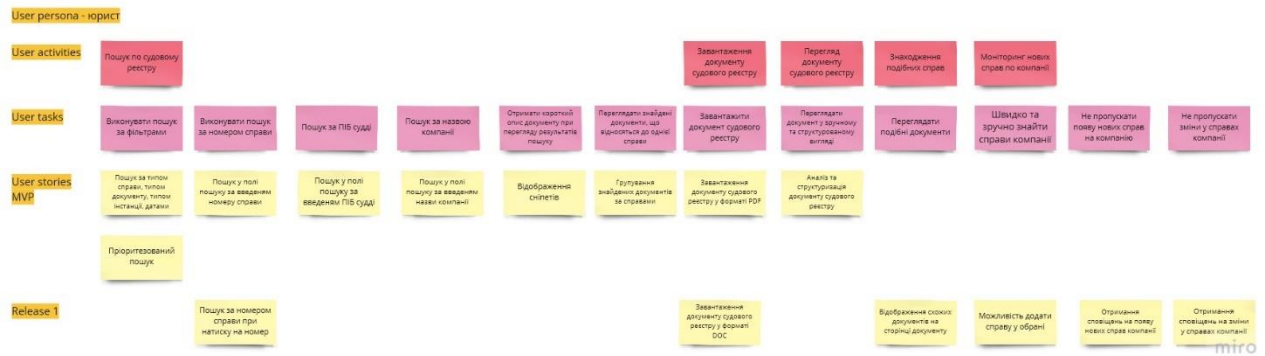


Рисунок 3 Карта користувацької історії персони юрист



Рисунок 4 Карта користувацької історії персони пересічної особи

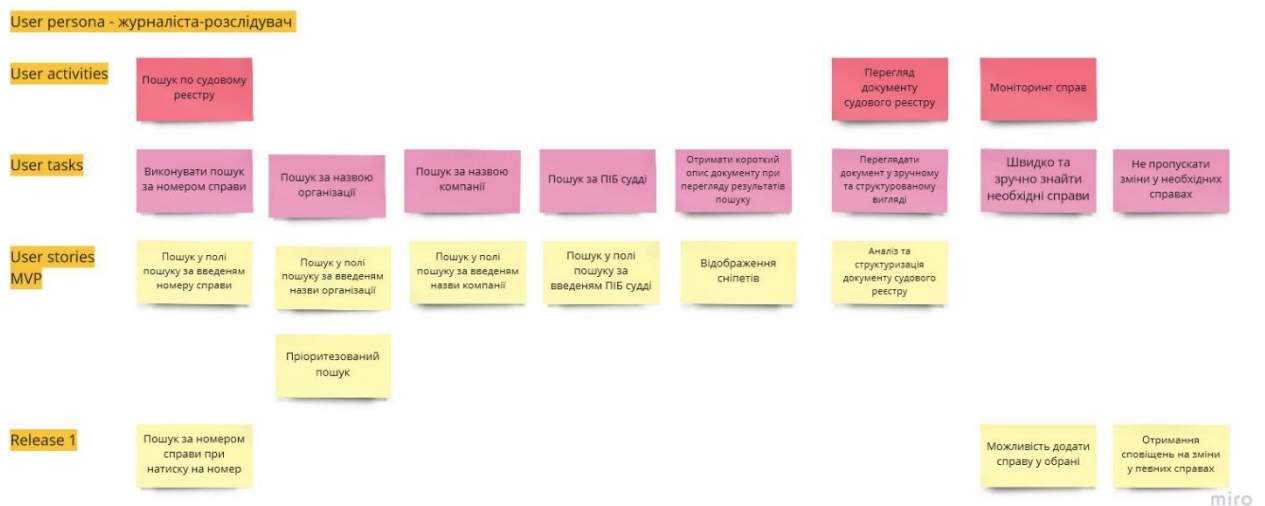
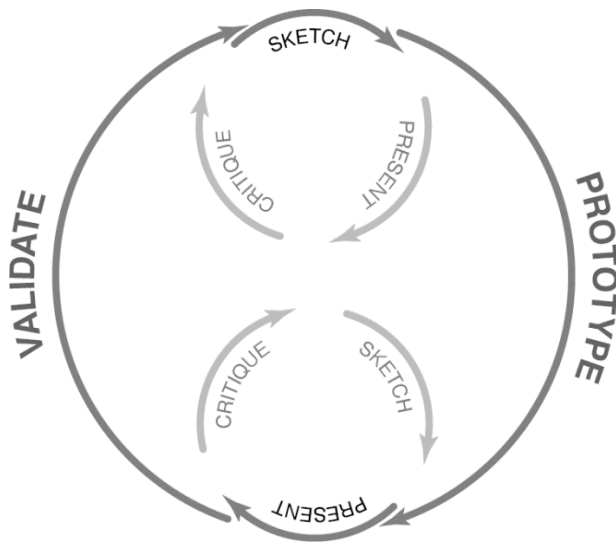


Рисунок 5 Карта користувацької історії персони журналіста-розслідувача

## 1.6 Прототипування

Наступний етап у проектуванні користувацького інтерфейсу – прототипування. Прототип - це примітивна модель або версія майбутнього продукту. Створення прототипу дає щось матеріальне для тестування на реальних та потенційних користувачах, що є надзвичайно важливим у забезпеченні придатності продукту. Прототипування може виявити проблеми дизайну продукту та місця, які потребують вдосконалення, адже коли модель продукту опиниться в руках реальних користувачів, то ви побачите, яким чином вони хочуть використовувати продукт. Таким чином дає змогу перевірити деякі гіпотези та підтвердити чи спростувати їх.

Прототипування – це ітеративний та еволюційний процес. Цикл його існування складається з створення скетчів, їх презентація та критика та згодом власного прототипування.



*Рисунок 6 Процес прототипування*

Прототипування є невід’ємною частиною процесу проектування з двох ключових причин:

- Візуалізація - прототипи дають змогу продемонструвати зацікавленим сторонам, як виглядатиме та функціонуватиме кінцевий продукт.

- Зворотній зв'язок - прототипи генерують зворотний зв'язок як від членів команди, так і від тестових груп користувачів. Потенційні клієнти можуть взаємодіяти з тестовою моделлю продукту та звернути увагу на елементи або функції, які є незручними або незрозумілими для них.

Прототипи мають 4 основні властивості:

- Репрезентація - фактична форма прототипу, паперова або цифрова.
- Точність - рівень деталізації, досконалість та реалістичність прототипу, тобто точність визначає рівень наближеності до кінцевого продукту.
- Інтерактивність - функціональність, що відкрита для користувача, наприклад, повністю функціональна, частково функціональна або доступна лише для перегляду.
- Еволюція - життєвий цикл прототипу. Деякі з них швидко створюються, тестуються, викидаються, а потім замінюються на вдосконалену версію (це називається «швидким прототипуванням»). Інші можуть бути побудовані та вдосконалені, в кінцевому підсумку перетворившись на кінцевий продукт.

Точність прототипу, як правило, залежить від етапу процесу проектування, наявних ресурсів та цілей. Наприклад, якщо метою є дати клієнту початкове уявлення про сценарій користувача, то може бути достатньо прототипу з низькою точністю. Але якщо мета полягає в тому, щоб запустити прототип фокус-групою і побачити, як потенційні клієнти взаємодіють з ним, то кращим рішенням буде більш естетично привабливий прототип високої точності.

Прототипування з низькою точністю - це швидкий та простий спосіб зробити візуальне відображення ідеї програмного продукту. Цілю створення такого прототипу є окреслити користувацький сценарій продукту та перевірити корисність та зручність його функціональності. Вони корисні на ранніх стадіях проектування інтерфейс, коли відбувається лише вибір напрямку розвитку дизайну. Таким чином, прототипи з низькою точністю не настільки візуально

продумані та естетичні, як прототипи з високою точністю, проте їх розробка більш швидка, проста та дешевша. Прикладами таких прототипів можуть бути скетчі, мокапи, вайфрейми, тощо.

Прототипи високої точності є більш досконалыми, ніж їхні аналоги з низькою точністю. Вони передбачають більш високий рівень деталізації, який нагадує реальний дизайн інтерфейсу користувача: вони мають містити реальний зміст, а також усі стилізовані належним чином функціональні елементи: кнопки, меню та форми. Функціонал прототипів високої точності максимально наближений до функціоналу кінцевого продукту. Такі прототипи зазвичай розробляються на етапі, коли є вже точне розуміння вигляду продукту, та частіше вони є кращим варіантом для тестування зручності використання, ніж прототипи низької точності. Приклади високоточних прототипів це - інтерактивні прототипи, цифрові прототипи, кодовані прототипи. [7].

### 1.6.1 Створення wireframes

Створення прототипу низької точності такого як вайфрейм є першим етапом прототипування. Вайфрейм (wireframe) – це схематична ілюстрація інтерфейсу сторінки, яка використовується для демонстрації розміщення вмісту та функціональних елементів на екрані сторінки. Вайфрейми слугують основою для прототипів, допомагаючи встановити взаємозв'язок між окремими сторінками (екранами) продукту.

Процес створення вайфреймів є необхідною частиною проектування користувацького інтерфейсу, оскільки він допомагає продемонструвати у простий спосіб інформаційну архітектуру та користувацький сценарій, описуючи структуру продукту. Вайфрейми можуть бути дуже простими та базовими, вони рідко містять кольори та стилі, адже забезпечення естетичного зовнішнього виду інтерфейсу не є ціллю їх створення. Головною метою

вайфреймів є зв'язати інформаційну архітектуру продукту з його візуальним дизайном. Вони відображають:

- розподіл місця на сторінці,
- розподіл зображень та вмісту,
- пріоритет вмісту,
- доступні функції,
- призначену поведінку.

Вайфрейми допомагають розкрити різні методи представлення або відображення різних типів вмісту та інформації, визначити пріоритети цього контенту в порядку важливості для користувача, його очікувань та цілей, а також визначити основний функціонал сторінки та підготуватися до створення прототипів більш високої точності. [8].

У ході проектування користувацького інтерфейсу пошуковика судового реєстру були розроблені вайфрейми основних екранів, щоб визначити пріоритети контенту та визначити основний функціонал головних сторінок майбутнього продукту. Спираючись на інформацію отриману у ході дослідження користувачів через проведення інтерв'ю та опитування були виокремлені та відображені рішення проблем потенційних користувачів такі як:

- групування документів по справам,
- створення сніппетів документів.

Main Frame

Themidas Project

Особистий кабінет

Пошук судових рішень

Сортувати за

Тип документу

Ухвала

-

+

Тип справи

Кримінальне

-

+

Справа

121/4363/24

Справа

123/3564/87

назва рішення

..сніпет..

Детальніше

назва рішення

..сніпет..

Детальніше

Doc page

Themidas Project

Особистий кабінет

Схожі рішення:

Вирок

Детальніше

Вирок

Детальніше

Зберегти як

Ухвала

Текст ухвали  
Текст ухвали  
Текст ухвали  
Текст ухвали

Рисунок 7 Створені вайфрейми

### 1.6.2 Створення інтерактивного прототипу

Створення інтерактивного прототипу – наступний етап прототипування. Інтерактивний прототип - це інтерактивна тестова модель наближена до представлення майбутнього продукту (сайту чи застосунку). Він складається з декількох клікабельних екранів (сторінок) з переходами між ними, кожен з яких імітує шлях користувача від початкової точки до кінцевої цілі.

Його метою є випробувати користувацький сценарій проектного рішення та зібрати відгуки про нього - як від внутрішньої, так і від зовнішньої сторони - перед побудовою кінцевого продукту. [7].

Під час тестування прототипу тестується поведінка потенційних клієнтів під час використання продукту та перевіряється чи виконує продукт їх завдання та чи є виконання їх задач зручним та ефективним. Інтерактивні прототипи дають змогу тестування без використання інженерних ресурсів. Інтерактивні засоби створення прототипів, що доступні на ринку це Axure RP, Adobe XD, Balsamiq, InVision, JustInMind та Figma.

Під час розробки пошуковика було створено інтерактивний прототип майбутнього продукту. Для виконання цього етапу було використано векторний онлайн-сервіс розробки інтерфейсів та прототипування Figma. Було обрано саме цей засіб, бо це безкоштовне кросплатформне рішення, яке має окрім десктопної та мобільної ще веб версію. Також Figma має опцію командної підписки, можливість спільного редагування та інші переваги серед конкурентів, які стали вирішальними при виборі сервісу для прототипування.

Було створено декілька екранів майбутнього продукту – пошуковика судового реєстру. Перший екран – головний екран, точка входження користувача, це сторінка пошуку з відображенням фільтрів. Після заповнення поля пошуку та фільтрів бажаними значеннями та натиску кнопки пошуку або ж натиску кнопки «Enter» користувач переходить на другий екран – екран пошуку



з відображенням отриманих результатів. Отримані результати групуються за справами та згортаються/розгортаються та можуть бути додані в обрані. За натиском на кнопку посередині з правого боку екрану розгортається блок обраних користувачем справ, який доступний майже на всіх екранах. Ще один екран – сторінка певного документу реєстру судових рішень України. На ньому відображений сам документ та блоки схожих документів та опція збереження документу. Також була розроблена сторінка обраних документів у особистому кабінеті.

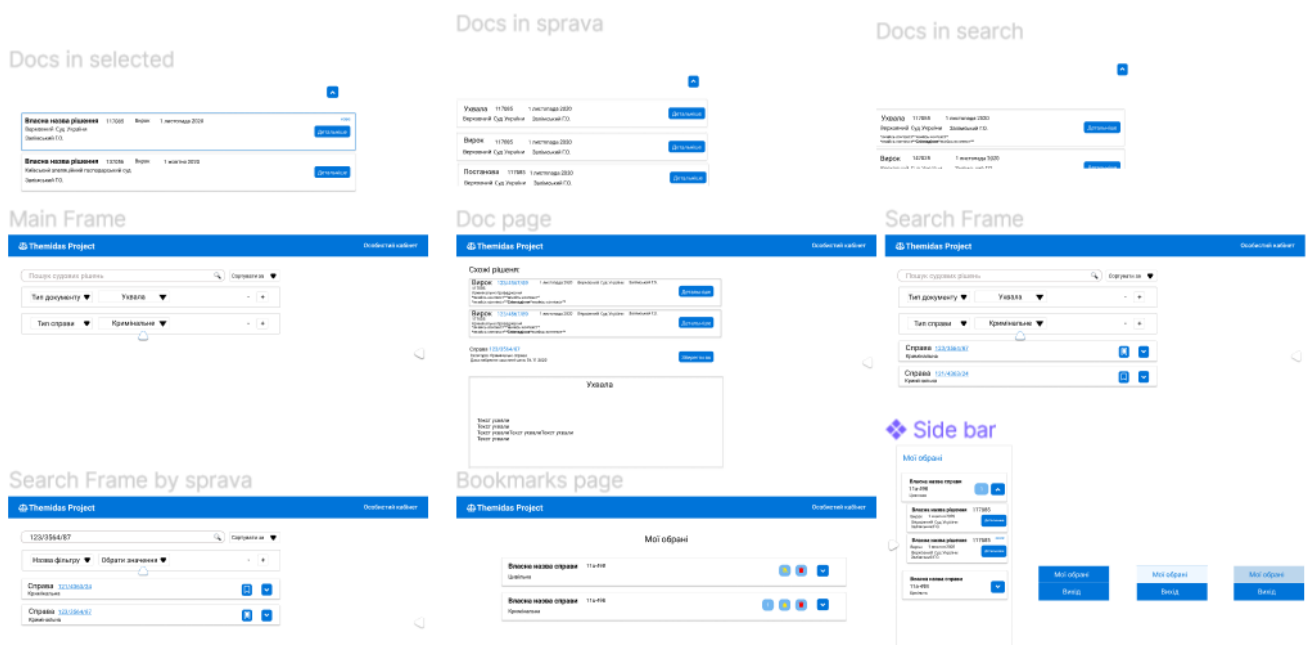
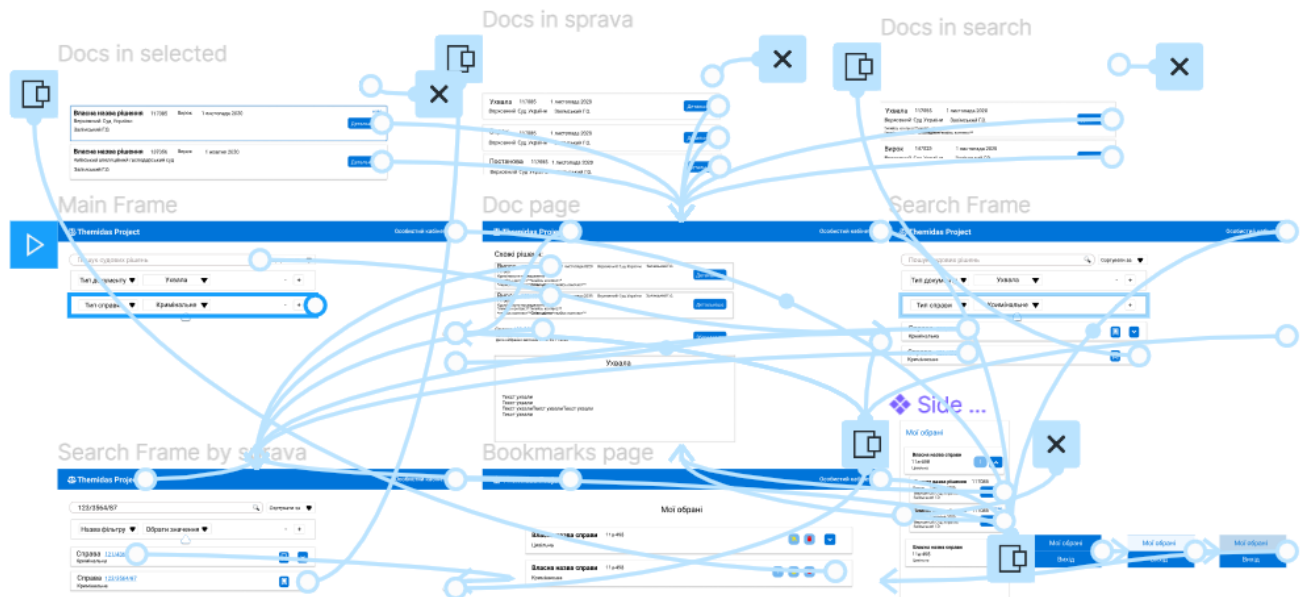


Рисунок 8 Загальний вигляд створеного інтерактивного прототипу у Figma



*Рисунок 9 Відображення зв'язків між екранами створеного інтерактивного прототипу у Figma*

## 1.7 Тестування

Тестування прототипу (версії продукту) є дуже важливою частиною проектування користувацького інтерфейсу. Воно складається з поширення прототипу чи вайфреймів серед тестових користувачів та потенційних клієнтів для оцінки життєздатності конструкції протягом усього циклу розробки.

Випробування прототипу дозволяє завчасно виявити несправності та недоліки та зробити вдосконалення продукту. А саме це дає можливість клієнту/замовнику переглянути прототип та висловити свої думки, фокусній групі виявити баги, проблеми, оцінити юзабіліті. Також прототип може бути протестований на відповідність стандартам безпеки або ж відповідним нормам законодавства. Оцінка прототипу також дозволяє оцінити виробничі витрати та у випадку фінансового обмеження вчасно внести зміни в дизайн або ж виробничі процеси задля вибору дешевших альтернатив. Тестування гарантує можливість поступового опрацювання будь-яких побажань користувача, щоб майбутній споживач міг ефективно та зручно використовувати продукт, що гарантує задоволення споживачів. [9].

Після створення інтерактивного прототипу пошуковика документів судового реєстру в Figma даний прототип був продемонстрований та поширений серед фокус групи, що дало змогу перевірити та спростувати деякі гіпотези. Серед фідбеку отриманого від тестових користувачів було зауваження щодо незрозумілості та незручності користування фільтрами пошуку, тож згодом вони були вдосконалені з урахуванням побажань.

## 2 Реалізація веб системи

### 2.1 Структура веб системи

Веб система була розроблена з використанням клієнт-серверної архітектури, в основі якої лежить RESTful API.

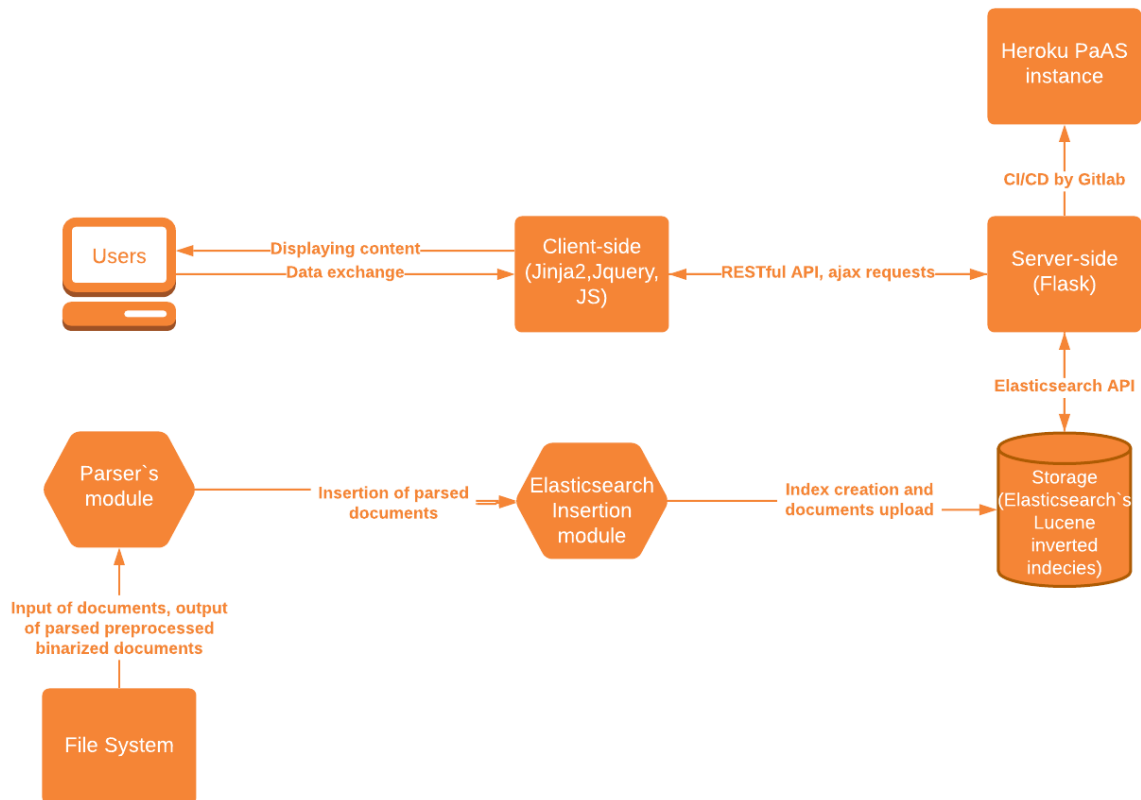


Рисунок 10 Архітектура розробленої веб системи

Структура серверної частини веб-системи складається з *app.py*, що є основним файлом Flask, який керує усіма взаємодіями серверної та клієнтської частини та містить в собі усі необхідні імпорти. Також присутній незалежний модуль власне парсеру – *parser.py*, що відповідає за парсинг документів судового реєстру. Модуль *elastic\_insert.py* виконує автоматизоване завантаження вже пропаршених файлів в Elasticsearch. Також у системі присутнє сховище-меппінг для діставання документів судового реєстру з еластіку - *id\_container.pickle*, що складається з ключа – імені документа, а значенням є кортеж з індексу та айді документу.

Структура клієнтської частини веб-системи складається з модуля `templates`, що містить в собі файли шаблони, та модуля `static`, що включає в себе відповідно підмодулі з `css`, `js` файлами та зображеннями.

## 2.2 Обрані засоби розробки

Для розробки серверної частини веб-системи було обрано мову програмування Python.

Серед фреймворків було обраний відкритий високорівневий Python-фреймворк для розробки веб-систем Django. Основна мета Django - полегшити створення складних веб-сайтів, керованих базами даних. Django надає адмінський веб-інтерфейс для створення та управління об'єктами моделі бази даних. Django слідує концепції MVC (Model-View-Controller) та дотримується принципу Don't Repeat Yourself (DRY), що робить меншим час розробки. Серед відомих сайтів, які використовують Django, є Pinterest, Instagram, Dropbox, Washington Times, Spotify, Reddit, YouTube та Mozilla. [10]

Також було обраний мікрофреймворк для веб-додатків Flask, створений з використанням Python. Flask пропонує широкий вибір, але не змушує використовувати жодних залежностей або макета проекту. Flask дає змогу розробнику самостійно обирати інструменти та бібліотеки для використання. Він також пропонує багато розширень, які полегшують додавання нових функціональних можливостей. Flask базується на наборі інструментів Werkzeug WSGI та рушію шаблонів Jinja2.

Werkzeug - це набір інструментів WSGI (Web Server Gateway Interface, стандарт взаємодії між Python-програмою, яка виконується на стороні сервера, і самим веб-сервером), який реалізує запити, об'єкти відповіді та утилітні функції, що дозволяє побудувати на ньому веб-фрейм.

Jinja2 - це популярний шаблонізатор для Python. Система веб-шаблонів поєднує шаблон із певним джерелом даних для відображення динамічної веб-сторінки. [11].

Вирішенням проблеми пошуку по документам стало використання Elasticsearch. Elasticsearch це вільне програмне забезпечення, пошуковий сервер, розподілений механізм пошуку та аналітики RESTful, що був розроблений на базі Lucene та написаний на Java. Elasticsearch дозволяє ефективно зберігати та індексувати величезні обсяги даних будь-яких типів, підтримуючи NRT (near real-time), що було важливим для пошуковику судового реєстру. Elasticsearch дає можливість вийти за межі звичайного інфопошуку, накопичуючи інформацію, досліджуючи тренди і паттерни в даних. З ростом об'єму даних і запитів до них, ElasticSearch також дає можливість системі рости разом з ними без впливу на ефективність. [12].

Також для розробки фронтенд частини проекту було обрано JavaScript-бібліотеку jQuery з відкритим кодом, що виконує обхід та маніпулювання елементами DOM, обробку подій, анімації та пропонує зручний API для роботи з AJAX.

Для середовища програмування було обране середовище розробки для мови програмування Python PyCharm, розроблена компанією JetBrains на основі IntelliJ IDEA.

## 2.3 Шаблонізатор Jinja2

Jinja2 - це сучасний та зручний рушій шаблонів для Python, створений за зразком шаблонів Django. Він швидкий, широко використовується та захищений завдяки необов'язковому середовищу виконання шаблону в ізольованому середовищі. Синтаксис Jinja2 схожий на Django-шаблонізатор, але при цьому надає можливість використовувати чисті вирази Python та підтримує гнучке розширення системи.

### Особливості Jinja:

- виконання в пісочниці,
- потужна автоматична система екранування HTML для запобігання XSS (Cross-Site Scripting),
- наслідування та включення шаблонів,
- визначення та імпортування макросів в шаблонах,
- підтримка асинхронного вводу\виводу (Asynchronous I/O) та виклику асинхронних функцій,
- підтримка I18N за допомогою Babel,
- вчасна компіляція до оптимального коду python,
- компіляція шаблонів для оптимізованого коду Python вчасна та кешується,
- попередня компіляція шаблонів не є обов'язковою,
- легкий дебаг коду (вказування номерів рядків з отриманими помилками),
- розширювані фільтри, тести, функції та навіть синтаксис.

Шаблон Jinja це звичайний текстовий файл, в якому за допомогою різних виразів, можна вказати які значення в шаблоні повинні бути замінені. В шаблоні також можуть бути крім виразів різні конструкції, які керують логікою обробки даних при формуванні документа за шаблоном. Jinja використовує такі роздільники (вони можуть бути змінені):

- `{{...}}` - для виразів ,
- `{%...%}` - для структур, таких як for, if ,
- `{# ... #}` – для коментарів.

Змінні шаблону визначаються в словнику, який передається шаблоном, нею може бути будь-який об'єкт Python. При наявності можна звертатися до атрибутів змінної або до елементів в шаблоні. Змінні можна модифікувати за допомогою фільтрів, які відокремлюються від змінних символом `|` .

Найпотужніша частина Jinja - це наслідування шаблонів. Наслідування шаблону дозволяє створити базовий шаблон «скелет», який містить усі загальні

елементи сайту та визначає блоки, які можуть замінити дочірні шаблони. Наслідування відбувається з використанням тегу `{% extends %}`. У одному файлі можна включати декілька тегів `{% extends %}`, але одночасно може виконуватися тільки один.



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

### Лістинг 1 Приклад використання тегу `{% extends %}`

Тег `{% include %}` корисний для включення шаблону та повернення відтвореного вмісту цього файлу в поточний простір імен. Включені шаблони мають доступ до змінних активного контексту за замовчуванням. У Jinja 2.2 є можливість позначити включення «*ignore missing*;» - у цьому випадку Jinja проігнорує висловлювання, якщо шаблон, який буде включено, не існує. У поєднанні з «*with context*» або «*without context*» його слід розміщувати перед твердженням про видимість контексту, наприклад:



```
{% include "component.html" ignore missing with context %}
```

### Лістинг 2 Приклад 1 використання тегу `{% include %}`

Також можна надати список шаблонів, які перевіряються на існування перед включенням, при цьому буде включено перший існуючий шаблон. Якщо вказати «*ignore missing*;», то нічого не буде зрендерено, якщо жоден із шаблонів не існує, інакше це спричинить виняток:



```
{% include ['special_sidebar.html', 'sidebar.html'] ignore missing %}
```

### Лістинг 3 Приклад 2 використання тегу `{% include %}`



Jinja підтримує як і екранування вручну кожної змінної, так і автоматичне екранування всього. За замовчуванням конфігурація не передбачає автоматичного екранування, проте її можна змінити. [13].

## 2.4 Опис розробки фронтенд частини

При виборі засобів розробки фронтенд частини було зрозуміло, що використання тільки голого HTML, CSS та Vanilla JS буде недостатньо ефективним рішенням, проте у використанні популярних фреймворків такі як Angular чи React немає нагальної потреби. Тож зваженим рішенням стало використання шаблонізатору Jinja2, який і так є інтегрованим та використовується у Flask. Таким чином фронтенд частина не використовує зайві ресурси та її виконання є більш ефективним. Було використано Sass – препроцесор CSS, який дозволяє використовувати змінні, математичні операції, комбінації, цикли, функції, імпорт та інші цікаві функціональні можливості, які роблять написання CSS набагато потужнішим, а саме його новий синтаксис Scss. Було використано фреймворк Bootstrap версії 4.6 для полегшення створення адаптивного mobile-first сайту. Також було обрано JavaScript-бібліотеку jQuery для покращення роботи з елементами DOM.

Після встановлення Flask до створеного проекту необхідно було підключити шаблонізатор Jinja2 для рендеру сторінок. Підключення відбувається за допомогою імпорту `render_template`.



```
1 from flask import Flask, render_template, send_from_directory, jsonify, abort, request, redirect, session
```

*Лістинг 4 Підключення Jinja*

Рендеринг шаблону відбувається за допомогою виклику `render_template()`, де у якості обов'язкового аргументу приймається ім'я файлу, а необов'язкових - `title` та `description`.



```
1 @app.route('/', methods=['GET', 'POST'])
2 def index():
3     return render_template('index.html')
```

#### *Лістинг 5 Рендеринг шаблону*

Було розроблено основний шаблон *base.html*, який визначає скелет документів HTML. В ньому імпортувалися усі необхідні підключення такі як Bootstrap, jQuery та інше. В нього включається такий компонент як навібар за допомогою тегу `{% include "navbar.html" %}`, який є необхідним на всіх сторінках пошуковика, а також він містить блок самого контенту `{% block content %}`, який буде заповнений за допомогою різних дочірніх шаблонів *index.html* та *doc.html* в залежності від необхідного вмісту певної веб-сторінки.



```
1 <body>
2     {% block navbar %}{% endblock %}
3     <div class="container">
4         {% block content %} {% endblock %}
5     </div>
6 </body>
```

#### *Лістинг 6 Демонстрація блоків navbar та content у base.html*

Дочірній шаблон *index.html* відповідає за рендеринг контенту на основну сторінку пошуковика судового реєстру – сторінку з пошуком. Для визначення

що цей шаблон є дочірнім від *base.html* використовується тег `{% extends 'base.html' %}`.



```

1 {% extends 'base.html' %} <!-- extends to inherit from base.html -->
2 {% block content %} <!-- specific block with content -->
3 <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/daterangepicker
/daterangepicker.css"/>
4 <link rel="stylesheet" href=../static/css/index.css>
5 <link rel="stylesheet" href=../static/css/cards.css>
6
7 {# filters #}
8 <div class="filters-container container-sm">
9     ...
10 </div>
11 {# results #}
12 <div class="results-container container"></div>
13 <script type="text/javascript" src="https://cdn.jsdelivr.net/momentjs/latest/moment.min.js"></script>
14 <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/daterangepicker/daterangepicker.min.js">
</script>
15 {% endblock %}

```

### *Лістинг 7 Дочірній шаблон index.html*

Він складається з контейнеру, що містить в собі фільтри для пошуку та контейнеру з «картками» отриманих результатів пошуку. Останній контейнер заповнюється змістом динамічно за допомогою js. З використанням jQuery при пошуку користувачем відправляється ajax get запит на url `'/search'` з введеними даними для пошуку користувачем та у відповідь приходять дані про документи судового реєстру, що відповідають запиту, на основі яких відбувається рендеринг вмісту контейнеру результатів.

За допомогою проведення інтерв'ю та опитувань серед потенційних користувачів, як було вказано вище, були зроблені висновки про зручність сніпетів, які б демонстрували входження пошукового запиту у документ, та групування результатів за справами. Для зручності було також вирішено реалізувати можливість згортки/розгортки знайдених результатів, які відносяться до однієї справи. Через те що в MVP продукту були реалізовані лише парсери по деяким видам документів через складнощі, що виникали через те ще кожен тип судового документу відрізняється структурою, ключовими словами та кількістю структурних одиниць, а також через відсутність точних правил форматування, не всі поля документів які використовуються в фільтрах були

проаналізовані, тож розроблені фільтри, по яким пошук не відбувається у MVP були приховані на деплої.

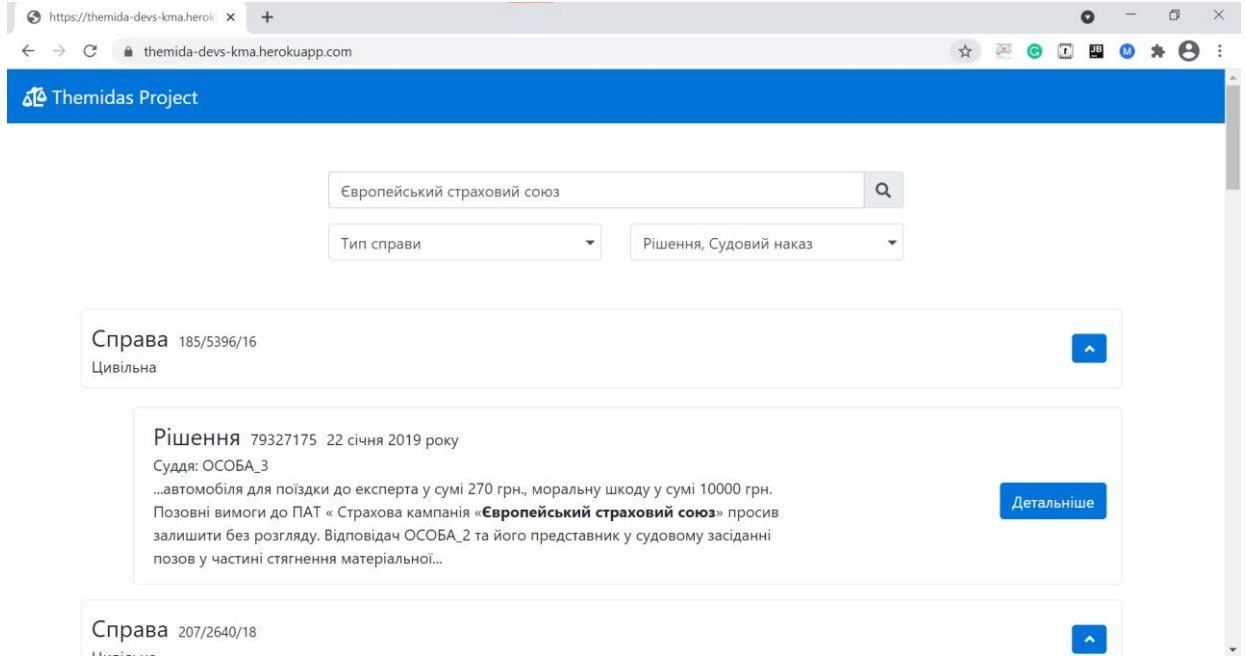


Рисунок 11 Відображення сторінки з результатами пошуку, яка була зрендерена шаблоном `index.html`, демонстрація сніпету з виділеним пошуковим запитом

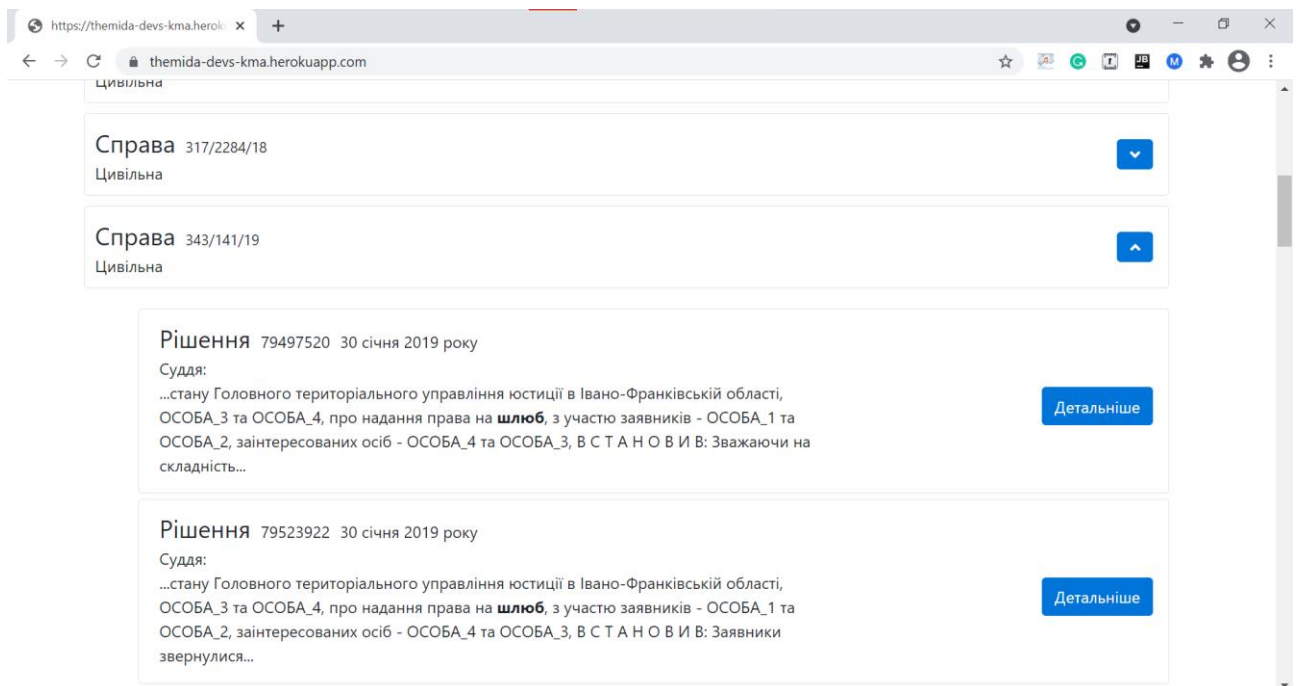


Рисунок 12 Демонстрація групування результатів за справами та можливості згортання/розгортання справ

Одним з дочірніх шаблонів є шаблон *doc.html*, який рендериться при відображенні сторінці конкретного документу судового реєстру.



```

1 {% extends 'base.html' %} <!-- extends to inherit from base.html -->
2 {% block content %} <!-- specific block with content -->
3     <link rel="stylesheet" href=../static/css/doc.css>
4     <link rel="stylesheet" href=../static/css/cards.css>
5     <div class="about-sprava-container">
6         .....
7     </div>
8     <div class="doc-container rounded border">
9         .....
10    </div>
11 {% endblock %}

```

*Лістинг 8 Дочірній шаблон doc.html*



```

1 @app.route('/doc/', methods=['GET'])
2 @app.route('/doc/<doc_id>', methods=['GET'])
3 def doc(doc_id):
4     ...
5     return render_template('doc.html', data=doc_parts, name=doc_id, doc_type = doc_type)

```

*Лістинг 9 Рендеринг doc.html з присвоєнням значень змінних*

Так як Jinja підтримує використання змінних, то вони активно використовуються у шаблонах у даному проєкті. Наприклад, у шаблон *doc.html* передається такі змінні як унікальний номер документу, тип документу та частини розпаршеного проаналізованого документу – на серверній частині документи судового реєстру аналізуються та розбиваються на логічні блоки. На клієнтську частину передається структура у вигляді об'єкту json, що містить всі структурні елементи судового документу. Це дає змогу забезпечити стаке відображення для всіх документів судового реєстру незалежно від їх початкового форматування та зробити красиве та зручне форматування, а також стилізувати деякі частин: додати відступів між вступною, описовою, мотивувальною та

резолютивною частинами, виділити деяких частин жирним шрифтом такі як, наприклад, ПБ судді, розмістити інакше певні текстові елементи для більш зручного візуального представлення.



```

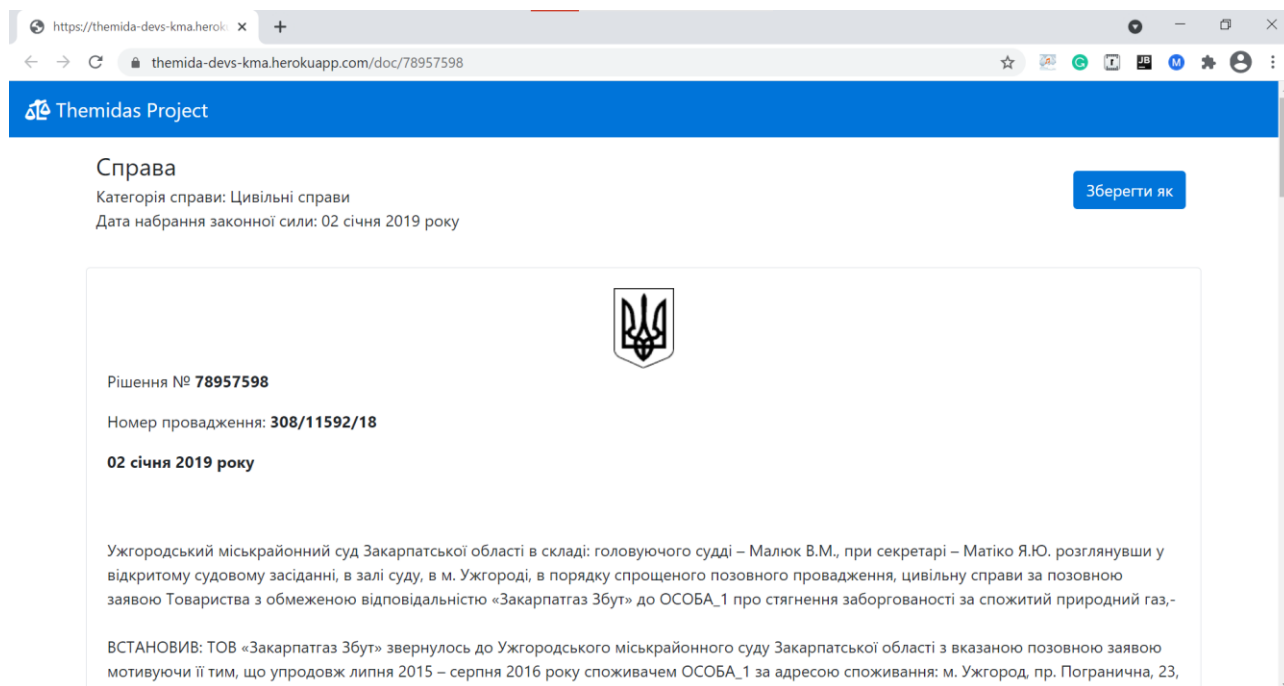
1 <div class="info-block">
2   <p class="doc-num">{{ doc_type }} №
3     <b>{{ name }}</b>
4   </p>
5   <p class="case-num">Номер провадження:
6     <b> {{ data[0] }}</b>
7   </p>
8   <p class="date">
9     <b> {{ data[1] }}</b>
10  </p>
11 </div>

```

### Лістинг 10 Приклад використання змінних у doc.html

The image shows two screenshots of the Ukrainian State Court Register (Єдиний державний реєстр судів) website. The top screenshot displays a court decision (РІШЕННЯ) from the Bilgorod-Dnistrovskyi District Court, dated March 11, 2011. The bottom screenshot shows another court decision (РІШЕННЯ) from the Kostiantynivka District Court, dated April 1, 2011. Both screenshots show the court's name, the date of the decision, and the names of the judges and the secretary.

Рисунок 13 Приклад різного форматування документів судового реєстру



*Рисунок 14 Відображення сторінки документу, яка була зрендерена шаблоном doc.html*

## Висновок

У цій роботі була розглянута задача проектування та розробки користувацького інтерфейсу. Для цього були розглянуті різні підходи до проектування користувацького інтерфейсу, UI/UX підходи та основні етапи проектування користувацького інтерфейсу. Також було розглянуто варіанти реалізації користувацького інтерфейсу.

Було проведено дослідження правової сфери та труднощі, з якими стикаються люди, що відносяться до неї, при автоматизації процесів для визначення продукту для розробки, та після аналізу отриманих результатів було обрано ідею розробки пошуковика по документам Єдиного державного реєстру судових рішень України.

Для проектування користувацького інтерфейсу був використаний підхід «design thinking», що забезпечило ефективне та швидке проектування та зручний користувацький інтерфейс. Були описані та проведені такі етапи проектування як: визначення продукту, дослідження користувачів, створення user story maps, прототипування, а саме створення вайфреймів та інтерактивного прототипу, та тестування. Аналіз результатів кожного з етапів дав змогу зробити висновки щодо завдань та потреб потенційних користувачів та покращити не тільки інтерфейс, а й функціонал веб системи, а саме: було розроблено групування знайдених документів при пошуку по справам, розроблені пошукові сніпети документів з виділенням пошукового запиту, сталий вигляд форматування документів з візуальним виділенням деяких ключових елементів та розроблений пріоритезований пошук.

У практичній частині роботи була реалізована фронтенд частина MVP веб системи пошуковика. Для цього були обрані такі засоби розробки як шаблонізатор Jinja2, Bootstrap, SASS, jQuery та JS, та їх вибір був обґрунтований у роботі. Розроблена веб система була задеплойована з використанням Heroku.



Було зроблено висновок, що проектування користувацького інтерфейсу є важливим та необхідною частиною розробки веб системи. Правильне проектування користувацького інтерфейсу допомагає зрозуміти бажання та реальні потреби потенційного клієнта, що дає змогу не тільки зробити продукт кращим, більш корисним та цікавим для користувача, а й ефективно підійти до самої розробки продукту, зробивши її менш енерго- та ресурсозатратною.

## Список літератури

1. User Experience (UX): Process and Methodology [Електронний ресурс] – Режим доступу до ресурсу: <https://uiuxtrend.com/user-experience-ux-process/> .
2. UX Approach & Principles [Електронний ресурс] – Режим доступу до ресурсу: <https://uxls.org/guide-to-ux/ux-approach-and-principles/>.
3. Do you know the Importance of UI/UX Development? [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://medium.com/@thinkwik/do-you-know-the-importance-of-ui-ux-development-773eae38436e> .
4. Babich N. The UX Design Process: Everything You Need to Know [Електронний ресурс] / Nick Babich. – 2020. – Режим доступу до ресурсу: <https://xd.adobe.com/ideas/guides/ux-design-process-steps/> .
5. Allabarton R. What Is The UX Design Process? A Complete, Actionable Guide [Електронний ресурс] / Rosie Allabarton – Режим доступу до ресурсу: <https://careerfoundry.com/en/blog/ux-design/the-ux-design-process-an-actionable-guide-to-your-first-job-in-ux/#1-what-is-ux-design> .
6. Patton J. User Story Mapping: Discover the Whole Story, Build the Right Product 1st Edition [Електронний ресурс] / Jeff Patton // O'Reilly Media – Режим доступу до ресурсу: 8. [www.jpattonassociates.com/wp-content/uploads/2015/03/story\\_mapping.pdf](http://www.jpattonassociates.com/wp-content/uploads/2015/03/story_mapping.pdf) .
7. Smith Q. Prototyping User Experience [Електронний ресурс] / Quincy Smith. – 2019. – Режим доступу до ресурсу: <https://www.uxmatters.com/mt/archives/2019/01/prototyping-user-experience.php> .
8. Wireframing [Електронний ресурс] – Режим доступу до ресурсу: <https://xd.adobe.com/ideas/process/wireframing/> .
9. Ryan V. TESTING AND EVALUATING A PROTOTYPE - WHY? [Електронний ресурс] / V. Ryan // 2013 – Режим доступу до ресурсу: [https://technologystudent.com/despro\\_flsh/evalintegr1.html](https://technologystudent.com/despro_flsh/evalintegr1.html) .

10. Django Tutorials [Электронный ресурс] – Режим доступа до ресурсу:  
<https://realpython.com/tutorials/django/> .
11. What is Flask Python [Электронный ресурс] – Режим доступа до ресурсу:  
<https://pythonbasics.org/what-is-flask-python/> .
12. What is Elasticsearch? [Электронный ресурс] – Режим доступа до ресурсу:  
<https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html> .
13. Jinja Documentation (2.11.x) [Электронный ресурс] – Режим доступа до ресурсу: <https://jinja.palletsprojects.com/en/2.11.x/> .