МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Аліна САВЧЕНКО

"__"_____2022 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"БАКАЛАВРА"

ЗА СПЕЦІАЛЬНІСТЮ 122 «КОМП'ЮТЕРНІ НАУКИ»

Тема: "Прототип графічного редактора для візуалізації дизайн-проектів"

Виконавиця: Личманюк Ірина Вікторівна

Керівник: к.т.н., доцент Колісник Олена Василівна

Нормоконтролер: Олександр ШЕВЧЕНКО
(підпис)

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Освітній ступінь: "Бакалавр"

Галузь знань, спеціальність, освітньо-професійна програма:

12 "Інформаційні технології, 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології" .

ЗАТВЕРДЖУЮ

Завідувач кафедри

Аліна САВЧЕНКО

"___" _____ 2022 р.

ЗАВДАННЯ

на виконання дипломного проекту студентки

Личманюк Ірини Вікторівни

(прізвище, ім'я, по батькові)

1. **Тема роботи проекту (проекту):** «Прототип графічного редактора для візуалізації дизайн-проектів» затверджена наказом ректора № 454/ст. від 29.04.2022.
2. **Термін виконання роботи проекту (проекту):** 09.05.2022 – 13.06.2022.
3. **Вихідні дані до роботи (проекту):** дані про графічні редактори, принципи створення архітектури веб-додатка, документація.
4. **Зміст пояснювальної записки (перелік питань, що підлягають розробці):** аналіз існуючих графічних редакторів, аналіз мов програмування, вибір середовища для розробки, розробка архітектури на Python Django, розробка фронтенд-частини додатка, UX аналіз.
5. **Перелік обов'язкового графічного матеріалу:** рисунки, презентація.

6. Календарний план-графік

№ з/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Опрацювати літературні джерела за темою дипломного проекту	10.05.2022р. – 13.05.2022р.	
2	Провести консультації з дипломним керівником щодо змісту та термінів створення проекту	14.05.2022р. – 17.05.2022р.	
3	Написати перший розділ: вибір технічних засобів	18.05.2022р. – 22.05.2022р.	
4	Написати другий розділ: розробка веб-додатка	23.05.2022р. – 26.05.2022р.	
5	Написати третій розділ: демонстрація створеного веб-додатка	27.05.2022р. – 02.06.2022р.	
6	Оформити пояснювальну записку	03.06.2022р. – 06.06.2022р.	
7	Підготувати презентацію та доповідь	07.06.2022р. – 13.06.2022р.	

Дата видачі завдання: 10.05.2022 р.

Керівник дипломного проекту _____ Олена КОЛІСНИК
(підпис керівника)

Завдання прийняла до виконання _____ Ірина ЛИЧМАНЮК
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Прототип графічного редактора для візуалізації дизайн-проектів» представлена на 54 сторінках, містить 22 рисунки, 13 літературних джерел.

Мета дипломного проекту: аналіз проблем та можливостей графічних редакторів, аналіз та визначення технологій для розробки веб-додатків, створення програмного рішення для простої та ефективної візуалізації дизайн-проектів.

Об'єкт дослідження: процес створення та обробки візуалізацій із зображень.

Предмет дослідження: графічний веб-редактор для створення візуалізацій для проектів, мудбордів.

Метод дослідження: пошук інформації в літературних джерелах (на веб-ресурсах), аналіз існуючих графічних редакторів, аналіз та порівняння різних технічних засобів та технологій розробки, створення архітектурних рішень та розробка фронтенд-частини додатка, дослідження UX-принципів.

Результат проекту: створений прототип графічного редактора.

ВЕБ-ДОДАТОК, ГРАФІЧНИЙ РЕДАКТОР, PYTHON DJANGO, HTML, CSS, JAVASCRIPT, АРХІТЕКТУРА, URL, ІНТЕРФЕЙС, UX.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

СУБД – система управління базами даних

CMYK – субтрактивна колірна модель, використовується у поліграфії

CSS – каскадні таблиці стилів

HEX – 16-бітна колірна модель для використання у веб-середовищі

HSV – розширена версія RGB

HTML – мова розмітки гіпертексту

HTTP – протокол передачі гіпертекстових документів

RGB – адитивна колірна модель

UI – інтерфейс користувача

URL – стандартизована адреса ресурсу

UX – досвід користувача

ВСТУП

З стрімким розвитком комп’ютерних технологій відбувається перехід багатьох галузей, зокрема дизайну всіх напрямків, в цифровий простір. Зростають вимоги (кількісні та якісні) до проектування та втілення проектів. Для їх створення дизайнери використовують графічні редактори – програмні засоби для обробки медійного контенту. Окрім основних функцій обробки зображень, з їх допомогою можна створювати та редагувати типографічні матеріали, працювати з візуальними ефектами, створювати власні об’єкти.

Початковим та одним з основних етапів розробки дизайн-проектів є візуалізація. Це процес збору зразків стильових рішень та ідей і їх подальшого об’єднання в так званий мудборд (mood board – “дошка настрою”). Цей етап може бути відокремленим від інших, оскільки він передує обговоренню та затвердженню перед самим проектуванням та реалізацією дизайну.

Наявні графічні редактори можуть задовільнити потребу користувачів у створенні візуалізацій, проте зазвичай це не є їх основним призначенням, а лише одна із можливостей внутрішнього функціоналу. У зв’язку з цим існує необхідність створити окремий додаток.

Новизна цього додатка полягає в підході до розробки графічного редактора – мінімум функціоналу, а відповідно, й зниження емоційного навантаження на користувача та зменшення розфокусування, і водночас достатня кількість інструментів для забезпечення однієї конкретної потреби – створення візуалізацій (мудбордів) для проектів.

Незважаючи на те, що візуалізації найчастіше використовуються в дизайн-проектах, їх застосування можливе і звичайними користувачами. Саме в повсякденному житті такі візуалізації зазвичай звучать як мудборд. Отож, крім дизайнерів, розроблюваним додатком зможуть користуватися значно більше людей з різних сфер, оскільки він і вузькоспеціалізований, і універсальний.

РОЗДІЛ 1
ВИБІР ТЕХНІЧНИХ ЗАСОБІВ

1.1. Аналіз існуючих графічних редакторів

Комп'ютерна графіка – створення зображень з допомогою комп'ютерних технологій для їх відображення, зберігання та використання на різних носіях.

Основними видами комп’ютерної графіки є: растрова та векторна. Для опису зображень у векторній графіці використовуються вектори, а у растровій зображення описуються як масив пікселів (кольорових точок).

В сучасному цифровому просторі існує багато програмних засобів – графічних редакторів, для роботи з графікою різного виду. Відрізняються вони між собою метою (на яку цільову аудиторію та на які області спеціалізованої діяльності спрямовані), типом підтримуваних графічних зображень/об’єктів (растрові, векторні, тривимірні, фрактальні), функціоналом (задачі, які покриваються використанням цих програм) та платформами, для роботи на яких створені.

Розглянемо найпопулярніші растрові і векторний графічні редактори, їх переваги та недоліки.

1.1.1. Adobe Photoshop

Найвідомішим графічним редактором сьогодення є Adobe Photoshop (рис.1) (далі – Photoshop). Він широко використовується спеціалістами в

області графічного дизайну та фотографії, відеомейкерами та ілюстраторами. Широкий функціонал програми здатен покрити найскладніші запити діджитал митців: від простої обробки зображень до створення складних креативів. Це і є головною перевагою Photoshop серед конкурентів.

До інших переваг можна віднести:

- настроюваний інтерфейс;
- має підтримку растрової та векторної графіки;
- величезна бібліотека інструментів для типографіки та малювання;
- зручна пошарова структуризація робіт;
- багато інструкцій та майстер-класів по створенню контенту в програмі.

Photoshop має й деякі недоліки:

- обов’язкова платна підписка (проте існують безкоштовні додатки для смартфонів: Photoshop Express і Photoshop Camera);
- складний для початківців;
- займає багато ресурсів комп’ютера. [1]

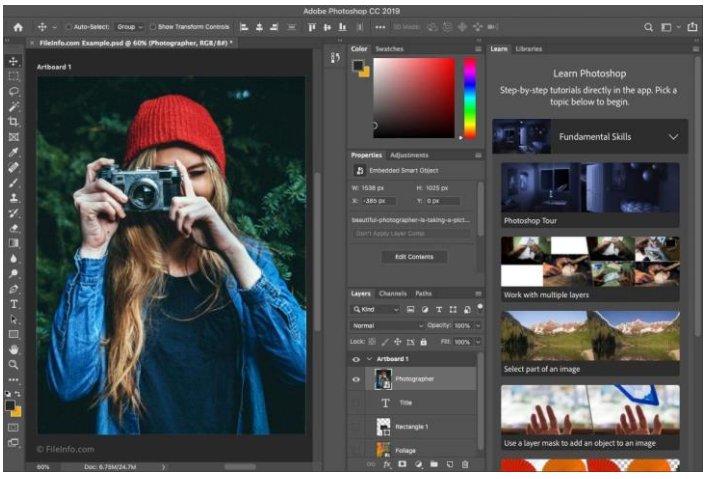

Рис.1. Інтерфейс графічного додатка Adobe Photoshop

1.1.2. Microsoft Paint

Одним з найзвичніших для пересічних користувачів є графічний редактор Paint (рис.2) від компанії Microsoft. Він працює на базі операційної системи Windows та йде встановленим на комп’ютер за замовчуванням.

Редактор має простий інтерфейс, що дає можливість розібратися з функціоналом навіть новачкам. Однак це і є його головним мінусом – програма надто примітивна для складних робіт, виникає потреба в допоміжних інструментах або і в повному переході в інші програми після одного з етапів створення/редагування зображення.

Отож, основні переваги графічного редактора Paint:

- безкоштовний;
- встановлений на комп’ютер (з ОС Windows) за замовчуванням;
- простий для вивчення основ комп’ютерної графіки початківцями;
- оновлений інтерфейс (переміщена та розширена панель інструментів).

Недоліки Paint:

- недосконало продуманий UX;
- немає підтримки багатошарової структури зображень;
- замалий функціонал для розробки великих проектів.

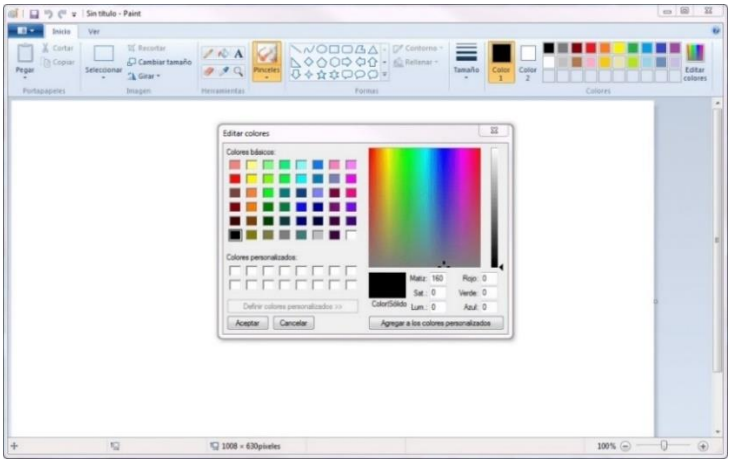

Рис.2. Інтерфейс графічного додатка Paint

1.1.3. Figma

Figma (рис.3) – порівняно новий сервіс для створення дизайн-рішень. Використовується, в основному, для розробки інтерфейсів мобільних, десктопних та веб-додатків.

На відміну від Photoshop та Paint, цей графічний редактор – мультиплатформний, оскільки працює у веб-середовищі без прив'язки до комп'ютера. Всі дані зберігаються у хмарі, завдяки чому одразу вся команда може мати доступ до перегляду та роботи з проектом.

Добре продуманий функціонал є вузько-направленим, проте повністю покриває потреби дизайнерів. Крім цього, програма включає в себе й інші інструменти, не пов'язані напряму з графікою (наприклад, формування характеристик об'єктів для подальшої розробки, симуляція роботи сайту із взаємодією елементів).

Хоча ми розглядаємо цей графічний редактор як сервіс для роботи з растровою графікою, також в ньому є можливість створювати векторні зображення та ілюстрації.

З інших переваг Figma:

- зручний інтерфейс;
- багатошарова структура роботи (аналогічно до Photoshop);
- крім основної веб-версії, доступна десктопна версія для MacOS та Windows;
- можна без дискомфорту працювати з безкоштовним тарифом.

Недоліки цього графічного редактора:

- немає підтримки CMYK (хоча програма і не є призначена для поліграфії);
- мало інструментів для обробки зображень. [2]

Рис.3. Інтерфейс графічного додатка Figma

1.1.4. Canva

Окрім стандартних графічних редакторів існує багато корисних сервісів та інструментів для роботи з графікою. Однією з таких є Canva (рис.4) – платформа для графічного дизайну. З її допомогою можна створювати будь-який медійний контент: презентації, інфографіку, різні креативи, веб-дизайн тощо.

Проте Canva спрямована не на професіоналів. Головною метою сервісу є – зробити дизайн доступним кожному. І тому вона працює за принципом drag-and-drop: з допомогою великого банку шаблонів, зображень, шрифтів та ілюстрацій користувачі без досвіду в дизайні можуть створювати власні роботи.

З переваг Canva можна виділити:

- юзер-гайд для новачків;
- добре розроблений UX;
- можливість в безкоштовній версії зберігати файли як JPEG та PDF.

Недоліки, які має сервіс Canva:

- не можна поєднувати елементи з різних шаблонів;
- обмежена кількість стандартних шаблонів і форматів збереження в безкоштовній версії;
- не підходить для цілей професіонального дизайну інтерфейсів. [3]

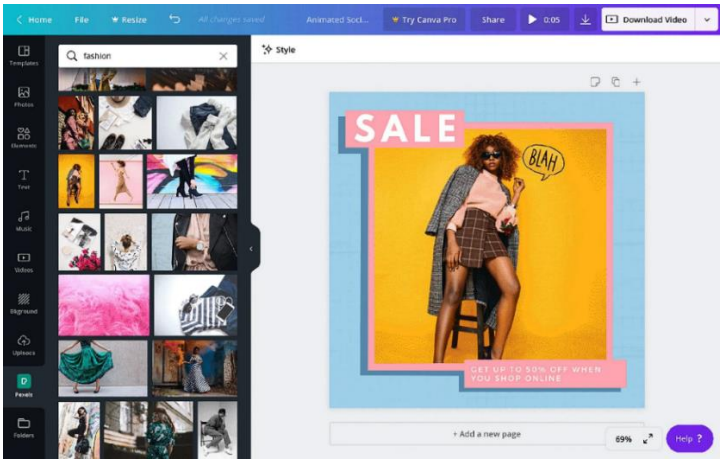

Рис.4. Інтерфейс графічного додатка Canva

1.2. Огляд мов програмування

Платформа, для якої розробляємо додаток – веб, а отже, є сенс розглянути та оцінити можливості різних мов програмування в межах цього середовища. В рейтингу найпопулярніших зараз: JavaScript, Java, Python, C#. Кожна з них має свою аудиторію фанатів, які не бажають застосовувати інші для розробки. Проте ефективним є саме метод поєднання в проекті декількох мов програмування. Причиною є те, що незважаючи на схожі загальні можливості, усі мають свою унікальну функціональність та інші характеристики, за якими і можна створити ідеальний стек технологій і покрити всі потреби при розробці.

1.2.1. JavaScript

Одним із лідерів серед мов програмування є JavaScript. Використовується як мова програмування на стороні клієнта на майже 98% відсотках усіх веб-сайтів. JavaScript дозволяє налаштувати динамічний та інтерактивний вміст користувацьких інтерфейсів, візуальний дизайн, анімовану графіку та інші складні функції на веб-сервісах.

На відміну від інших мов, які розглядаються, JavaScript є переважно мовою клієнта, оскільки він виконується у веб-браузері. Однак все частіше він використовується і на стороні сервера з такими платформами, як Node.js (платформа з відкритим вихідним кодом, виконує високопродуктивні мережеві застосунки, написані мовою JavaScript) і Frameworks (виконує роль структурної основи, викликає і використовує код розробника).

JavaScript має найбільший відсоток використання в порівнянні з іншими мовами програмування. Для початку його вивчення достатньо розуміння принципів та синтаксису HTML і CSS. Це дає можливості швидкого старку новачкам в програмуванні, а також сприяє підвищеному інтересу зі сторони досвідчених розробників, які працюють з іншими мовами.

Кросплатформність – застосування на різних платформах (десктоп, мобільні додатки, веб), є однією з головних характеристик JavaScript. Серед інших:

- моментальне виконання на стороні браузера;
- зменшення навантаження на сервер;
- простота у способі імплементації.

Проте є недолік – відображення і робота веб-сторінок може бути різним в залежності від браузера. Але і ця проблема поступово зникає, оскільки браузери вчаться підтримувати все більше і більше форматів та функцій. [4][5]

1.2.2. Java

Java є об’єктно-орієнтованою мовою програмування високого рівня. Базується на принципі “write once, run anywhere” (WORA), що означає, що скомпільований код працюватиме на всіх Java-сумісних платформах без необхідності перекомпіляції. Саме тому ця мова програмування стала стандартом для додатків, які можна використовувати незалежно від платформи (Mac, Windows, Android, iOS тощо).

Як результат, Java відома своєю підтримкою на різних платформах, від центрів обробки даних на мейнфреймах (великі за розмірами, обсягом пам'яті, потужністю обробки та високим рівнем надійності комп'ютери) до смартфонів. Серед можливостей Java – доступ та маніпуляції з найважливішими функціями комп'ютера, такими як файлова система, графіка і звук для будь-якої складних сучасних програм.

Java широко використовується у веб-розробці та розробці додатків так само, як і у роботі з Big Data. Термін Big Data, або «великі дані», відноситься до набору прийомів і методів для аналізу та обробки великих обсягів структурованих і неструктурованих даних для отримання нових якісних знань. Загалом, це інформація, яку неможливо обробити класичним способом через величезний обсяг. [6]

Серед прикладів використання Java у серверній частині багато популярних веб-сайтів, у тому числі Google, Amazon, Twitter і YouTube.

Великій популярності сприяє поява нових фреймворків Java, таких як Spring, Struts і Hibernate. Завдяки тісній мережі розробників по всьому світу вивчення Java стає все більш доступним.

Java вважається складнішою для вивчення, ніж Python, але легшою, ніж C-подібні мови. Основна причина полягає в тому, що Java змінилася у сторону C, а Python – до напрямку Java. Великою перевагою вивчення Java є те, що після неї значно легше освоювати щось типу Python.

1.2.3. Python

Python є адаптивною та високоефективною мовою програмування. Це пояснюється тим, що Python пропонує для веб-сайтів можливості динамічного введення тексту. Ця універсальна мова програмування дозволяє розробникам створювати інформаційні додатки, програми з графікою, ігри, різні утиліти, веб-додатків та багато іншого. Кожен створений за допомогою Python веб-

додаток буде відрізнятися серед аналогів високофункціональністю і успішністю.

Ще одна перевага створення веб-додатка на Python полягає в тому, що можна використовувати та публікувати його безкоштовно. Python відомий як продукт з відкритим кодом, а тому всю необхідну інформацію для розробки можна знайти в Інтернеті. Існує величезна кількість готових рішень (модулів) для реалізації різноманітної функціональності на Python. Крім того, їх копіювання та вбудовування у власних додатках не обмежене.

Python відрізняється легкою інтеграцією з веб-сервісами, структурами даних і додатками на основі графічного інтерфейсу. Як мова загального призначення на стороні сервера Python використовується для виконання різноманітних завдань: від простих скриптів до просунутих веб-додатків і штучного інтелекту, оскільки має високу надійність та швидкість виконання. Він цікавить також розробників, що займаються розробками в області наук про дані або машинним навчанням. Python використовується для розробки пакетів 2D-зображень і 3D-анімації (Blender, Inkscape, Autodesk тощо), для наукових і обчислювальних програм (FreeCAD і Abacus), для розробки веб-сайтів відомих компаній, таких як Pinterest, Facebook, Google (YouTube), Dropbox, Microsoft, Mozilla та Intel.

Порівняно з іншими мовами програмування, синтаксис Python досить простий, завдяки чому підвищується читабельність коду і, відповідно, спрощується його подальша підтримка.

Python надає розробникам широкий спектр фреймворків. Існує два типи фреймворків Python – Full Stack Framework і Non-Full Stack Framework. Повний стек фреймворків Python пропонує розробникам повну підтримку, включно з базовими компонентами, такими як генератори форм, перевірка форм і макети шаблонів. Замість того, аби писати схожий код для кожного додатка, розробники на Python можуть користуватися готовими рішеннями – компонентами фреймворку.

Python відрізняється надійністю та швидкістю виконання. [7]

Для розробки на кожній з платформ існує багато фреймворків, завдяки яких значно спрощується проектування архітектури додатків. Для веб-середовища найпопулярнішими є Django та Flask.

1.2.4. C#

C# – універсальна об'єктно-орієнтована мова програмування високого рівня, розроблена компанією Microsoft. C# відноситься до сімейства мов програмування C, тому вивчення C# полегшить вивчення C, C++ або Java, оскільки ці мови використовують схожий синтаксис і принципи програмування.

Управління розподілом пам’яті є одним із важливих завдань для підтримки продуктивності програми. Для цієї цілі в C# є вбудований сортувальник сміття – менеджер пам’яті, який відстежує невикористані об’єкти та автоматично видаляє їх. Однак продуктивність мови не найкраща. ЇЇ можна виміряти часом компіляції та фактичною продуктивністю програми. У порівнянні зі своїм найближчим аналогом Java, C# має подібний час компіляції, але в порівнянні з C++, продуктивність C# є нижчою.

C# багато в чому покладається на ресурси .NET (платформа для розробки додатків) для роботи на різних операційних системах або платформах. Однак, якщо .NET не розглядається як основний технологічний стек, сам по собі C# не гнучкий, Відповідно, потрібно використовувати різні середовища виконання і адаптувати код відповідно до системних вимог тієї чи іншої платформи.

Незважаючи на те, що мова називається універсальною, є кілька областей, де її найбільш доцільно застосовувати. C# є стандартним вибором для додатків Windows через вбудовану підтримку фреймворків .NET, який надає безліч компонентів, бібліотек класів інтерфейсу користувача та інших

ресурсів для прискорення розробки. Ця мова вважається хорошим вибором для розробки ігор, оскільки ігровий движок Unity побудований на C#. Ігри можна створювати для різних платформ, таких як пристрої Xbox, PlayStation, а також мобільні ігри для мобільних пристроїв (Android та iOS) і для ПК (Windows, Mac, Linux).

На C# розробляють якісні й надійні веб-сервіси, використовуючи ті ж ресурси, що й платформа .NET. Більше 30% розробників постійно використовують C# для своїх веб-додатків. C# має велику бібліотеку, яка може забезпечити вищий рівень функціональності, ніж інші мови. Якщо додаток працює з іншими технологіями Microsoft, це додатково дає можливості якісної інтеграції. [8]

1.3. Вибір середовища для розробки

Sublime Text і Visual Studio Code є найкращими середовищами для розробки. Sublime Text 3 – це комерційний продукт, створений Sublime HQ, а Visual Studio Code – сервіс з відкритим вихідним кодом, створений Microsoft на платформі Electron. Ці редактори працюють на Windows, Mac і Linux.

Порівняємо Sublime Text та Visual Studio Code за наступними критеріями:

- інтерфейс;
- зручність написання коду;
- управління пакетами;
- додаткові корисні функції;
- продуктивність.

1.3.1. Visual Studio Code

Visual Studio Code (далі – VS Code) дозволяє редагувати код на різних мовах програмування. Наявні кілька вбудованих мов програмування, але можна встановити та налаштувати й інші (наприклад, Python чи C#), через Extension Marketplace.

VS Code має простий та інтуїтивно зрозумілий інтерфейс, який максимально збільшує простір, наданий редактору, залишаючи достатньо місця для перегляду та доступу до всіх компонентів вашої папки або проекту. Інтерфейс користувача розділено на п’ять областей:

- редактор – основна область редагування файлів з можливістю відкривати скільки завгодно файлів поруч по вертикалі та горизонталі;
- бічна панель: є своєрідним провідником до папок і файлів;
- рядок стану – нформація про відкритий проект і файли, з якими ви працюєте;
- панель активності: розташована в крайній лівій частині, дає змогу перемикатися між сторінками з різною функціональністю;
- додаткові панелі – для відображення під областю редактора (або можна перемістити вправо, щоб збільшити простір по вертикалі) областей для виведення або відлагодження інформації, помилок і попереджень або вбудованого терміналу.

Microsoft створила в VS Code функцію під назвою Intellisense, яка є альтернативою пошуку рішень в Google.

IntelliSense аналізує семантику (букви) того, що ви вводите, і також решту створеного коду, а потім надає пропозиції щодо того, як завершити написаний код. Через Intellisense ви можете отримати доступ до цілого ряду, але різних завершень, включаючи пропозиції мовного сервера, фрагменти та текстові завершення на основі слів.

IntelliSense доступний для найпоширеніших мов програмування, включаючи HTML, CSS та Javascript, для інших мов вам потрібно буде встановити розширення.

VS Code має ряд інших корисних функцій, допомагаючи краще візуалізувати свій код і прискорити програмування. Сюди входять:

- підтримка шаблонів: це дозволяє створювати каталог невеликих частин багаторазового коду, який можна вставляти в більші частини коду;
- функція «Перейти до»: це дозволяє швидко знаходити символи, файли, рядки, значення та переходити до них;
- підсвічування синтаксису: відображає код різними кольорами та шрифтами відповідно до типу мови програмування, що використовується;
- режим “Дзен” – повноекранний режим, який дозволяє редагувати код без відволікань.

1.3.2. Sublime Text

Перша велика відмінність між Sublime Text і VS Code полягає в тому, що перший має ліцензійну плату, хоча у нього є безкоштовна пробна версія, якої цілком достатньо для забезпення усіх основних потреб під час програмування.

При відкритті Sublime Text вперше, бачимо простий текстовий редактор у вигляді одного вікна. Немає бічної панелі, немає опцій для пошуку та немає можливості перейти безпосередньо до бічної панелі розширення. Це забезпечує більш цілеспрямоване середовище для зосередження на написанні коду. У верхньому правому куті є невелика карта, яка дає змогу оглянути весь код. Більшість потрібних налаштувань знаходиться в меню вгорі, інші можна викликати через палітру команд (без спойлерів – про це нижче).

Запуск програми відбувається досить швидко в порівнянні з VS Code, і крім цього, Sublime Text займає набагато менше пам’яті, що говорить про хорошу продуктивність.

Управління пакетами в Sublime Text подібне до розширень VS Code, але воно недоступне з коробки. Для отримання додаткової функціональності, потрібно встановити Package Control. Існує багато пакетів і плагінів для підтримки різних мов програмування та підвищення ефективності і якості розробки. Наприклад, популярним є плагін Emmet, який допомагає писати HTML і CSS код швидше, використовувати скорочення, які автоматично розширюються.

Чудовим рішенням в Sublime Text є палітра команд. Вона значно спрощує доступ до функцій з меню. Крім того, працює підбір пропозицій команд та автозаповнення. Для отримання доступу до палітри команд, потрібно натиснути CTRL+SHIFT+P.

В Sublime Text є функція з організації папок і файлів у вигляді проекту. Це означає, що всі папки та файли зберігаються в одному місці та швидко доступні з бічної панелі замість того, щоб відкривати кожен окремо і вручну. Це також дає можливість швидкого пошуку у всіх файлах проекту одночасно. Аналогічно до VS Code можна працювати з кількома файлами паралельно.

Sublime Text має безліч інших, «менших» функцій, про які тут варто згадати. Це включає:

- шаблони: працюють так само, як у VS Code, але крім власних, можна встановити їх з розширень;
- меню, присвячене всім функціям “Перейти до” у цьому текстовому редакторі: є значно функціональнішою версією власного меню VS Code “Перейти”.
- Багаторазове редагування: коли ви натискаєте CTRL+D, усі екземпляри слова або команди, які ви зараз використовуєте,

будуть виділені у файлі, а також можна натиснути CTRL+F, щоб знайти та замінити слова.

У Sublime Text, на відміну від VS Code, немає функції відлагодження, що може стати проблемою для розробників. [10]

РОЗДІЛ 2
РОЗРОБКА ВЕБ-ДОДАТКА

2.1. Створення Django архітектури

Django – це бекенд фреймворк Python з відкритим вихідним кодом, що використовується для створення високорівневих веб-сайтів. Веб-розробка на Python з допомогою Django дозволяє розробникам зосередитися на створенні власних веб-сайтів, не починаючи розробку з нуля. Крім цього, така розробка швидка та проста у застосуванні, з високим рівнем захисту, підтримує будь-яку кількість проектів веб-додатків, й загалом, застосовується протягом тривалого часу. Django підходить для розробки будь-яких веб-додатків і, незалежно від складності проекту, є сумісним.

2.1.1. Створення проекту

Перед початком розробки за допомогою командного рядка переконуємось, що на комп'ютері встановлений Python, бажано найновішої версії. Інформацію по актуальних командах для перевірки та/або встановлення можна знайти на офіційному сайті Python.

Примітка. Всі згадані команди без уточнень повинні бути виконані в командному рядку – консолі.

Наступним кроком буде встановлення або оновлення до останньої версії фреймворку Django. Аналогічно, інформацію знаходимо на відповідному сайті.

Коли Python і Django встановлені, починаємо етап розробки.

Переходимо до потрібної директорії та за допомогою команди “django-admin startproject WebVisualBoard” створюємо проект. Після виконання команди в директорії з’явилась папка з заданим ім’ям – WebVisualBoard.

Переходимо до папки проекту, оглядаємо отриману структуру та файли (рис.5):

- зовнішнія коренева директорія WebVisualBoard/ – контейнер для всього проекту, її можна перейменувати, оскільки її назва не важлива для Django, але ми не будемо цього робити;
- manage.py – утиліта командного рядка, яка дозволяє взаємодіяти з проектом;
- внутрішня директорія WebVisualBoard/ – фактичний пакет Python для створеного проекту, і його ім’я потрібно буде використовувати, щоб імпортувати компоненти всередині нього;
- WebVisualBoard/__init__.py – порожній файл, який повідомляє Python, що цей каталог слід вважати пакетом Python;
- WebVisualBoard/settings.py – налаштування/конфігурація для цього проекту Django;
- WebVisualBoard/urls.py – оголошення URL для проекту, так званий зміст сайту на базі Django;
- WebVisualBoard/asgi.py – точка входу для ASGI-сумісних веб-серверів для обслуговування проекту при розміщенні на хостингу;
- WebVisualBoard/wsgi. py – точка входу для WSGI-сумісних веб-серверів для обслуговування проекту при розміщенні на хостингу.

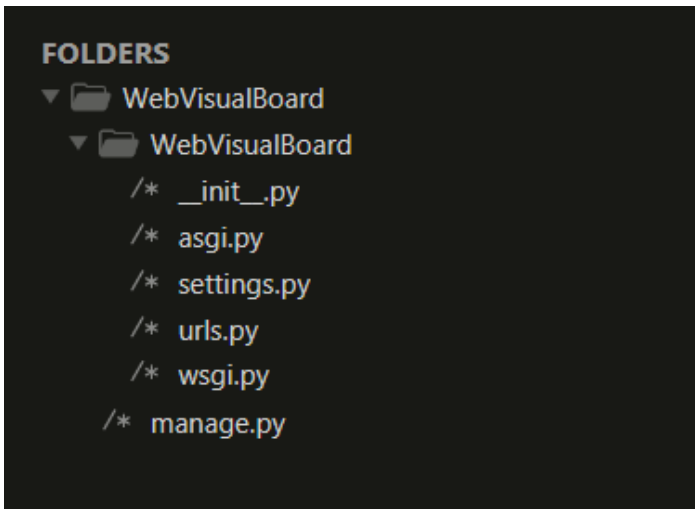

Рис.5. Початкова структура проекту

2.1.2. Створення додатків

Для подальшої розробки потрібно створити ще одні структурні одиниці – додатки, в файлах яких буде прописуватись вигляд і логіка роботи веб-додатка. Обов’язково проект повинен містити хоча б один додаток.

Тому в консолі переходимо в зовнішню директорію проекту – WebVisualBoard (надалі всі операції будуть проводитись в цій директорії), та створюємо наш основний додаток main за допомогою команди “python manage.py startapp main”.

Далі нам потрібно до нашого додатка main додати файл urls.py, в якому ми будемо задавати шляхи до різних сторінок. Структура проекту тепер набула наступного вигляду (рис.6):

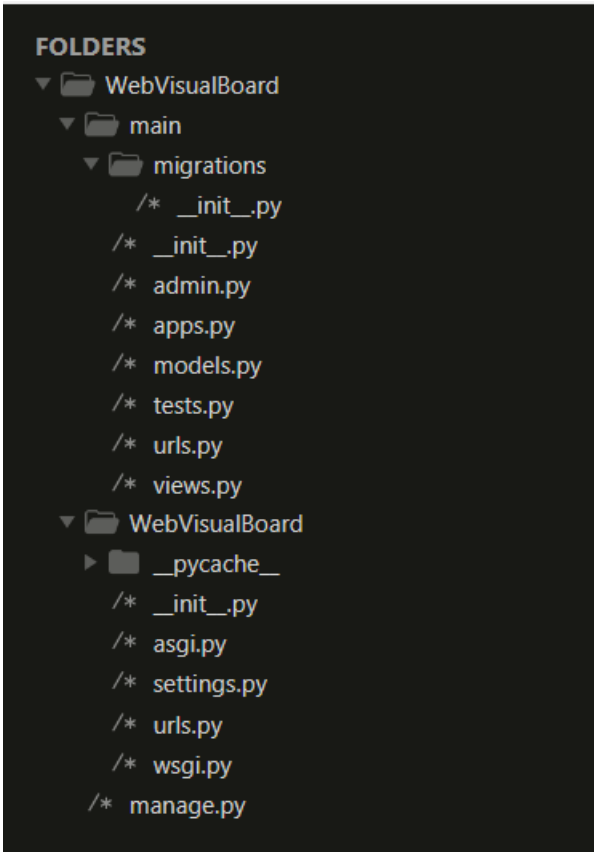

Рис.6. Структура проекту після створення додатка main

До наявних файлів додалось ще декілька, розглянемо основні, які будуть використовуватись в процесі розробки нашого додатка:

- WebVisualBoard/main/ – директорія створеного додатка main, де зберігатимуться усі файли;
- WebVisualBoard/main/admin.py – конфігурація панелі адміністратора;
- WebVisualBoard/main/apps.py – конфігурація додатка;
- WebVisualBoard/main/urls.py – оголошення URL в межах додатка main;
- WebVisualBoard/main/views.py – опис логіки в додатка;
- WebVisualBoard/main/models.py – створення і конфігурація моделей бази даних;
- WebVisualBoard/main/migrations/ – директорія для відображення міграцій моделей.

2.1.3. Взаємодія всередині проекту

При будь-яких діях на сторінках веб-додатка відбувається постійне “спілкування” між клієнтом (браузером) та сервером – відправляємо HTTP-запити від клієнта, ініціюємо та отримуємо HTTP-відповіді від сервера.

Розглянемо основні принципи взаємодії компонентів в проектах, розроблених на Django (рис.7).

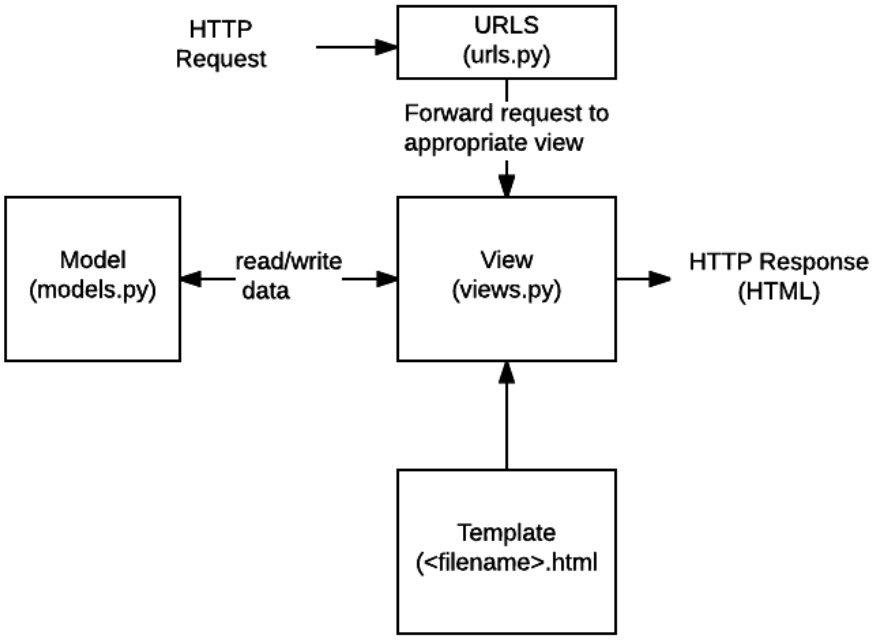

Рис.7. Схема взаємодії в проекті Django

Бачимо, що HTTP-запит від клієнта спочатку звертається до файлу urls.py, який в свою чергу скеровує його до urls.py відповідного додатка (main). Наступні переходи здійснюватимуться вже в межах цього додатка. Наступним відбувається звернення до views.py (відображення якого задане в відповідному темплейті <filename>.html, який розглянемо пізніше), з допомогою якого потрібний елемент зі сторінки знаходимо в models.py. На цьому етапі проводяться необхідні дії з базою даних (зчитання/редагування даних). Тоді відбувається повернення до views.py, і нарешті HTTP-відповідь на наш запит, яку ми отримуємо в браузері.

2.2. Робота з даними

Кожен додаток працює з певним набором інформації, певними відомостями про об’єкти та користувачів. Для ефективної роботи та управління цими відомостями в проекти вбудовують бази даних.

База даних – це система, головною функцією якої є збір та впорядкування інформації.

Бази даних можуть бути сформовані у вигляді звичайних списків в коді. Проте це робочий варіант тільки для крихітних програм, а для більших проектів вони створюються на основі систем управління базами даних (СУБД).

За замовчуванням для конфігурації баз даних в проектах Django використовується SQLite – спрощена реляційна система управління базами даних, представлена у вигляді бібліотеки. SQLite входить до пакету Python, тому не потрібно встановлювати додаткових інструментів для підтримки бази даних, створеної на її основі.

Перед початком роботи з даними і, загалом, всіма файлами в директорії main/ підключимо наш додаток до проекту – до INSTALLED_APPS у файлі WebVisualBoard/settings.py додаємо рядок із значенням 'main.apps.MainConfig'

Тепер додаток зареєстрований і Django має до нього доступ.

2.2.1. Моделі і міграції

Формуємо структуру бази даних для нашого проекту (рис.8) з урахуванням об’єктів, які створюватиме користувач під час роботи.

Тепер перейдемо до створення та налаштування моделей в WebVisualBoard/main/models.py, який і є фактично макетом бази даних з визначеними параметрами.

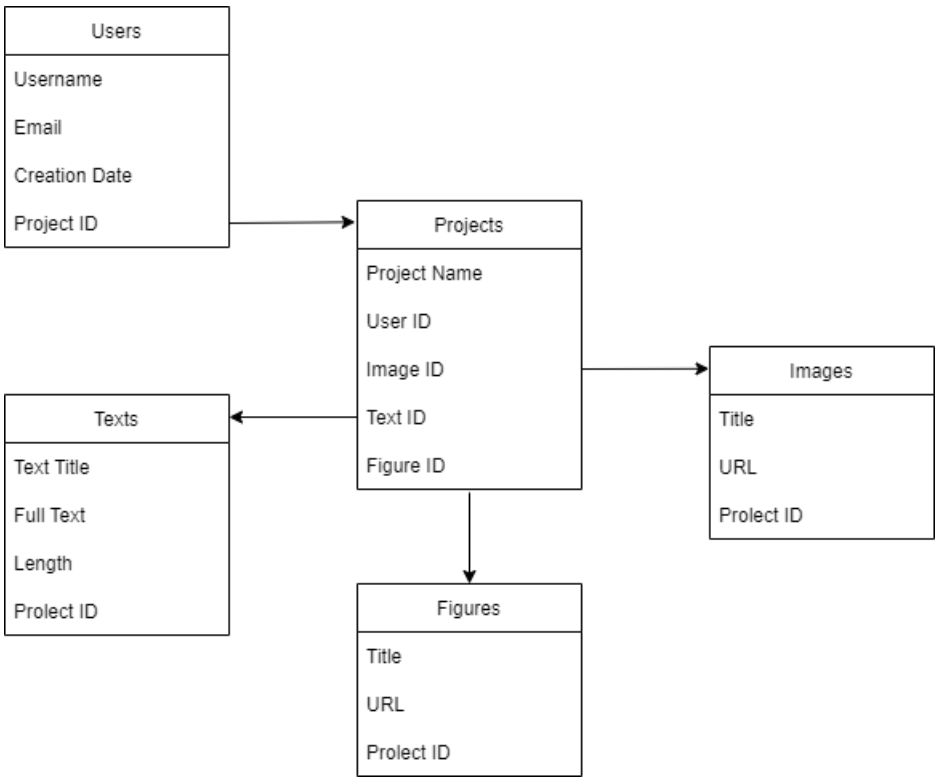

Рис.8. Схема зв’язку інформації в базі даних

Перша обов'язкова таблиця, яку потрібно створити, це Users, де зберігатиметься основна інформація про користувачів. Users матиме такі поля: Username, Email, Creation Date, Project ID.

Для цього у нашому файлі з django.db імпортуємо models і створюємо клас User(models.Model). Кожен клас в моделях – це окрема таблиця, яку пізніше додамо в базу.

Задаємо поля для таблиці User з визначенням їх властивостей (обов’язково вказуємо тип даних, при потребі додаткові параметри – для прикладу, максимальна довжина для поля username):

```
username = models.CharField('Username', max_length = 100)
email = models.EmailField('Email', max_length = 100)
creation_date = models.DateTimeField('Creation Date', auto_now_add = True)
project_id = models.IntegerField('Project ID')
```

Аналогічно створюємо класи Project, Image, Text, Figure з відповідними полями.

Наступним кроком буде активація створених моделей – міграції. В консолі прописуємо команду “python manage.py makemigrations main”, і після цього – “python manage.py migrate”.

Першою командою ми повідомляємо Django про внесені зміни у свої моделі (в даному випадку про створення моделей) і що їх потрібно зберегти для міграцій.

Міграції – це файли на диску з інформацією про зміни в моделях і, відповідно, у базі даних. Файли міграцій для наших моделей зберігатимуться в директорії main/migrations і виглядатимуть наступним чином: {0001}_initial.py. Вони створені для редагування вручну.

Міграції дозволяють змінювати моделі з часом без необхідності видаляти базу даних або таблиці і створювати нові – вони оновлюють бази даних без втрати інформації.

2.2.2. Панель адміністратора

Панель адміністратора – це сервіс для управління налаштуваннями веб-сайтів. З її допомогою можна додавати та видаляти сторінки, змінювати зовнішній вигляд і редагувати контент.

В Django є готова панель адміністратора, якою одразу можна користуватися, перейшовши за URL-шляхом “/admin”. Бачимо просту форму авторизації (рис.9).

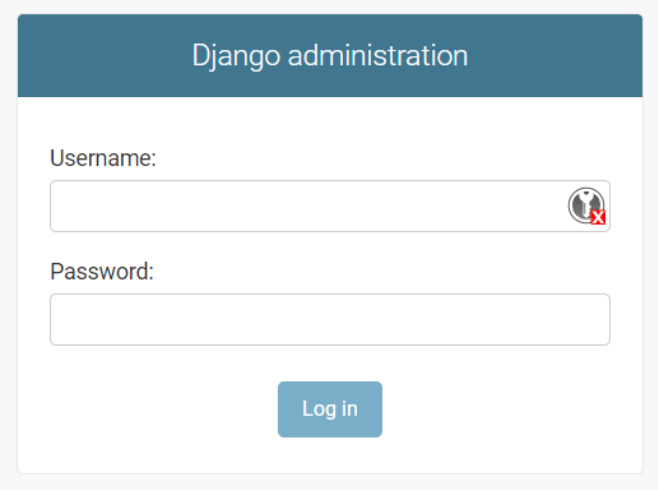

Рис.9. Форма авторизації для входу до панелі адміністратора

Оскільки в нас немає користувача, потрібно його створити. Для цього в консолі прописуємо команду “python manage.py createsuperuser”, вказуємо ім’я користувача і створюємо пароль. Далі з даними створеного користувача авторизуємось в панелі адміністратора.

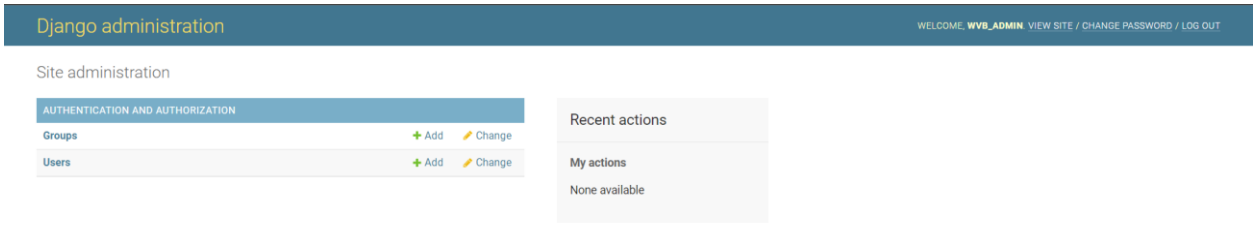

Рис.10. Вигляд кабінету адміністратора Django за замовчуванням

Потрапляємо до кабінету користувача (власне, адміністратора)(рис.10). Поки що з наявних даних бачимо лише розділ “Authentication and authorization”, який створюється за замовчуванням у панелі адміністратора Django.

Тепер нам потрібно адаптувати панель адміністратора під свій проект, щоб надалі через неї керувати даними без прямого звернення до бази.

Переходимо до файлу WebVisualBoard/main/admin.py. Імпортуємо з файлу models.py модель User, прописавши “from .models import User” і за

допомогою функції admin.site.register(User) реєструємо її в панелі користувача. Аналогічно до User реєструємо модель Project. Після цього в панелі адміністратора з'являється розділ з інформацією додатка main – нашими зареєстрованими моделями (рис.11). Тепер ми можемо додавати, редагувати та видаляти дані в цих таблицях через панель адміністратора. Якщо виникне потреба в зміні чи додаванні нових елементів в структуру таблиць, необхідно повторити дії з моделями та міграціями.

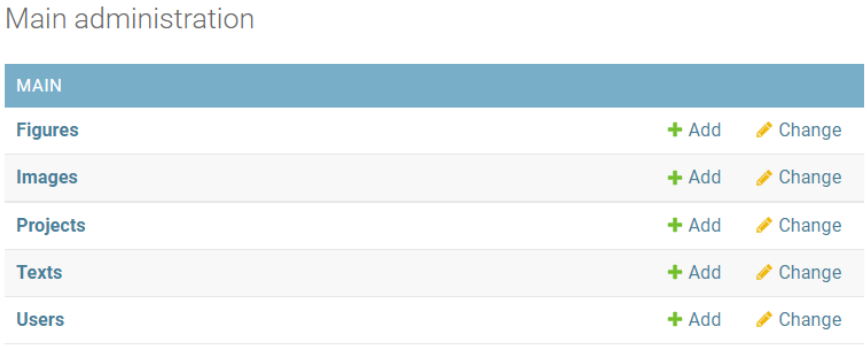

Рис.11. Відображення таблиць в панелі адміністратора

2.3. Розробка фронтенд частини

Фронтенд – це візуальна складова будь-якого веб-додатка, де вимагається взаємодія з користувачем. Звичайно, ця взаємодія може відбуватися і через консоль. Але в такому випадку це є не ефективно і абсолютно не зручно. Тому виникає потреба в розробці фронтенду. Це практично все, що бачить користувач, включаючи операції на стороні клієнта (браузера).

При розробці фронтенду важливим є UI/UX дизайн. Ці два види дизайну тісно взаємодіють між собою, проте мають відмінності в меті застосування і впливі на користувача.

UI (User Interface) – це візуальний дизайн інтерфейсу продукту, система

макетів, значків, кнопок, інформаційних структур, кольорів, типографіки, анімації та ілюстрації сторінок, вкладок і панелей, до яких переходить або з якими стикається (теоретично чи фактично) користувач в процесі роботи. Головне завдання UI – забезпечити користувача приємним враженням від візуального вигляду додатка.

UX (User Experience) – це складова дизайну, яка буквально відповідає за досвід користувача від користування веб-додатком. Він забезпечує основну структуру, рішення по розміщенню елементів. На етапі відображення створеного графічного веб-редактора розглянемо UX і його вплив детальніше.

2.3.1. Налаштування URL переходів

На цьому етапі нам потрібно задати логіку переходів по URL в межах нашого веб-додатку. Будемо керуватись схемою взаємодій файлів в проекті Django (рис.7).

Найперше, в файлі WebVisualBoard/WebVisualBoard/urls.py імпортуємо path, include – “from django.urls import path, include”, і в urlpatterns додаємо шляхи до всіх майбутніх сторінок (пізніше це можна буде змінити):

```
path(' ', include('main.urls'))
path('image_editor', include('main.urls'))
path('text_editor', include('main.urls'))
path('drawer', include('main.urls'))
path('login', include('main.urls'))
path('my_account', include('main.urls'))
path('user_guides', include('main.urls'))
```

Далі в WebVisualBoard/main/urls.py імпортуємо views і майже аналогічно до попереднього кроку вказуємо всі ці шляхи в urlpatterns; ось приклад для головної сторінки:

```
path(' ', views.index, name='index')
```

Крім значення в адресному рядку, вказується до якої саме функції у views.py ми звертаємось.

Тепер переходимо до WebVisualBoard/main/views.py і створюємо функції, відповідно до заданих нами URL посилань в urls.py. Ці функції визначатимуть реакції додатку при переході по цій URL. Наприклад, для переходу до головної сторінки це буде функція def index(request) з методом render, що покаже конкретну HTML-сторінку.

Ось ми і побудували зв’язки між URL, і далі перейдемо до проектування інтерфейсу.

2.3.2. Створення інтерфейсу

За стандартом структура для усіх веб-сторінок будується і контент відображається в браузері за допомогою HTML коду.

HTML – це мова розмітки гіпертексту, що означає наявність зв'язків з іншими документами, до яких можна перейти з вихідного тексту за внутрішніми посиланнями.

Створимо директорію, де зберігатимуться макети сторінок. Спочатку нам буде достатньо одного файлу, а пізніше зможемо доповнити директорію усіма потрібними:

1. Переходимо до WebVisualBoard/main/
2. Створюємо нову директорію “templates”
3. В створеній директорії створюємо дочірню з назвою, аналогічною до назви нашого додатку – main.
4. В директорії WebVisualBoard/main/templates/main/ створюємо файл base.html
5. В тій же директорії створюємо інші HTML файли: index.html; image_editor.html; text_editor.html; drawer.html; login.html; my_account.html; user_guides.html.

Для написання HTML коду використовуються теги та атрибути. Тег – це назва елементу сторінки, який визначає тип контенту, розміщеного в ньому. Записується у кутових дужках (“<” та “>”). Кожен елемент має початковий та кінцевий тег (але для деяких елементів не обов'язково використовувати кінцевий тег).

Кожен HTML тег має унікальну назву та визначений синтаксис, що записується латинськими літерами і не є чутливим до регістру. [11]

Для веб-сторінок нашого додатку використовуємо багато елементів різних типів. Основні теги, які задаємо в HTML коді:

<body> – тіло веб-сторінки;

<button> – кнопка;

 – блок;

<footer> – футер веб-сторінки;

<form> – форма;

<head> – інформація про документ;

<header> – хедер веб-сторінки;

<html> – початок HTML-документа;

<img> – графічне зображення;

<input> – поле вводу даних;

<meta> – метадані документа;

<p> – текстовий абзац;

<script> – підключення JavaScript скрипта;

<style> – стиль елементів веб-сторінки;

<title> – заголовок веб-сторінки.

Атрибути – це параметри, які визначаються всередині кожного тега. Одними із загальних атрибутів, які можуть використовуватись з усіма тегами, є наступні:

title – додаткова текстова підказка;

class – один або декілька класів для зв’язку елемента з таблицею стилів CSS;

hidden – прихований від перегляду вміст елемента;

id – унікальний ідентифікатор елемента;

style – вбудований CSS стиль для елемента.

Використовуючи ці основні й інші додаткові теги та атрибути, будуємо структуру головної сторінки за таким макетом (рис.12):

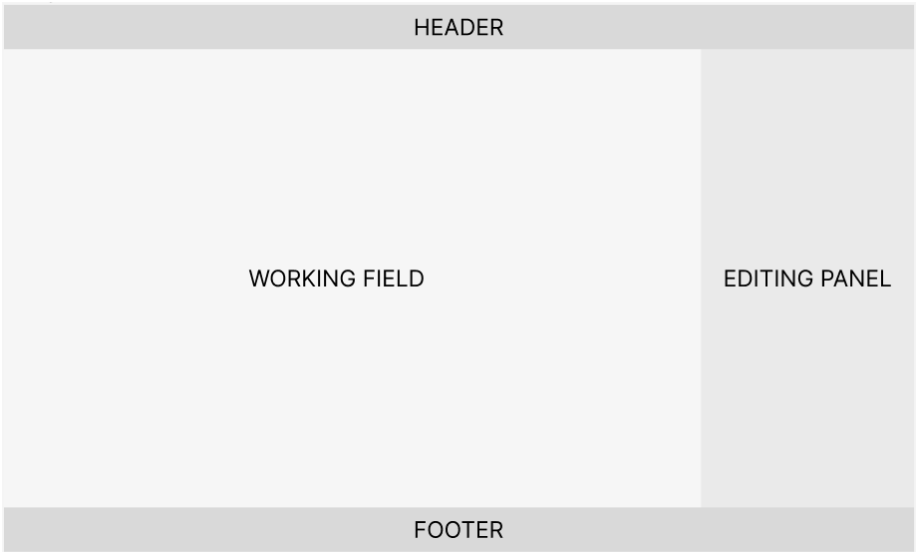

Рис.12. Макет головної сторінки графічного редактора

Наповнюємо base.html контентом, цей файл буде нашим HTML шаблоном. Переходимо до наступного етапу. Нам потрібно визначити частини і елементи, які будуть однаковими для всіх сторінок, а які будуть відрізнятися. Робимо це для того, щоб з допомогою шаблонізатора Jinja усунути повторення однакового коду в різних файлах.

Однаковими буде хедер, центральна частина та футер. А от наповнення цих структур буде відрізнятися. Тому все, крім них, задаємо в окремих HTML файлах, створених раніше під конкретні URL.

В файли підключаємо наш шаблон base.html. Для цього в них зверху прописуємо “{% extends 'main/base.html' %}”, це означатиме наслідування

коду з шаблонного файлу. Тепер для кожного елементу/блоку елементів, що підлягають шаблонізації, в base.html прописуємо структуру такого типу: {% block right-panel %}{% endblock %}. Тоді у всіх інших файлах задаємо контент, який підтягуватиметься для кожного з них. Щоб це прописати, використовуємо ідентичну структуру (примітка: “right-panel” – назва, яка є унікальною для кожного із елементів, і відповідно, змінною в даній структурі) в потрібних місцях файлів із заданим унікальним HTML кодом між {% block right-panel %} та {% endblock %}.

Таким чином розробляємо інтерфейс на всіх сторінках: створюємо форми логіну та реєстрації, акаунт користувача з розділом “My projects”, сторінку з інструкцією з користування функціями графічного редактора, панелі для різних операцій редагування.

2.3.3. Підключення статичних файлів

Окрім HTML, веб-додатки повинні обробляти додаткові файли, такі як зображення, JavaScript або CSS, необхідні для візуальної наповненості та функціональної працездатності веб-сторінок. У Django такі файли називаються статичними.

CSS (каскадні таблиці стилів) – спеціальна мова, яку використовують для задання оформлення веб-сторінок, написаних мовою HTML. Концепція працює наступним чином: спершу текст виводиться, а потім форматується за допомогою стилів CSS. Стилі допомагають відділити вміст веб-сторінки від оформлення.

JavaScript як мову програмування ми розглядали у Розділі 1. В нашому проекті застосуємо її для динамічності сторінок. Код знаходитиметься в окремому файлі .js формату. Посилання на конкретні методи та функції будуть знаходитись в тегах елементів HTML сторінки.

В маленьких проектах питання зі статичними файлами не викликає багато проблем, тому що їх можна зберігати в тому місці, де їх може знайти наш веб-сервер. Проте у великих проектах робота з кількома наборами таких файлів ускладнюється, можуть виникати конфлікти.

Для вирішення цієї проблеми призначений django.contrib.staticfiles: він збирає статичні файли з кожного додатку (чи інших місць, які вказуємо для розміщення файлів) в єдине місце, яке можна легко обслуговувати при виконанні. [12]

В нашому проекті тільки один додаток main, але це не виключає потреби у правильному зберіганні статичних файлів.

Переходимо в WebVisualBoard/settings.py і задаємо три параметри:

STATIC_URL = 'static/' – префікс URL адреси для статичних файлів;

STATIC_ROOT = "/static/" – шлях до загальної директорії зі статичними файлами при використанні на реальному веб-сервері;

STATICFILES_DIRS = [ ] – список додаткових/нестандартних шляхів, що використовуються у режимі налагодження.

Далі в директорії нашого додатку main створюємо папку static/, і в ній ще одну дочірню main/, щоб уникнути конфіктів у випадку розширення проекту до кількох додатків. В новоствореній папці main/ ми й будемо зберігати всі статичні файли або папки з файлами. Для зберігання зображень, використовуваних на наших сторінках, додаємо таку папку – “images/”, і завантажуємо всі зображення туди. Аналогічно створюємо папки “css/” та “js/” для зберігання відповідно .css та .js файлів.

Тепер підключаємо наші файли безпосередньо в HTML:

1. зверху в файлі base.html прописуємо {% load static %};
2. в тезі <head> підключаємо CSS: <link rel="stylesheet" href="{% static 'main/css/style.css' %}" type="text/css" />;

3. в тому ж тезі підключаємо JavaScript: <script src="{% static 'main/js/myscript.js' %}" type="text/javascript"></script>;
4. для кожного локально збереженого зображення шлях вказуємо в такому вигляді: href="{% static 'main/images/logo.jpg' %}".

За допомогою команди “python manage.py check” перевіряємо проект на наявність помилок (рис.13).

```
D:\Python\diploma project\WebVisualBoard>python manage.py check
System check identified no issues (0 silenced).
```

Рис.13 . Перевірка проекту на наявність помилок

РОЗДІЛ 3
ДЕМОНСТРАЦІЯ СТВОРЕНОГО ВЕБ-ДОДАТКА

3.1. Виконання додатка

Після основних стадій розробки графічний веб-редактор Web Visual Board (WebVB) буде готовий до виконання своїх функцій. Для всіх користувачів ці функції є загально доступними, але порядок виконання або їх невиконання відрізняються для кожного залежно від шляху, який проходить користувач (рис.14).

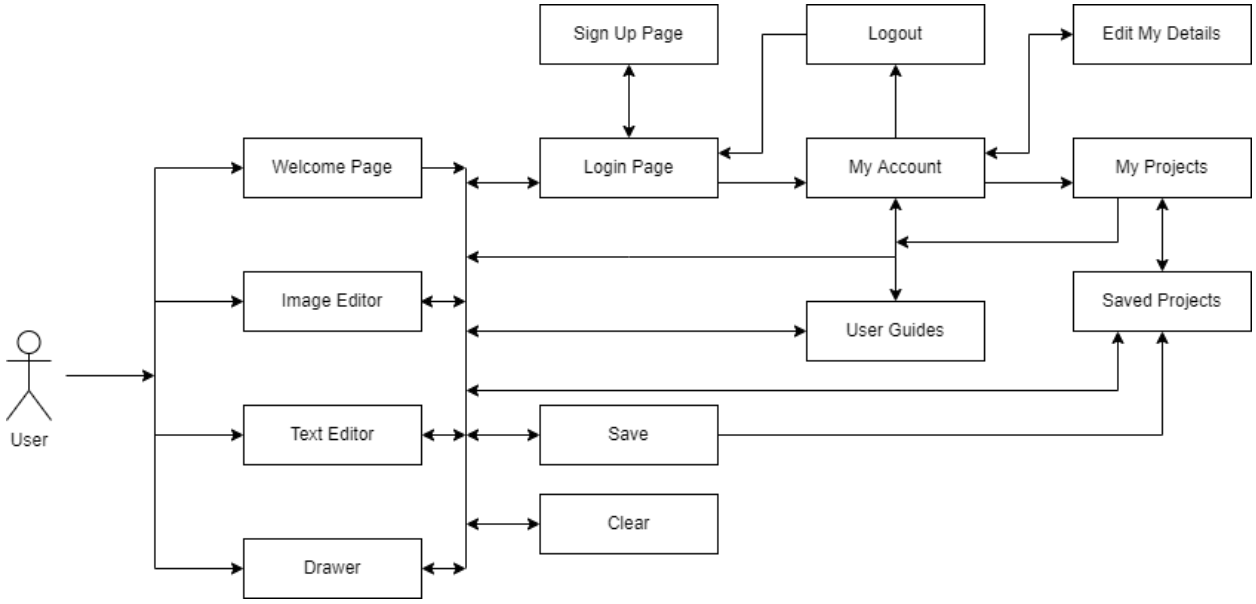

Рис.14. Шлях користувача

Розглянемо один із варіантів шляху користувача, який він може пройти під час користування нашим веб-додатком.

3.1.1. Головна сторінка

Коли користувач заходить на веб-сайт вперше або коли він почистить дані в браузері (кеш та куки – файли, які зберігають інформацію про відвідування користувачем сторінки, щоб прискорити завантаження даних в майбутньому), йому показується головна сторінка – графічний веб-редактор Web Visual Board (рис.15).

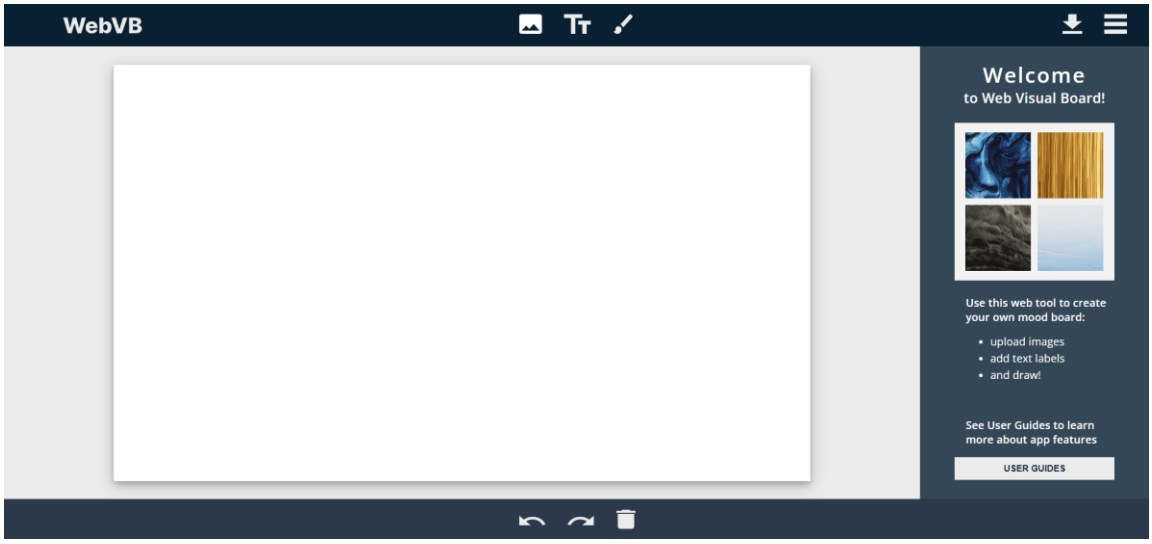

Рис.15. Головна сторінка графічного редактора Web Visual Board

На головній сторінці знаходиться хедер (верхня панель), основна (центральна) частина та футер (нижня панель). У хедері розташований логотип, інструменти для створення роботи (завантаження зображень, додавання тексту, малювання), кнопка збереження та випадаюче навігаційне меню. У футері знаходяться кнопки для таких операцій як: скасування дії, крок на дію вперед, видалення виділеного об’єкта.

Найцікавішою частиною є центральна. Ця область поділена ще на дві. Перша – робоче поле, на якому і відбуватиметься контактна взаємодія з зображеннями, текстом та малюнками, друга – панель, на якій розташована основна інформація про наш графічний редактор. Знизу знаходиться CTA (“call to action” – заклик до дії) кнопка, яка веде нас до юзер-гайдів – інструкцій

по користуванню. Переходячи за посиланнями відповідно до потрібної функції додатку, можна знайти інформацію по роботі з нею.

Тепер авторизуємось. Для цього у випадаючому списку обираємо пункт “My account”, який перенаправить на форму для входу.

3.1.2. Авторизація та кабінет користувача

На формі авторизації (рис.16) бачимо ще одну варіацію логотипа, поля для вводу імені користувача та пароля, пункт для запам’ятовування даних про вхід, і нарешті, CTA кнопка для входу. Після натисканні кнопки додаток відправить введені дані для пошуку в базі, і якщо такий користувач існує, авторизація пройде успішно. У випадку, якщо акаунта немає, можна зареєструватися за допомогою посилання під кнопкою.

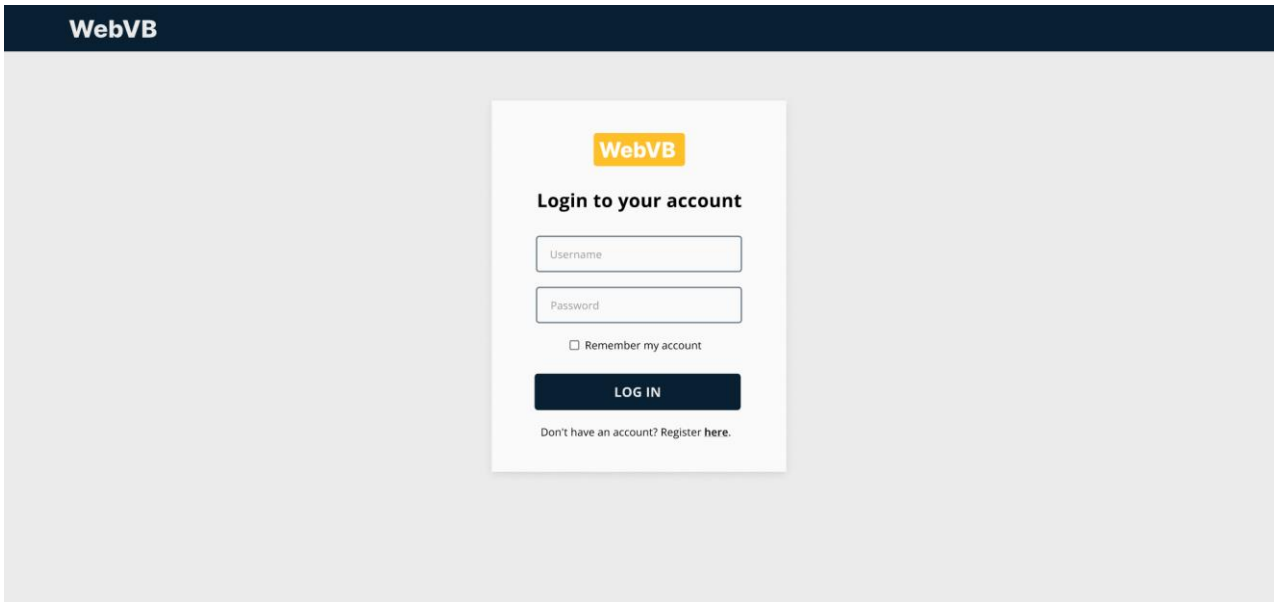

Рис.16. Форма авторизації в акаунт користувача

Після успішної авторизації користувач потрапить до свого кабінету (рис.17). Тут можна побачити фотографію користувача (аватар), його ім’я, контактні дані (електронна пошта, телефон, місце проживання), та так звана позиція в творчому суспільстві, яку користувач сам вказує при реєстрації (для

прикладу “meme creator” – той, хто створює меми – іронічні зображення, що мають зв’язок з історичними подіями чи повсякденним життям людей). Крім цього, є можливість відредагувати дані профілю та змінити аватар.

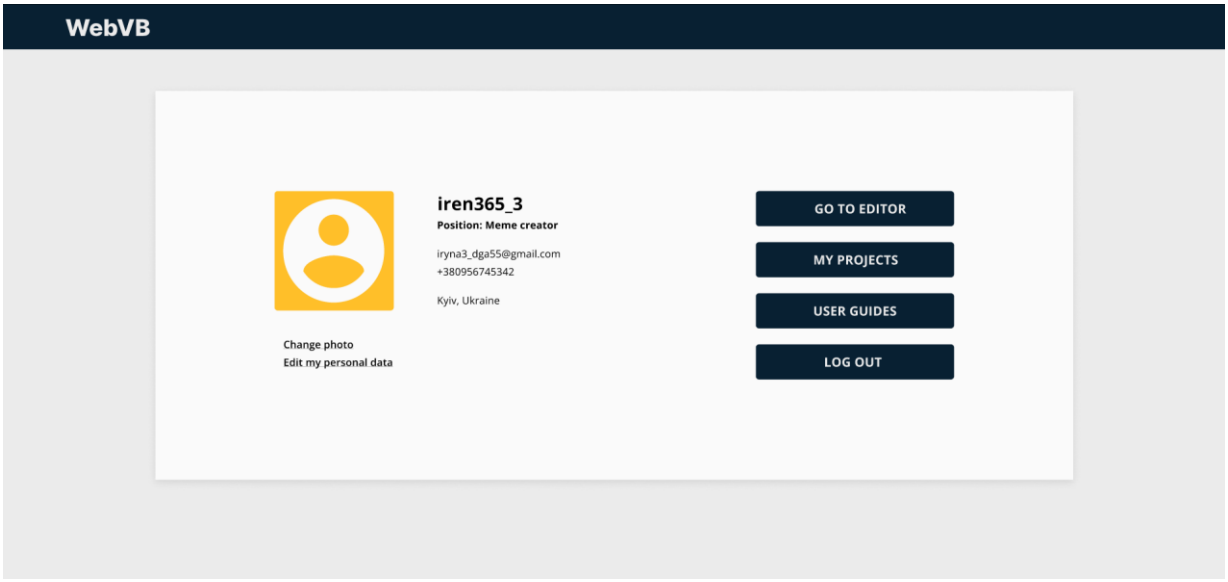

Рис.17. Кабінет користувача графічного додатка Web Visual Board

Справа в кабінеті користувача бачимо кілька CTA кнопок, які перенаправлять нас відповідно до свого призначення:

- “Go to editor” відкриє графічний редактор з поточним проектом;
- “My projects” перенаправить до сторінки зі збереженими проектами, якщо такі є, або запропонує створити проект, якщо в користувача немає жодного;
- “User guides” відкриє сторінку з інструкціями по користуванню;
- “Log out” виведе користувача з його акаунта та перенаправить на форму авторизації.

Натискаємо кнопку “Go to editor” і переходимо до графічного редактора.

3.1.3. Імпорт зображень і їх редагування

Основна функція додатка – робота з зображеннями. Для того, щоб почати, необхідно імпортувати файл у форматі .jpg, .jpeg або .png, скориставшись відповідною кнопкою на верхній панелі.

Після натискання на кнопку імпорту панель справа зміниться з інформаційної на робочу – для редагування зображень (рис.18). Тут ми можемо побачити різні інструменти для вирівнювання, зміни розміру, обрізання, переміщення, повороту, дублювання, редагування контрасності тощо. Дії застосовуватимуться тільки до завантажених і виділених зображень.

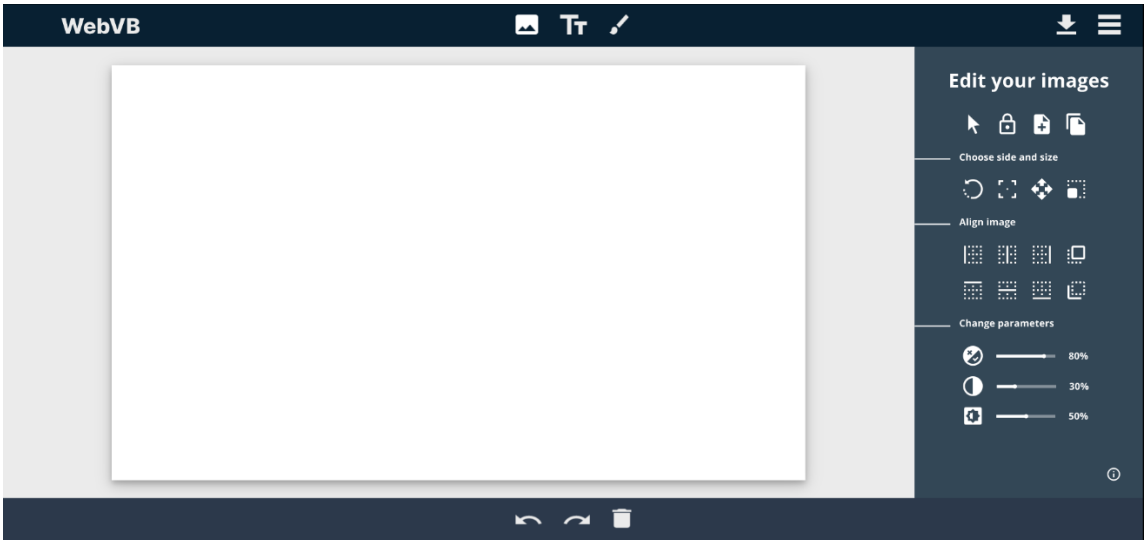

Рис.18. Панель редагування зображень

3.1.4. Додавання тексту і малюнків

Схоже до функції роботи з зображеннями працюватиме додавання тексту. Але у цьому випадку замість створення об’єктів-зображень створюватимуться текстові поля. Для цього призначена кнопка на верхній панелі у вигляді літер “Тт”. Бічна робоча панель зміниться на відповідну для редагування тексту.

Функція малювання дещо відрізняється від двох попередніх (рис.19). Для створених малюнків не буде жодних можливостей редагування, оскільки графіка є растровою. Серед опцій малювання – вибір кольору, товщини лінії та гумка, для стирання лишнього (лише для намальованих елементів). Найцікавішим є вибір кольору – реалізований у вигляді віджета, де можна обрати кольоровий спектр, і вже тоді відтінок в його межах. Можна вручну задати HEX, RGB, HSV код для обрання конкретного потрібного відтінку в палітрі.

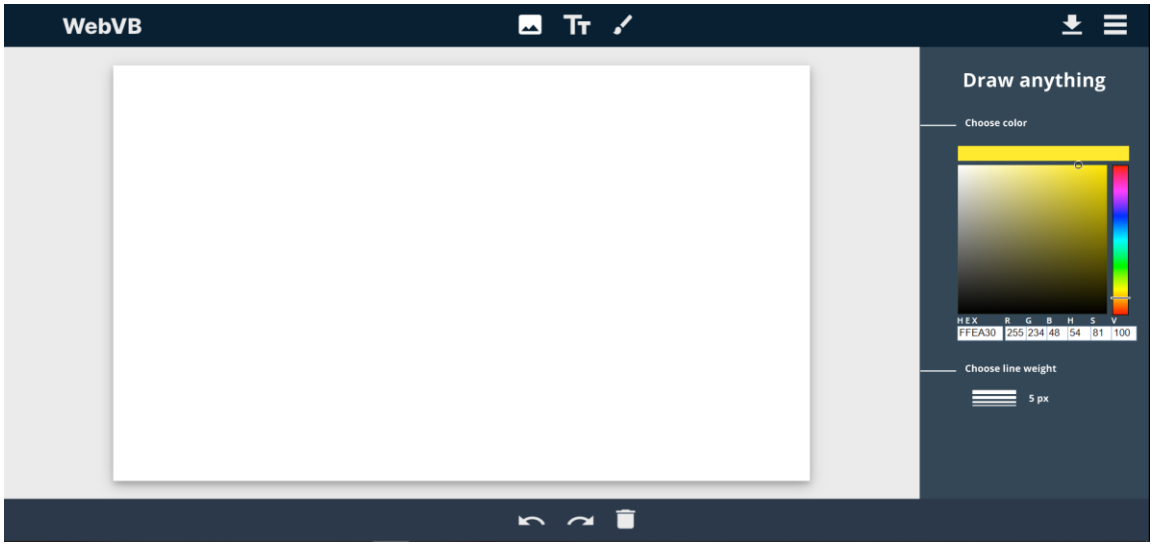

Рис.19. Панель з налаштуваннями для малювання

3.1.5. Збереження проекту

Завершальним етапом роботи буде збереження проекту. Відповідна кнопка також знаходиться на верхній панелі, поряд з меню. Доступними форматами для збереження є стандартні для зображень (.jpg, .jpeg, .png).

Якщо проект створюватиметься авторизованим користувачем, то при збереженні на комп’ютер, він автоматично збережеться і в розділ “My projects” в кабінеті користувача.

3.2. Принципи UX дизайну

Перед проектуванням інтерфейсу ми розглядали UI/UX дизайн. На відміну від UI, UX створює не візуально приємну сторону сервісу, а комфортну з точки зору функціональності.

Хоч UX дизайн є досить мінливою сферою і часто здатною приймати цікаві творчі рішення, але існує кілька основних принципів, які варто пам’ятати і дотримуватись.

Тепер розглянемо ці основні принципи і проаналізуємо, наскільки нам вдалось реалізувати вимоги з точки зору саме UX.

Задоволення потреб користувача. Головним із принципів UX є зосередження уваги на користувачеві протягом усього процесу проектування. Робота повинна мати за ціль покращення досвіду користувача з нашим продуктом. Не завжди та поведінка користувачів, яку передбачаємо ми під час розробки, збігається з тією, яку здійснюють користувачі фактично (рис.20).

Під час проектування зміни в веб-додатку постійно відслідковувались і адаптувалися під користувача.

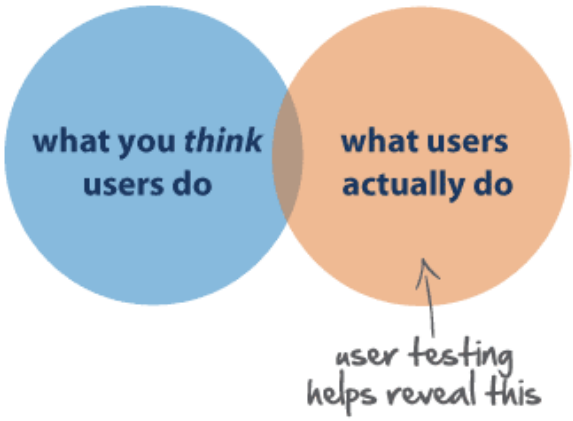

Рис.20. Зосередження на користувачеві як основна ідея UX

Поетапне проектування. Важливо розуміти, на якому етапі які речі є найважливіші. Хоч зручність використання певних елементів (кнопок,

посилань і т.д.) є важливою, але це не є першочерговим завданням. Тому ми спочатку розробили зручну структуру сторінки, а тоді наповнювали її контентом, зокрема, меншими структурними елементами (рис.21).

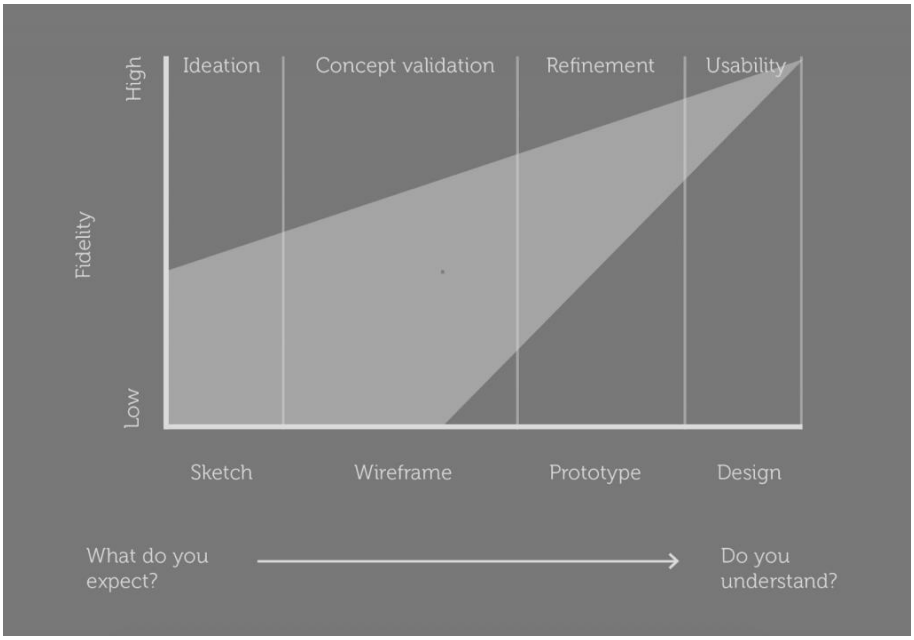

Рис.21. Зміна пріоритетів в процесі розробки

Чітка ієрархія. Ієрархію легко сприймати як належне в кохному проекті, але це є принципом UX, що забезпечує логічну і плавну навігацію по всьому дизайну. Спочатку йде ієрархія, яка пов’язана з тим, як вміст або інформація організовані в рамках дизайну (наприклад, панелі та меню).

Існує також візуальна ієрархія, яка дозволяє користувачам легко переміщатися по контенту в межах сторінки або розділу. Найкраще характеризує візуальну ієрархію величина структур і елементів, їх розташування та інтервали між ними. Завдяки цьому користувач бачить, що є головним, а що другорядним.

Наслідуваність. Звичайно, що кожен проект мусить мати свою унікальність, але користувачі мають певні очікування щодо поведінки тих чи інших додатків в порівнянні з іншими, які вони регулярно використовують. Це полегшує їм ознайомлення з новим продуктом і, безумовно, покращує їх

досвід. Тому при розробці ми орієнтувалися на існуючі графічні редактори, розглянуті в Розділі 1.

Доступність. Дедалі важливішим правилом UX-дизайну врахування доступності. Тобто, що доступнішим буде продукт, тим більшою кількістю людей він буде використовуватись. З цією метою ми застосовували контрастні кольори, великі зрозумілі іконки.

Зручність використання. Весь процес розробки за принципами UX зводиться до одного – аби користувачеві було зручно. Для цього використовуємо інтуїтивно зрозумілі користувачеві іконки, обираємо розташування кнопок, зображуємо їх так, щоб закликати користувача її натиснути (CTA кнопки). Завдяки цьому ми полегшуємо навігацію та збільшуємо взаємодію користувача з нашим продуктом. Для цього теж є дуже важливим проводити постійне тестування додатку (рис.22).

Менше означає більше. В UX мета цього принципу проста: зниження операційних і когнітивних витрат користувачів. Зважаючи на це, покращується зручність та узгодженість дизайну. Підхід «менше – більше» підкреслює простоту на відміну від безладу або надмірного декорування, перевантаження елементами в дизайні.

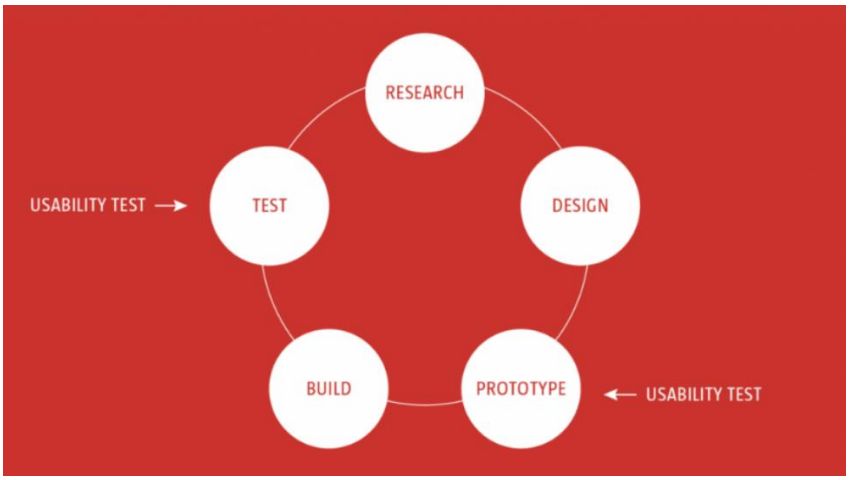

Рис.22. Тестування в циклі розробки

В нашому веб-додатку ми дотримуємось цього, прибравши весь лишній функціонал, і уникнувши емоційного навантаження на користувача.

Типографіка. Не менш важливою є створення правильної типографіки. Це може суттєво впливати покращити доступність і зробити дизайн більш зручним для користувачів, якщо задати типографічну ієрархію. Крім цього, текст повинен бути читабельним і знайомим користувачам. Керуючись цим, ми підібрали необхідний шрифт і розміри текстових елементів і блоків.

Зворотній зв'язок. Дизайн повинен бути інтерактивним. Користувачеві важливо розуміти, що його команда отримана, коли він якось взаємодіє з продуктом. В нашому додатку ми дотримуємось цього, змінюючи колір, розмір, розташування елементів при наведенні на них, а також використовуємо текстові підказки, що допомагає зрозуміти користувачеві призначення цих елементів.

Контроль користувача. Цей принцип зосереджений на більшій гнучкості використання та кращому контролі того, що здійснює користувач. Прикладом застосування цього у створеному веб-редакторі є кнопки “Undo” (скасувати) та “Redo” (відновити, або крок вперед), завдяки чому користувач зможе керувати своїми діями і не потребує переробляти проект у випадку помилки.

Візуальна граматика. Граматика у відображенні інтерфейсу продукту відіграє важливу роль у досвіді користувачів. Вона складається з усього, що становить візуальні елементи дизайну: значки, ілюстрації, кнопки тощо. [13]

В нашому веб-додатку для створення візуальної граматики застосовується структура блоків контенту, побудова правильної ієрархії, дотримання інтервалів між елементами і блоками елементів та їх вирівнювання.

ВИСНОВКИ

Візуалізація – важливий етап розробки дизайн-проектів. На цій стадії створюються стильові рішення, поєднуються готові об’єкти для подальшої їх заміни чи адаптації. Таким чином формується мудборд (mood board), що буквально задає “настрій” проекту. Така робота має виконуватись швидко, елементи повинні бути динамічні для майбутніх виправлень.

Оглянувши графічні редактори різного типу та проаналізувавши їхні відмінності, можна стверджувати, що найкращими варіантами для створення складних професійних проектів (розробка інтерфейсів, проектування) серед наявних графічних редакторів є Photoshop як десктопна версія та Figma як веб-рішення, для новачків же краще підійде стандартний Paint (для вивчення основ дизайну) та Canva (для створення своїх робіт на основі шаблонів). Однак, призначення цих редакторів зовсім різні, і для створення візуалізацій є не надто зручними.

На основі проведеного аналізу визначили основні вимоги до власного графічного редактора, який розробляємо – простий веб-додаток з інтуїтивно зрозумілою функціональністю та візуально приємним інтерфейсом. Головною задачею цього графічного редактора є створення мудбордів – візуалізацій проектів. Склали загальну картину по інтерфейсу (необхідні панелі, вікна) та функціоналу (необхідні і допоміжні інструменти для виконання поставленої задачі).

Серед розглянутих мов програмування для виконання розробки обрали JavaScript та Python. Python є найдоступнішим варіантом для розробки веб-додатків і водночас надзвичайно потужним. Це мова програмування з численними можливостями, його фреймворки забезпечують створення ефективної структуризації проекту, легкого доступу до всіх складових та швидкої взаємодії між ними. Крім цього, є можливості підключати бібліотеки та модулі для роботи з графічним інтерфейсом, створювати бази даних й

керувати ними. JavaScript застосували для задання динамічності веб-сторінкам додатка.

Java та C# мають свої переваги, проте їх для даного дипломного проекту не використовували, щоб не створювати надто об’ємної роботи, не перевантажувати структуру і не ускладнювати взаємодію.

Головними вимогами до середовища розробки були швидкість запуску та роботи, зручність написання коду та переходу між різними файлами, відсутність лишніх відволікаючих елементів. Такими характеристиками володіє Sublime Text. До того ж, його можна налаштувати під свої вподобання, наприклад, обрати темну тему для редактора, підключити додаткові плагіни. Sublime Text особливо вирізняється кольоровими варіаціями у підсвітці різних елементів в коді (привіт естетам!). Для розробки веб-додатка обрали саме цей редактор.

В ході розробки розглянули найпопулярніший фреймворк для проектування веб-додатків на Python – Django. На його основі в рамках дипломного проекту вдалось створити архітектуру додатку, взаємозв’язати URL для переходів між веб-сторінками. Здійснили імплементацію бази даних, яку можна використовувати в подальшій розробці для зберігання даних та швидкого доступу до них, зокрема, через панель адміністратора. Розробили інтерфейс веб-додатку з допомогою HTML, CSS, JavaScript.

Метою дипломного проекту були аналіз проблем та можливостей графічних редакторів, аналіз та визначення технологій для розробки веб-додатків, створення програмного рішення для простої та ефективної візуалізації дизайн-проектів. Задана мета у ході виконання роботи виконана і як результат – створено прототип графічного редактора. Він не є кінцевим продуктом, може удосконалюватись для збільшення ефективності і розширення можливостей.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. TOP-10 GRAPHICS EDITORS: ULTIMATE LIST [Електронний ресурс] – Режим доступу: https://bytescout.com/blog/top-graphics-editors.html (дата звернення 18.05.22) – Назва з екрана
2. Що таке Figma: функції, інструменти та переваги [Електронний ресурс] – Режим доступу: https://wezom.academy/ua/chto-takoe-figma-funktsii-instrumenty-ipreimuschestva/ (дата звернення 18.05.22) – Назва з екрана
3. Як створювати візуальний контент за допомогою зручного і дешевого інструменту Canva [Електронний ресурс] – Режим доступу: http://slaidik.com.ua/yak-stvoryuvati-vizualnij-kontent-za-dopomogoyu-zruchnogo-i-deshevogo-instrumentu-canva/ (дата звернення 19.05.22) – Назва з екрана
4. Best Programming Languages to Learn in 2022 [Електронний ресурс] – Режим доступу: https://www.simplilearn.com/best-programming-languages-start-learning-today-article (дата звернення 19.05.22) – Назва з екрана
5. The best programming languages to learn in 2022 [Електронний ресурс] – Режим доступу: https://www.techrepublic.com/article/the-best-programming-languages-to-learn-in-2022/ (дата звернення 19.05.22) – Назва з екрана
6. Що таке Big Data? [Електронний ресурс] – Режим доступу: http://thefuture.news/bigdata (дата звернення 20.05.22) – Назва з екрана
7. Top 11 Python Frameworks for Web Development In 2022 [Електронний ресурс] – Режим доступу: https://www.netsolutions.com/insights/top-10-python-frameworks-for-web-development-in-2019/ (дата звернення 21.05.22) – Назва з екрана

8. The Good and the Bad of C# Programming [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/ (дата звернення 21.05.22) – Назва з екрана
9. Why do we write websites using python? [Електронний ресурс] – Режим доступу: https://directlinedev.com/web-development/why-do-we-write-websites-using-python/ (дата звернення 22.05.22) – Назва з екрана
10. Visual Studio vs. Sublime Text: Which Editor Should You Choose? [Електронний ресурс] – Режим доступу: https://www.codecademy.com/resources/blog/visual-studio-code-sublime-text/ (дата звернення 22.05.22) – Назва з екрана
11. Український веб-довідник [Електронний ресурс] – Режим доступу: https://css.in.ua/ (дата звернення 24.05.22) – Назва з екрана
12. Django documentation [Електронний ресурс] – Режим доступу: https://docs.djangoproject.com/en/4.0/ (дата звернення 25.05.22) – Назва з екрана
13. Important UX Design Principles for Newcomers [Електронний ресурс] – Режим доступу: https://www.springboard.com/blog/design/ux-design-principles/ (дата звернення 21.05.22) – Назва з екрана